# Overview

rtf65004 is a 65C02 compatible superscalar processor with a 64-bit native operating mode. Native mode makes use of a 32-entry register file. Native mode instructions vary in length up to 48-bits.

## 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 to 29 |  |  |  |
| R30 | fp | frame pointer |  |
| R31 | sp | stack pointer |  |

## Memory Addressing

The cpu is word oriented with byte addressable memory. The default size of a memory operand is word size (64 bits). This may be overridden using size prefixes for byte (8 bit), wyde (16 bit) or tetra (32 bit) data. Words and characters must be aligned on appropriate boundaries. Up to 264 Bytes of data are supported and 264 bytes of code.

# 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 | | | | | | Opcode |  | Bytes |
| 47 32 | | 31 24 | 23 | 22 18 | 17 13 | 12 8 | 7 0 |  |  |
|  | | 00h | ~ | Rb5 | Ra5 | Rt5 | 02h | ADD Rt,Ra,Rb | 4 |
|  | | | Imm6 | | Ra5 | Rt5 |  | ADD Rt,Ra,#imm6 | 3 |
|  | | Imm14 | | | Ra5 | Rt5 | 79h | ADD Rt,Ra,#imm14 | 4 |
| Imm30 | | | | | Ra5 | Rt5 | 69h | ADD Rt,Ra,#imm30 | 6 |
|  | | Disp9 | | Rb5 | Ra5 | Rt5 | 75h | ADD Rt,Ra,d,Rb | 4 |
|  | | Addr9 | | Rb5 | Ra5 | Rt5 | 61h | ADD Rt,Ra,(zp,Rb) | 4 |
|  | | Addr9 | | Rb5 | Ra5 | Rt5 | 71h | ADD Rt,Ra,(zp),Rb | 4 |
| Addr25 | | | | Rb5 | Ra5 | Rt5 | 7Dh | ADD Rt,Ra,abs,Rb | 7 |
|  | | | ~ | Rb5 | Ra5 | Rt5 | 72h | ADD Rt,Ra,(Rb) | 3 |
|  | | | Disp9..4 | | Ra5 | Rt5 | 63h | ADD Rt,Ra,d,sp | 3 |

|  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| SUB | Flags: v c n z | | | | | | Opcode |  | Bytes |
| 47 32 | | 31 24 | 23 | 22 18 | 17 13 | 12 8 | 7 0 |  |  |
|  | | 01h | ~ | Rb5 | Ra5 | Rt5 | 02h | SUB Rt,Ra,Rb | 4 |
|  | | Imm14 | | | Ra5 | Rt5 | E7h | SUB Rt,Ra,#imm14 | 4 |
| Imm30 | | | | | Ra5 | Rt5 | F9h | SUB Rt,Ra,#imm30 | 6 |
|  | | Disp9 | | Rb5 | Ra5 | Rt5 | F5h | SUB Rt,Ra,zpg,Rb | 4 |
|  | | Addr9 | | Rb5 | Ra5 | Rt5 | E1h | SUB Rt,Ra,(zp,Rb) | 4 |
|  | | Addr9 | | Rb5 | Ra5 | Rt5 | F1h | SUB Rt,Ra,(zp),Rb | 4 |
| Addr25 | | | | Rb5 | Ra5 | Rt5 | FDh | SUB Rt,Ra,abs,Rb | 7 |
|  | | | ~ | Rb5 | Ra5 | Rt5 | F2h | SUB Rt,Ra,(Rb) | 3 |
|  | | | Disp9..4 | | Ra5 | Rt5 | E3h | SUB Rt,Ra,d,sp | 3 |

