|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **opcode** | **x0** | **x1** | **x2** | **x3** | **x4** | **x5** | **x6** | **x7** | **x8** | **x9** | **xA** | **xB** | **xC** | **xD** | **xE** | **xF** |
| **0x** | SUB abs | SUB imm | SUB abs,X |  | ADD abs | ADD imm | ADD abs,X |  | CMP abs | CMP imm | CMP abs,X |  | NOR abs | NOR imm | NOR abs,X |  |
| **1x** | LDA abs | LDA imm | LDA abs,X |  | STA abs | STA imm | STA abs,X |  | LDX abs | LDX imm | CPX abs | CPX imm | STX abs |  |  |  |
| **2x** |  |  | BNE  abs |  |  |  | BEQ abs |  |  |  | BCC abs |  |  |  | BCS abs |  |
| **3x** | PLA | PLX | RETURN |  | PHA | PHX | JMP abs | CALL abs | INX | DEX |  |  |  |  |  |  |

**Addressing Modes:**

imm immediate LDA #$1F operand is literal byte $1F

abs absolute LDA $1FF operand is contents of address $1FF

abs,X absolute, X-indexed LDA $1FF,X operand is contents of address formed by adding $1FF to the value in the X register

impl implied INX operand is implied by the instruction

**Encoding:**

The instruction’s opcode is packed into the most significant six bits of a program byte. Instructions with no operands (implied addressing) require only a single program byte. Address operands are 10 bits, formed from the least significant two bits of the first program byte, and all eight bits of the second program byte. Immediate operands are 8 bits, taken from the second program byte.

**Programmer-Visible Registers:**

PC program counter (10 bit)

SP stack pointer (6 bit)

A accumulator (8 bit)

X index register (8 bit)

SR status register [carry, zero] (2 bit)

**Processor Stack:**

LIFO, top down, 64 entry, $3C0 - $3FF

**Instructions:**

**ADD** add to accumulator with carry-out

A, C <- A + OPERAND

status flags affected: C, Z

**BCC** branch if carry flag is clear

PC <- OPERAND if C = 0

status flags affected: none

**BCS** branch if carry flag is set

PC <- OPERAND if C = 1

status flags affected: none

**BEQ** branch is zero flag is set

PC <- OPERAND if Z = 1

status flags affected: none

**BNE** branch is zero flag is clear

PC <- OPERAND if Z = 0

status flags affected: none

**CALL** push return address onto stack, and branch to new location

(SP) <- PC

PC <- OPERAND

status flags affected: none

**CMP** compare with accumulator

A – OPERAND

status flags affected: C, Z

**CPX** compare with X register

X – OPERAND

status flags affected: C, Z

**DEX** decrement X register

X <- X – 1

status flags affected: Z

**INX** increment X register

X <- X + 1

status flags affected: Z

**JMP** branch to new location

PC <- OPERAND

status flags affected: none

**LDA** load accumulator

A <- OPERAND

status flags affected: Z

**LDX** load X register

X <- OPERAND

status flags affected: Z

**NOR** bitwise nor with accumulator

A <- A NOR OPERAND

status flags affected: Z

**PHA** push accumulator onto stack

(SP) <- A

status flags affected: none

**PHX** push X register onto stack

(SP) <- X

status flags affected: none

**PLA** pull stack value into accumulator

A <- (SP)

status flags affected: none

**PLX** pull stack value into X register

X <- (SP)

status flags affected: none

**RETURN** pull stack value into program counter, and branch to new location

PC <- (SP)

status flags affected: none

**STA** store accumulator

OPERAND <- A

status flags affected: none

**STX** store X register

OPERAND <- X

status flags affected: none

**SUB** subtract from accumulator with carry-out

A, C <- A - OPERAND

status flags affected: C,Z