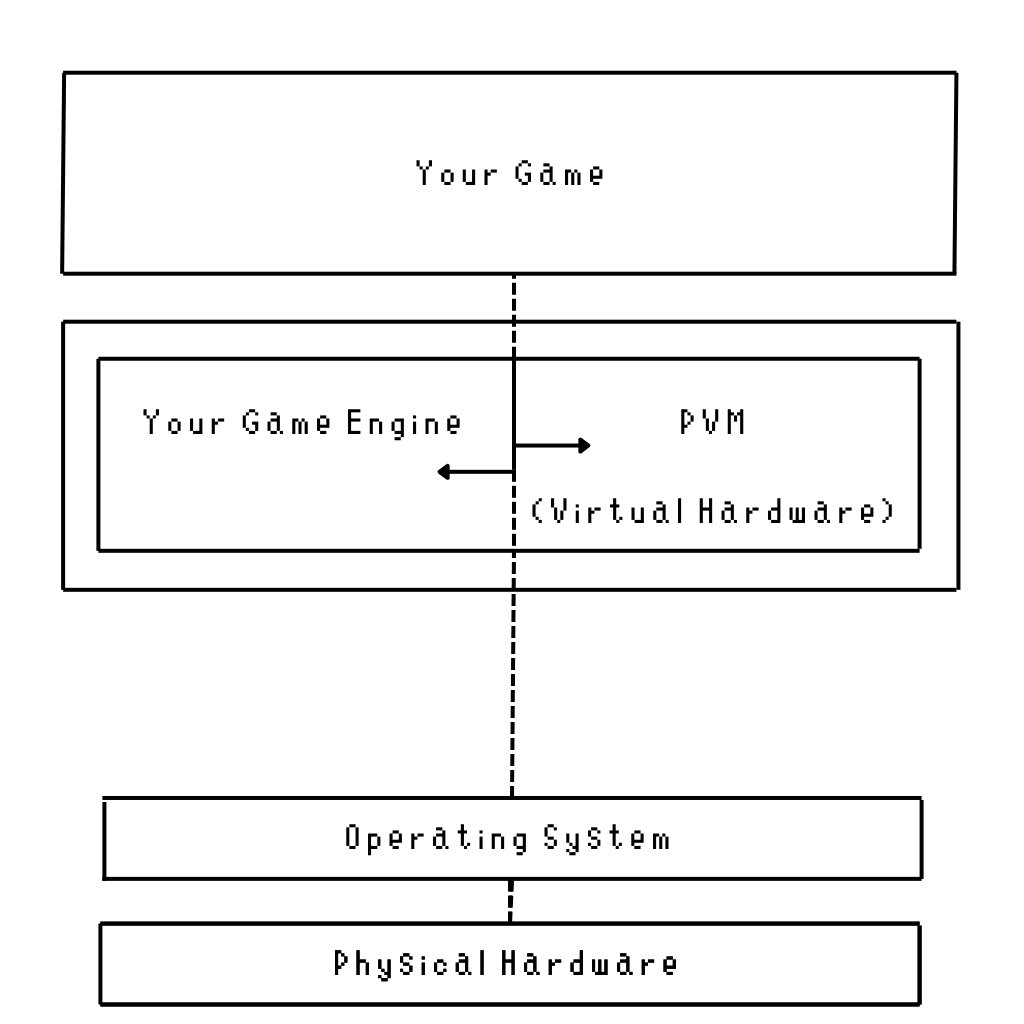
ZVM Design

FZ Interactive

1. Intro

What is ZVM ? ZVM (Zero Virtual Machine) is a Virtual Machine, kind of. ZVM is an open source tool made with the purpose of being incorporated into game engines. ZVM is made up of emulated hardware components, A CPU (ZPU), Temporary Memory, and Stored Memory. The Zero Virtual Machine runs a custom 64 bit instruction set (ZCode). ZVM as a stand alone project does not do very much, It cannot display graphics or play sound, It was made with the purpose of being extended and tailored to your needs.

1. Hardware Specs

CPU:

64 Bit Processor - (ZPU)

32 General Purpose Registers (r0-r31) (64 Bits)

Stack Pointer - SP

Stack Bank - Standard Size (16MB) *Can be Changed if you need a larger stack*

SZC - Standard ZCode Instructions

* Memory IO
* Bit Shifting
* Branching
* Stack Control
* Float and Integer Math Operations (Addition, Subtraction, Multiplication, Division)
* Vector Operations (FLOAT ONLY)

Memory:

Temporary:

Clears after the program instance is done with execution.

When a program is loaded into memory it is loaded into (TRAM).