|  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| CMP | Flags: c n z | | | | | | | Opcode |  | Bytes |
| 47 32 | | 31 24 | 23 | 22 18 | 17 13 | | 12 8 | 7 0 |  |  |
|  | | 01h | ~ | Rb5 | Ra5 | | 05 | 02h | CMP Ra,Rb | 4 |
|  | | Imm14 | | | Ra5 | | 05 | E7h | CMP Ra,#imm14 | 4 |
| Imm30 | | | | | Ra5 | | 05 | F9h | CMP Ra,#imm30 | 6 |
|  | | Disp9 | | Rb5 | Ra5 | | 05 | F5h | CMP Ra,zpg,Rb | 4 |
|  | | Addr9 | | Rb5 | Ra5 | | 05 | E1h | CMP Ra,(zp,Rb) | 4 |
|  | | Addr9 | | Rb5 | Ra5 | | 05 | F1h | CMP Ra,(zp),Rb | 4 |
| Disp25 | | | | Rb5 | Ra5 | | 05 | FDh | CMP Ra,abs,Rb | 7 |
|  | | | ~ | Rb5 | Ra5 | | 05 | F2h | CMP Ra,(Rb) | 3 |
|  | | | Disp9..4 | | Ra5 | | 05 | E3h | CMP Ra,d,sp | 3 |
|  | | | | | | Imm8 | | C5h | CMP Acc,#imm8 | 2 |

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

|  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| AND | Flags: n z | | | | | | Opcode |  | Bytes |
| 47 32 | | 31 24 | 23 | 22 18 | 17 13 | 12 8 | 7 0 |  |  |
|  | | 03h | ~ | Rb5 | Ra5 | Rt5 | 02h | Rt,Ra,Rb | 4 |
|  | | Imm14 | | | Ra5 | Rt5 | 39h | Rt,Ra,#imm14 | 4 |
| Imm30 | | | | | Ra5 | Rt5 | 29h | Rt,Ra,#imm30 | 6 |
|  | | Disp9 | | Rb5 | Ra5 | Rt5 | 35h | Rt,Ra,zpg,Rb | 4 |
|  | | Addr9 | | Rb5 | Ra5 | Rt5 | 21h | Rt,Ra,(zp,Rb) | 4 |
|  | | Addr9 | | Rb5 | Ra5 | Rt5 | 31h | Rt,Ra,(zp),Rb | 4 |
| Addr25 | | | | Rb5 | Ra5 | Rt5 | 3Dh | Rt,Ra,abs,Rb | 7 |
|  | | | ~ | Rb5 | Ra5 | Rt5 | 32h | Rt,Ra,(Rb) | 3 |
|  | | | Disp9..4 | | Ra5 | Rt5 | 23h | Rt,Ra,d,sp | 3 |

|  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| BIT | Flags: v n z | | | | | | Opcode |  | Bytes |
| 47 32 | | 31 24 | 23 | 22 18 | 17 13 | 12 8 | 7 0 |  |  |
|  | | 03h | ~ | Rb5 | Ra5 | 05 | 02h | Rt,Ra,Rb | 4 |
|  | | Imm14 | | | Ra5 | 05 | 39h | Rt,Ra,#imm14 | 4 |
| Imm30 | | | | | Ra5 | 05 | 29h | Rt,Ra,#imm30 | 6 |
|  | | Disp9 | | Rb5 | Ra5 | 05 | 35h | Rt,Ra,zpg,Rb | 4 |
|  | | Addr9 | | Rb5 | Ra5 | 05 | 21h | Rt,Ra,(zp,Rb) | 4 |
|  | | Addr9 | | Rb5 | Ra5 | 05 | 31h | Rt,Ra,(zp),Rb | 4 |
| Addr25 | | | | Rb5 | Ra5 | 05 | 3Dh | Rt,Ra,abs,Rb | 7 |
|  | | | ~ | Rb5 | Ra5 | 05 | 32h | Rt,Ra,(Rb) | 3 |
|  | | | Disp9..4 | | Ra5 | 05 | 23h | Rt,Ra,d,sp | 3 |

