Opcode Map – 8 bit mode W65C816 compatible

|  |  |
| --- | --- |
|  | = W65C816S instructions |

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

Opcode Map – Page 2 Opcodes

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  | -0 | -1 | -2 | -3 | -4 | -5 | -6 | -7 | -8 | -9 | -A | -B | -C | -D | -E | -F |
| 0- | BRK2 | ORA {d,x} |  |  |  |  |  |  |  |  | ASR acc | PHDS | TSB xlabs | ORA xlabs | ASL xlabs |  |
| 1- | BGT disp | ORA {d},y | ORA {d} | ORA {d,s},y |  |  |  |  | CMC | OR xlabs,y |  | CS: | TRB xlabs | ORA xlabs,x | ASL xlabs,x |  |
| 2- | JSR [{d},y] | AND {d,x} | JSF seg:offs |  |  |  |  |  |  |  |  | PLDS | BIT xlabs | AND xlabs | ROL xlabs |  |
| 3- | BLT disp | AND {d},y | AND {d} | AND {d,s},y |  |  |  |  |  | AND xlabs,y |  | SEG: | BIT xlabs,x | AND xlabs,x | ROL xlabs,x |  |
| 4- |  | EOR {d,x} | WDM2 |  | STS |  |  |  |  |  |  | PHCS |  | EOR xlabs | LSR xlabs |  |
| 5- |  | EOR {d},y | EOR {d} | EOR {d,s},y |  |  |  |  |  | EOR xlabs,y |  | SEG0: | JMF seg:offs | EOR xlabs,x | LSR xlabs,x |  |
| 6- |  | ADC {d,x} |  |  |  |  |  |  |  |  |  | RTF |  | ADC xlabs | ROR xlabs |  |
| 7- |  | ADC {d},y | ADC {d} | ADC {d,s},y |  |  |  |  |  | ADC xlabs,y |  | IOS: | JML [xlabs,x] | ADC xlabs,x | ROR xlabs,x |  |
| 8- |  | STA {d,x} |  |  |  |  |  |  | DEY4 |  |  |  | STY xlabs | STA xlabs | STX xlabs |  |
| 9- | BGE disp | STA {d},y | STA {d} | STA {d,s},y |  |  |  |  |  | STA xlabs,y |  |  | STZ xlabs | STA xlabs,x | STZ xlabs,x |  |
| A- |  | LDA {d,x} |  |  |  |  |  |  |  |  |  |  | LDY xlabs | LDA xlabs | LDX xlabs |  |
| B- | BLE disp | LDA {d},y | LDA {d} | LDA {d,s},y |  |  |  |  |  | LDA xlabs,y |  |  | LDY xlabs,x | LDA xlabs,x | LDX xlabs,x |  |
| C- |  | CMP {d,x} |  |  |  |  |  |  | INY4 |  | DEX4 |  | CPY xlabs | CMP xlabs | DEC xlabs |  |
| D- |  | CMP {d},y | CMP {d} | CMP {d,s},y | PEA { } |  |  |  |  | CMP xlabs,y |  | CLK | JML [xlabs] | CMP xlabs,x | DEC xlabs,x |  |
| E- | CACHE # | SBC{d,x} |  |  |  |  |  |  | INX4 |  | NOP2 |  | CPX xlabs | SBC xlabs | INC xlabs |  |
| F- | PCHIST | SBC {d},y | SBC{d} | SBC {d,s},y | PEA xlabs |  |  |  |  | SBC xlabs,y |  |  | JSL [xlabs,x] | SBC xlabs,x | INC xlabs,x |  |

Support for 32 bit indirect addresses. These are denoted with { } characters.

Support for 32 bit absolute and indexed addresses.

CMC = complement carry flag

INY4,DEY4,INX4,DEX4 – increment or decrement the X or Y registers by a value of four.

BGT,BGE,BLT,BLE are signed branch versions for greater than, greater than or equal, less than or less than or equal

CS: - forces use of the code segment register value rather than the data segment

SEG0: - forces the segment value zero to be used during address calculations.

IOS: - forces the segment value to $FFD00000 an address range reserved for I/O

SEG: - forces use of the specified segment value for address calculations

PHDS – pushes the data segment on the stack

PLDS – pulls the data segment from the stack

PHCS – pushes the code segment on the stack

JSF – Jump to Subroutine Far, allows specification of a new segment when calling a subroutine. Both the code segment and program counter value are pushed onto the stack.

JMF – Jump Far allows specification of a new segment when jumping to a target address. The special segment value $FFFFFFFF causes a switch to 8 bit emulation mode. The special segment value $FFFFFFFE causes a switch to 16 bit emulation mode.

RTF – does a far return, popping both the code segment and program counter from the stack.

{ } forces use of a four byte indirect address

CACHE – issues a command to the cache. Currently only two commands are supported

00 – invalidate entire instruction cache, and 01 – invalidate instruction cache line identified by accumulator