Pixie Virtual Machine

# Overview

The PixieVM CPU is a 16-bit Big Endian CPU with 4 16-bit general purpose registers A, B, C, D, one 16-bit index register X, a 16-bit stack pointer SP, a 16-bit instruction pointer IP and a 16-bit flags register FLAGS. FLAGS holds various CPU state flags ie. negative, overflow, break, interrupt disable, zero and carry in the following format:

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| - | - | - | - | - | - | - | - | N | V | - | - | B | I | Z | C |

Each general purpose register can be referred to by it's high and low byte:

|  |  |  |  |
| --- | --- | --- | --- |
| AX | BX | CX | DX |
| |  |  | | --- | --- | | AH | AL | | |  |  | | --- | --- | | BH | BL | | |  |  | | --- | --- | | CH | CL | | |  |  | | --- | --- | | DH | DL | |

# Addressing Modes

The PixieVM CPU has a very orthogonal instruction set. The table below describes the various addressing modes supported with examples of their usage.

|  |  |  |
| --- | --- | --- |
| Mode | Description | Example |
| RR8 | 8-bit source and destination register | xor ch, ch |
| RR16 | 16-bit source and destination register | xor d, d |
| RM8 | 8-bit destination register, memory source | mov al, [d+x] |
| RM16 | 16-bit destination register, memory source | mov a, [b+x] |
| RA8 | 8-bit destination register, absolute source | mov bl, [$c000] |
| RA16 | 16-bit destination register, absolute source | mov a, [$ffe0] |
| RI8 | 8-bit destination register, 8-bit immediate source | mov dh, $c0 |
| RI16 | 16-bit destination register, 16-bit immediate source | mov d, $8000 |
| MR8 | memory destination, 8-bit register source | mov [d+x], al |
| MR16 | memory destination, 16-bit register source | mov [a+x], b |
| M8I8 | 8-bit memory destination, 8-bit immediate source | mov BYTE [a+x], $ff |
| M16I8 | 16-bit memory destination, 8-bit immediate source | mov WORD [a+x], $ff |
| MI16 | 16-bit memory destination, 16-bit immediate source | mov [b+x], $c000 |
| AR8 | 16-bit absolute destination, 8-bit register source | mov [$ce00], al |
| AR16 | 16-bit absolute destination, 16-bit register source | mov [$cd00], b |
| A8I8 | 8-bit absolute destination, 8-bit immediate source | mov BYTE [$e000], $1 |
| A16I8 | 16-bit absolute destination, 8-bit immediate source | mov WORD [$e000], $1 |
| AI16 | 16-bit absolute destination, 16-bit immediate source | mov [$0800], $ffff |
| R8 | 8-bit register | inc al |
| R16 | 16-bit register | inc b |
| M8 | 8-bit memory | dec BYTE [a+x] |
| M16 | 16-bit memory | dec WORD [a+x] |
| A8 | 8-bit absolute | dec BYTE [$fe00] |
| A16 | 16-bit absolute | inc WORD [$cd00] |
| IMPLIED | Implied | sei |
| I8 | Immediate 8-bit | push $c0 |
| I16 | Immediate 16-bit | jmp $8000 |

# Instructions

The CPU instructions are grouped into 7 different categories based on how they are used. Each group defines a set of addressing modes supported by that instruction.

|  |  |
| --- | --- |
| Address Group | Addressing modes |
| I | RR8, RR16, RM8, RM16, RA8, RA16, RI8,  RI16, MR8, MR16, M8I8, M16I8, MI16, AR8, AR16, A8I8, A16I8, AI16 |
| II | R8, R16, M8, M16, A8, A16 |
| III | IMPLIED |
| IV | R16, M16, A16, I16 |
| V | I8 |
| VI | R8, R16, M8, M16, A8, A16 |
| VII | R8, R16, M8, M16, A8, A16, I8, I16 |

The following table lists each CPU instruction mnemonic followed by what addressing group it belongs to and a brief description.

|  |  |  |
| --- | --- | --- |
| Instruction Mnemonic | Address Group | Description |
| ADC | I | Add with carry |
| AND | I | Bitwise AND |
| BIT | I | Test bits |
| BRK | III | Break execution |
| CALL | IV | Call subroutine |
| CLC | III | Clear carry flag |
| CLI | III | Clear interrupt flag |
| CMP | I | Compare |
| DEC | II | Decrement integer |
| DEX | III | Decrement X register |
| INC | II | Increment integer |
| INX | III | Increment X register |
| IRET | III | Return from interrupt |
| JCC | V | Jump on carry clear |
| JCS | V | Jump on carry set |
| JMI | V | Jump on negative flag set |
| JMP | IV | Unconditional jump |
| JNZ | V | Jump on result not zero |
| JPL | V | Jump on negative flag clear |
| JVC | V | Jump on overflow clear |
| JVS | V | Jump on overflow set |
| JZ | V | Jump on zero flag set |
| MOV | I | Move data |
| NOP | III | No operation |
| OR | I | Bitwise OR |
| POP | VI | Pop from stack |
| POPA | III | Pop from stack into .A |
| POPF | III | Pop from stack into FLAGS |
| PUSH | VII | Push onto stack |
| PUSHA | III | Push .A onto stack |
| PUSHF | III | Push FLAGS onto stack |
| RET | III | Return from subroutine call |
| ROL | II | Rotate bits left |
| ROR | II | Rotate bits right |
| SBB | I | Subtract with borrow |
| SEC | III | Set carry flag |
| SEI | III | Set interrupt-disable flag |
| SHL | II | Shift bits left |
| SHR | II | Shift bits right |
| XOR | I | Bitwise exclusive-OR |

# Relative Branching

Address group V is the group of relative branching instructions. Relative branching refers to the property that instruction execution will take place relative to the current IP register. This is distinct from absolute branching, where branching instructions such as JMP are given an absolute address. This group includes the following instructions:

|  |
| --- |
| Relative Branching Instructions |
| JCC |
| JCS |
| JMI |
| JNZ |
| JPL |
| JVC |
| JVS |
| JZ |

Each relative branching instruction takes a single 8-bit byte argument. The argument encodes the offset to branch to if the instruction condition is true. The 8-bit argument is treated as a signed value. Since a signed 8-bit byte has a range of -128 to +127, a negative value will move the IP register backwards up to 128 bytes. A positive value will move the IP register forward up to 127 bytes.