Bit is the AND operation with no target register; the overflow status is set to bit 62 of the memory op

|  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| OR | Flags: n z | | | | | | Opcode |  | Bytes |
| 47 32 | | 31 24 | 23 | 22 18 | 17 13 | 12 8 | 7 0 |  |  |
|  | | 05h | ~ | Rb5 | Ra5 | Rt5 | 02h | Rt,Ra,Rb | 4 |
|  | | Imm14 | | | Ra5 | Rt5 | 19h | Rt,Ra,#imm14 | 4 |
| Imm30 | | | | | Ra5 | Rt5 | 09h | Rt,Ra,#imm30 | 6 |
|  | | Disp9 | | Rb5 | Ra5 | Rt5 | 15h | Rt,Ra,zpg,Rb | 4 |
|  | | Addr9 | | Rb5 | Ra5 | Rt5 | 01h | Rt,Ra,(zp,Rb) | 4 |
|  | | Addr9 | | Rb5 | Ra5 | Rt5 | 11h | Rt,Ra,(zp),Rb | 4 |
| Addr25 | | | | Rb5 | Ra5 | Rt5 | 1Dh | Rt,Ra,abs,Rb | 6 |
|  | | | ~ | Rb5 | Ra5 | Rt5 | 12h | Rt,Ra,(Rb) | 3 |
|  | | | Disp9..4 | | Ra5 | Rt5 | 03h | Rt,Ra,d,sp | 3 |

|  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| EOR | Flags: n z | | | | | | Opcode |  | Bytes |
| 47 32 | | 31 24 | 23 | 22 18 | 17 13 | 12 8 | 7 0 |  |  |
|  | | 04h | ~ | Rb5 | Ra5 | Rt5 | 02h | Rt,Ra,Rb | 4 |
|  | | Imm14 | | | Ra5 | Rt5 | 59h | Rt,Ra,#imm14 | 4 |
| Imm30 | | | | | Ra5 | Rt5 | 49h | Rt,Ra,#imm30 | 6 |
|  | | Disp9 | | Rb5 | Ra5 | Rt5 | 55h | Rt,Ra,zpg,Rb | 4 |
|  | | Addr9 | | Rb5 | Ra5 | Rt5 | 41h | Rt,Ra,(zp,Rb) | 4 |
|  | | Addr9 | | Rb5 | Ra5 | Rt5 | 51h | Rt,Ra,(zp),Rb | 4 |
| Addr25 | | | | Rb5 | Ra5 | Rt5 | 5Dh | Rt,Ra,abs,Rb | 6 |
|  | | | ~ | Rb5 | Ra5 | Rt5 | 52h | Rt,Ra,(Rb) | 3 |
|  | | | Disp9..4 | | Ra5 | Rt5 | 43h | 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 | | | | | | Opcode |  | Bytes |
| 47 32 | | 31 24 | 23 | 22 18 | 17 13 | 12 8 | 7 0 |  |  |
|  | | 05h | ~ | Rb5 | 05 | Rt5 | 02h | Rt,Ra,Rb | 4 |
|  | | Imm14 | | | 05 | Rt5 | 19h | Rt,Ra,#imm14 | 4 |
| Imm30 | | | | | 05 | Rt5 | 09h | Rt,Ra,#imm30 | 6 |
|  | | Disp9 | | Rb5 | 05 | Rt5 | 15h | Rt,Ra,zpg,Rb | 4 |
|  | | Addr9 | | Rb5 | 05 | Rt5 | 01h | Rt,Ra,(zp,Rb) | 4 |
|  | | Addr9 | | Rb5 | 05 | Rt5 | 11h | Rt,Ra,(zp),Rb | 4 |
| Addr25 | | | | Rb5 | 05 | Rt5 | 1Dh | Rt,Ra,abs,Rb | 6 |
|  | | | ~ | Rb5 | 05 | Rt5 | 12h | Rt,Ra,(Rb) | 3 |
|  | | | Disp9..4 | | 05 | Rt5 | 03h | Rt,Ra,d,sp | 3 |

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

|  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| LDA | Flags: n z | | | | | | Opcode |  | Bytes |
| 47 32 | | 31 24 | 23 | 22 18 | 17 13 | 12 8 | 7 0 |  |  |
|  | | 05h | ~ | Rb5 | 05 | 15 | 02h | Rt,Ra,Rb | 4 |
|  | | Imm14 | | | 05 | 15 | 19h | Rt,Ra,#imm14 | 4 |
| Imm30 | | | | | 05 | 15 | 09h | Rt,Ra,#imm30 | 6 |
|  | | Disp9 | | Rb5 | 05 | 15 | 15h | Rt,Ra,zpg,Rb | 4 |
|  | | Addr9 | | Rb5 | 05 | 15 | 01h | Rt,Ra,(zp,Rb) | 4 |
|  | | Addr9 | | Rb5 | 05 | 15 | 11h | Rt,Ra,(zp),Rb | 4 |
| Addr25 | | | | Rb5 | 05 | 15 | 1Dh | Rt,Ra,abs,Rb | 6 |
|  | | | ~ | Rb5 | 05 | 15 | 12h | Rt,Ra,(Rb) | 3 |
|  | | | Disp9..4 | | 05 | 15 | 03h | Rt,Ra,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 | | | | | | Opcode |  | Bytes |
| 47 32 | | 31 24 | 23 | 22 18 | 17 13 | 12 8 | 7 0 |  |  |
|  | | 05h | ~ | Rb5 | 05 | 25 | 02h | Rt,Ra,Rb | 4 |
|  | | Imm14 | | | 05 | 25 | 19h | Rt,Ra,#imm14 | 4 |
| Imm30 | | | | | 05 | 25 | 09h | Rt,Ra,#imm30 | 6 |
|  | | Disp9 | | Rb5 | 05 | 25 | 15h | Rt,Ra,zpg,Rb | 4 |
|  | | Addr9 | | Rb5 | 05 | 25 | 01h | Rt,Ra,(zp,Rb) | 4 |
|  | | Addr9 | | Rb5 | 05 | 25 | 11h | Rt,Ra,(zp),Rb | 4 |
| Addr25 | | | | Rb5 | 05 | 25 | 1Dh | Rt,Ra,abs,Rb | 6 |
|  | | | ~ | Rb5 | 05 | 25 | 12h | Rt,Ra,(Rb) | 3 |
|  | | | Disp9..4 | | 05 | 25 | 03h | Rt,Ra,d,sp | 3 |

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

|  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| LDY | Flags: n z | | | | | | Opcode |  | Bytes |
| 47 32 | | 31 24 | 23 | 22 18 | 17 13 | 12 8 | 7 0 |  |  |
|  | | 05h | ~ | Rb5 | 05 | 35 | 02h | Rt,Ra,Rb | 4 |
|  | | Imm14 | | | 05 | 35 | 19h | Rt,Ra,#imm14 | 4 |
| Imm30 | | | | | 05 | 35 | 09h | Rt,Ra,#imm30 | 6 |
|  | | Disp9 | | Rb5 | 05 | 35 | 15h | Rt,Ra,zpg,Rb | 4 |
|  | | Addr9 | | Rb5 | 05 | 35 | 01h | Rt,Ra,(zp,Rb) | 4 |
|  | | Addr9 | | Rb5 | 05 | 35 | 11h | Rt,Ra,(zp),Rb | 4 |
| Addr25 | | | | Rb5 | 05 | 35 | 1Dh | Rt,Ra,abs,Rb | 6 |
|  | | | ~ | Rb5 | 05 | 35 | 12h | Rt,Ra,(Rb) | 3 |
|  | | | Disp9..4 | | 05 | 35 | 03h | Rt,Ra,d,sp | 3 |

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

|  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| ST | Flags: n z | | | | | | Opcode |  | Bytes |
| 47 32 | | 31 24 | 23 | 22 18 | 17 13 | 12 8 | 7 0 |  |  |
|  | | Disp9 | | Rb5 | Ra5 | 05 | 95h | Ra,zpg,Rb | 4 |
|  | | Addr9 | | Rb5 | Ra5 | 05 | 81h | Ra,(zp,Rb) | 4 |
|  | | Addr9 | | Rb5 | Ra5 | 05 | 91h | Ra,(zp),Rb | 4 |
| Addr25 | | | | Rb5 | Ra5 | 05 | 9Dh | Ra,abs,Rb | 6 |
|  | | | ~ | Rb5 | Ra5 | 05 | 92h | Ra,(Rb) | 3 |
|  | | | Disp9..4 | | Ra5 | 05 | 83h | Ra,d,sp | 3 |

|  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| STA | Flags: n z | | | | | | Opcode |  | Bytes |
| 47 32 | | 31 24 | 23 | 22 18 | 17 13 | 12 8 | 7 0 |  |  |
|  | | Disp9 | | Rb5 | 15 | 05 | 95h | Ra,zpg,Rb | 4 |
|  | | Addr9 | | Rb5 | 15 | 05 | 81h | Ra,(zp,Rb) | 4 |
|  | | Addr9 | | Rb5 | 15 | 05 | 91h | Ra,(zp),Rb | 4 |
| Addr25 | | | | Rb5 | 15 | 05 | 9Dh | Ra,abs,Rb | 6 |
|  | | | ~ | Rb5 | 15 | 05 | 92h | Ra,(Rb) | 3 |
|  | | | Disp9..4 | | 15 | 05 | 83h | Ra,d,sp | 3 |

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

|  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| STX | Flags: n z | | | | | | Opcode |  | Bytes |
| 47 32 | | 31 24 | 23 | 22 18 | 17 13 | 12 8 | 7 0 |  |  |
|  | | Disp9 | | Rb5 | 25 | 05 | 95h | Ra,zpg,Rb | 4 |
|  | | Addr9 | | Rb5 | 25 | 05 | 81h | Ra,(zp,Rb) | 4 |
|  | | Addr9 | | Rb5 | 25 | 05 | 91h | Ra,(zp),Rb | 4 |
| Addr25 | | | | Rb5 | 25 | Rt5 | 05 | Ra,abs,Rb | 6 |
|  | | | ~ | Rb5 | 25 | 05 | 92h | Ra,(Rb) | 3 |
|  | | | Disp9..4 | | 25 | Rt5 | 05 | Ra,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: n z | | | | | | Opcode |  | Bytes |
| 47 32 | | 31 24 | 23 | 22 18 | 17 13 | 12 8 | 7 0 |  |  |
|  | | Disp9 | | Rb5 | 35 | 05 | 95h | Ra,zpg,Rb | 4 |
|  | | Addr9 | | Rb5 | 35 | 05 | 81h | Ra,(zp,Rb) | 4 |
|  | | Addr9 | | Rb5 | 35 | 05 | 91h | Ra,(zp),Rb | 4 |
| Addr25 | | | | Rb5 | 35 | 05 | 9Dh | Ra,abs,Rb | 6 |
|  | | | ~ | Rb5 | 35 | 05 | 92h | Ra,(Rb) | 3 |
|  | | | Disp9..4 | | 35 | 05 | 83h | Ra,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: n z | | | | | | Opcode |  | Bytes |
| 47 32 | | 31 24 | 23 | 22 18 | 17 13 | 12 8 | 7 0 |  |  |
|  | | Disp9 | | Rb5 | 05 | 05 | 95h | Ra,zpg,Rb | 4 |
|  | | Addr9 | | Rb5 | 05 | 05 | 81h | Ra,(zp,Rb) | 4 |
|  | | Addr9 | | Rb5 | 05 | 05 | 91h | Ra,(zp),Rb | 4 |
| Addr25 | | | | Rb5 | 05 | 05 | 9Dh | Ra,abs,Rb | 6 |
|  | | | ~ | Rb5 | 05 | 05 | 92h | Ra,(Rb) | 3 |
|  | | | Disp9..4 | | 05 | 05 | 83h | Ra,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 | | | | | | Opcode |  | Bytes |
| 47 32 | | 31 24 | 23 | 22 18 | 17 13 | 12 8 | 7 0 |  |  |
|  | |  |  |  |  |  | 0Ah | Acc | 1 |
|  | | ??h | ~ | Rb5 | Ra5 | Rt5 | 02h | Rt,Ra,Rb | 4 |
|  | | | Imm6 | | Ra5 | Rt5 | 24h | Rt,Ra,#imm6 | 3 |
|  | | Disp9 | | Rb5 | Ra5 | Imm5 | 16h | zpg,Rb,#imm | 4 |
| Addr25 | | | | Rb5 | Ra5 | Imm5 | 1Eh | abs,Rb,#imm | 6 |

|  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| ROL | Flags: c n z | | | | | | Opcode |  | Bytes |
| 47 32 | | 31 24 | 23 | 22 18 | 17 13 | 12 8 | 7 0 |  |  |
|  | |  |  |  |  |  | 2Ah | Acc | 1 |
|  | | ??h | ~ | Rb5 | Ra5 | Rt5 | 02h | Rt,Ra,Rb | 4 |
|  | | Disp9 | | Rb5 | Ra5 | Imm5 | 36h | Rt,Ra,zpg,Rb | 4 |
| Addr25 | | | | Rb5 | Ra5 | Imm5 | 3Eh | Rt,Ra,abs,Rb | 6 |

|  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| LSR | Flags: c n z | | | | | | Opcode |  | Bytes |
| 47 32 | | 31 24 | 23 | 22 18 | 17 13 | 12 8 | 7 0 |  |  |
|  | |  |  |  |  |  | 4Ah | Acc | 1 |
|  | | ??h | ~ | Rb5 | Ra5 | Rt5 | 02h | Rt,Ra,Rb | 4 |
|  | | | Imm6 | | Ra5 | Rt5 | 34h | Rt,Ra,#imm6 | 3 |
|  | | Disp9 | | Rb5 | Ra5 | Imm5 | 56h | Rt,Ra,zpg,Rb | 4 |
| Addr25 | | | | Rb5 | Ra5 | Imm5 | 5Eh | Rt,Ra,abs,Rb | 6 |

|  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| ROR | Flags: c n z | | | | | | Opcode |  | Bytes |
| 47 32 | | 31 24 | 23 | 22 18 | 17 13 | 12 8 | 7 0 |  |  |
|  | |  |  |  |  |  | 6Ah | Acc | 1 |
|  | | ??h | ~ | Rb5 | Ra5 | Rt5 | 02h | Rt,Ra,Rb | 4 |
|  | | Disp9 | | Rb5 | Ra5 | Imm5 | 76h | Rt,Ra,zpg,Rb | 4 |
| Addr25 | | | | Rb5 | Ra5 | Imm5 | 7Eh | Rt,Ra,abs,Rb | 6 |

|  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| INC | Flags: n z | | | | | | | Opcode |  | Bytes |
| 47 32 | | 31 24 | 23 | 22 18 | 17 13 | | 12 8 | 7 0 |  |  |
|  | |  |  |  |  | |  | 1Ah | Acc | 1 |
|  | |  | | | | Imm3 | Rt5 | E6h | Rt,Ra,Rb | 2 |
|  | | Disp9 | | Rb5 | Ra5 | | Imm5 | F6h | Rt,Ra,zpg,Rb | 4 |
| Addr25 | | | | Rb5 | Ra5 | | Imm5 | FEh | Rt,Ra,abs,Rb | 6 |

|  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| DEC | Flags: n z | | | | | | | Opcode |  | Bytes |
| 47 32 | | 31 24 | 23 | 22 18 | 17 13 | | 12 8 | 7 0 |  |  |
|  | |  |  |  |  | |  | 3Ah | Acc | 1 |
|  | |  | | | | Imm3 | Rt5 | C6h | Rt,Ra,Rb | 2 |
|  | | Disp9 | | Rb5 | Ra5 | | Imm5 | D6h | Rt,Ra,zpg,Rb | 4 |
| Addr25 | | | | Rb5 | Ra5 | | Imm5 | DEh | Rt,Ra,abs,Rb | 6 |

## 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 |
|  | |  | Immediate32 | | | | | | | A9h | LDA #i32 | 5 |
|  | |  | | | Immediate16 | | | | | B9h | LDA #i16 | 3 |
|  | |  | | | | | Immediate8 | | | A5h | LDA #i8 | 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 |
|  | |  | Immediate32 | | | | | | A2h | LDX #i32 | 5 |
|  | |  | | | Immediate16 | | | | B2h | LDX #i16 | 3 |
|  | |  | | | | Immediate8 | | | A6h | LDX #i8 | 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 |
| DEX | Flags: n z | | | | | | | | CAh | DEX | 1 |
| 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 |
|  | | Address40 | | | | | 5Ch | JMP abs | 6 |
|  | |  | | Address16 | | | 4Ch | JMP abs | 3 |
|  | | Address40 | | | | | 6Ch | JMP (abs) | 6 |
|  | | Address40 | | | | | 7Ch | JMP (abs,x) | 6 |
|  | |  |  | | 0 | Ra | D2h | JMP (Rn) | 2 |
| JSR | Flags: | | | | | | | | Bytes |
|  | | Address40 | | | | | 22h | JSR abs | 6 |
|  | |  | | Address16 | | | 20h | JSR abs | 3 |
|  | | Address40 | | | | | FCh | JSR (abs,x) | 6 |
|  | |  |  | | 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 |
| BEQ | Flags: | | | Disp8 | F0h | BEQ disp | 2 |
| BNE | Flags: | | | Disp8 | D0h | BNE disp | 2 |
| BPL | Flags: | | | Disp8 | 10h | BPL disp | 2 |
| BMI | Flags: | | | Disp8 | 30h | BMI disp | 2 |
| BVS | Flags: | | | Disp8 | 70h | BVS disp | 2 |
| BVC | Flags: | | | Disp8 | 50h | BVC disp | 2 |
| BCS | Flags: | | | Disp8 | B0h | BCS disp | 2 |
| BCC | Flags: | | | Disp8 | 90h | BCC disp | 2 |
| BHI | Flags: | | | Disp8 | 13h | BHI disp | 2 |
| BLS | Flags: | | | Disp8 | 33h | BLS disp | 2 |
| BLT | Flags: | | | Disp8 | B3h | BLT disp | 2 |
| BLE | Flags: | | | Disp8 | F3h | BLE disp | 2 |
| BGT | Flags: | | | Disp8 | D3h | BGT disp | 2 |
| BGE | Flags: | | | Disp8 | 93h | BGE disp | 2 |
| BEQ | Flags: | Disp16 | | 00h | F0h | BEQ disp | 4 |
| BNE | Flags: | Disp16 | | 00h | D0h | BNE disp | 4 |
| BPL | Flags: | Disp16 | | 00h | 10h | BPL disp | 4 |
| BMI | Flags: | Disp16 | | 00h | 30h | BMI disp | 4 |
| BVS | Flags: | Disp16 | | 00h | 70h | BVS disp | 4 |
| BVC | Flags: | Disp16 | | 00h | 50h | BVC disp | 4 |
| BCS | Flags: | Disp16 | | 00h | B0h | BCS disp | 4 |
| BCC | Flags: | Disp16 | | 00h | 90h | BCC disp | 4 |
| BRL | Flags: | | Disp16 | | 82h | BRL disp | 3 |

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

## 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 | Flags: | ~3 | Ra5 | 0Bh | PUSH Ra | 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 | Flags: | ~3 | Rt5 | 2Bh | POP Rt | 2 |

## Status Register Operations

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

## Register to register transfer short forms.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| TAX | Flags: n z | AAh | TAX | 1 |
| TXA | Flags: n z | 8Ah | TXA | 1 |
| TAY | Flags: n z | A8h | TAY | 1 |
| TYA | Flags: n z | 98h | TYA | 1 |
| TAS | Flags: | 1Bh | TAS | 1 |
| TSA | Flags: n z | 3Bh | TSA | 1 |
| TYX | Flags: n z | BBh | TYX | 1 |
| TXY | Flags: n z | 9Bh | TXY | 1 |
| TSX | Flags: n z | BAh | TSX | 1 |
| TXS | Flags: | 9Ah | TXS | 1 |

## 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 |
| WYDE | 97h | Cause a signed (16 bit) load / store |
| UWYDE | B7h | Causes an unsigned (16 bit) load / store |
| TETRA | C7h | Causes a signed 32 bit op |
| UTETRA | D7h | Causes an unsigned 32 bit op |

# Opcode Map – Native 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 | WYDE | 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 | UWYDE | 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 | TETRA | INY | CMP #i32 | DEX | WAI | CPY abs | ~ | DEC abs | ~ |
| D- | BNE disp | BXZ | JMP (r) | BGT | ~ |  | DEC d,r | UTETRA | 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 | ~ |

## Micro-op Bundle Layout

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| Len4 | Flag Mask8 | whflg2 | Instr024 | Instr124 | Instr224 | Instr324 |

## Micro-op Instruction Format

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Opcode8 | Cnst4 | Src24 | Src14 | Tgt4 |

## Micro-op Fields

|  |  |
| --- | --- |
| Tgt4 | Meaning |
| 0 | The value 0 |
| 1 | Get from instruction bits 8 to 12 |
| 2 | The accumulator register |
| 3 | The X register |
| 4 | The Y Register |
| 5 | The stack pointer |
| 6 | the tmp register |
| 7 | the SR register |

|  |  |
| --- | --- |
| Src14 | Meaning |
| 0 | The value 0 |
| 1 | Get from register spec instruction bits 13 to 17 |
| 2 | The accumulator register |
| 3 | The X register |
| 4 | The Y Register |
| 5 | The stack pointer |
| 6 | the tmp register |
| 7 | the SR register |
| 12 | Get from register spec instruction bits 13 to 17, bitwise or with instruction bits 8 to 12 |

|  |  |
| --- | --- |
| Src24 | Meaning |
| 0 | The value 0 |
| 1 | Get from register spec instruction bits 18 to 22 |
| 2 | The accumulator register |
| 3 | The X register |
| 4 | The Y Register |
| 5 | The stack pointer |
| 6 | the tmp register |
| 7 | the SR register |
| 9 | the value 1 |
| 11 | the value 1,2,4 or 8 depending on size |
| 15 | the value -1 |

|  |  |  |
| --- | --- | --- |
| Cnst4 | Native Mode | Emulation mode |
| 0 | the value 0 | the value 0 |
| 1 | the value 1 | the value 1 |
| 2 | the value 2 | the value 2 |
| 3 | the value 3 | the value 3 |
| 4 |  |  |
| 5 |  |  |
| 6 | if size <> 0 then -1 else 0 |  |
| 7 | bits 18 to 23 shifted left by 4 | bits 8 to 31 of instruction |
| 8 | bits 18 to 31 of instruction | bits 8 to 15 of instruction |
| 9 | bits 18 to 47 of instruction | bits 8 to 23 of instruction |
| 10 | bits 23 to 31 of instruction |  |
| 11 | bit 23 to 48 of instruction |  |
| 13 | the value -3 | the value -3 |
| 14 | the value -2 | the value -2 |
| 15 | the value -1 | the value -1 |

Micro-op Lists for Instructions

MVN

LD tmp,[X]

ST tmp,[Y]

ADD X,X,8

ADD Y,Y,8

SUB AC,AC,#1

BNE PC

MVP

LD tmp,[X]

ST tmp,[Y]

SUB X,X,8

SUB Y,Y,8

SUB AC,AC,#1

BNE PC

MVC

ST X,[Y]

ADD Y,Y,#8

SUB AC,AC,#1

BNE PC

CMPS

LD tmp1,[X]

LD tmp2,[Y]

ADD X,X,#8

ADD Y,Y,#8

SUB ac,ac,#1

BEQ PC+1

CMP tmp1,tmp2

BEQ PC

ROR zp,x,Ra

LD tmp,zp[X]

ROR tmp,Ra|#

ST tmp,zp[X]