Size - Standard (128mb) *Can and should be altered.*

Stored (Global):

Never clears, and is accessible to all programs.

Can be exported to a file for usage after the game engine process exits.

Size - Standard (16mb) *Can and should be altered.*

TRAMMemory Map:

0x**00000000 -** 0x00000064 - PVM Control Space.

0x00000064 - 0x00002864 - Variable Space

0x00002864 - 0x01000000 - Free Memory ( Programs start at 0x00002864 )

TRAM Special Address:

0x**00000001 - Program Exit Flag**

0x**00000002 - Generate Random Number Flag**

0x**00000003 - Random Number Min (int)**

0x**00000004 - Random Number Max (int)**

0x**00000005 - Random Number Result (int)**

**0x00000006 - Stack Pointer**

SRAMMemory Map:

All Free Memory

2. ZPU Instructions:

**General IO**

Load Register: ldr reg, value

First Word

| OPCODE | REGISTER |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| 40 | 0-31 | 0 | 0 | 0 | 0 | 0 | 0 |

Second Word

| VALUE |
| --- |

Load Address: lda bank, reg, address

First Word

| OPCODE | BANK | INPUT REG |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| 12 | 0-3 | 0-31 | 0 | 0 | 0 | 0 | 0 |

Second Word

| ADDRESS |
| --- |

Load Address Register: ldar bank, reg, reg(address)

| OPCODE | BANK | INPUT REG | ADDRESS REG |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| 19 | 0-3 | 0-31 | 0-31 | 0 | 0 | 0 | 0 |

Store Address: sta bank, reg, address

First Word

| OPCODE | BANK | REGISTER |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| 13 | 0-3 | 0-31 | 0 | 0 | 0 | 0 | 0 |

Second Word

| OUTPUT ADDRESS |
| --- |

Store Address Register: star bank, reg, reg(address)

| OPCODE | BANK | REGISTER | OUTPUT ADDRESS (reg) |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| 20 | 0-3 | 0-31 | 0-31 | 0 | 0 | 0 | 0 |

Copy Register: cpr reg1, reg2

| OPCODE | REG | REG |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| 23 | 0-31 | 0-31 | 0 | 0 | 0 | 0 | 0 |

Swap Register: swapr reg1, reg2

| OPCODE | REG | REG |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| 27 | 0-31 | 0-31 | 0 | 0 | 0 | 0 | 0 |

Convert Register from int to float: fconv reg

| OPCODE | REG |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| 28 | 0-31 | 0 | 0 | 0 | 0 | 0 | 0 |

Convert Register from float to int: iconv reg

| OPCODE | REG |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| 29 | 0-31 | 0 | 0 | 0 | 0 | 0 | 0 |

**Stack Manipulation**

Push Register: pushr reg

| OPCODE | REGISTER |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| 14 | 0-31 | 0 | 0 | 0 | 0 | 0 | 0 |

Pull Register: pullr reg

| OPCODE | INPUT REGISTER |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| 15 | 0-31 | 0 | 0 | 0 | 0 | 0 | 0 |

Pop Stack: pop

| OPCODE |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| 16 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

**Math Operations**

Add Registers (int) : iaddr reg1, reg2

| OPCODE | RESULT REGISTER | ADDEND  REGISTER |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| 17 | 0-31 | 0-31 | 0 | 0 | 0 | 0 | 0 |

Subtract Registers (int): isubr reg1, reg2

| OPCODE | Minuend REGISTER | Subtrahend  REGISTER |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| 18 | 0-31 | 0-31 | 0 | 0 | 0 | 0 | 0 |

Multiply Registers (int): imulr reg1, reg2

| OPCODE | RESULT REGISTER | FACTOR  REGISTER |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| 57 | 0-31 | 0-31 | 0 | 0 | 0 | 0 | 0 |

Divide Registers (int): idivr reg1, reg2

| OPCODE | Dividend REGISTER | Divisor  REGISTER |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| 21 | 0-31 | 0-31 | 0 | 0 | 0 | 0 | 0 |

Add Registers (float) : faddr reg1, reg2

| OPCODE | RESULT REGISTER | ADDEND  REGISTER |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| 22 | 0-31 | 0-31 | 0 | 0 | 0 | 0 | 0 |

Subtract Registers (float): fsubr reg1, reg2

| OPCODE | Minuend REGISTER | Subtrahend  REGISTER |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| 24 | 0-31 | 0-31 | 0 | 0 | 0 | 0 | 0 |

Multiply Registers (float): fmulr reg1, reg2

| OPCODE | RESULT REGISTER | FACTOR  REGISTER |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| 25 | 0-31 | 0-31 | 0 | 0 | 0 | 0 | 0 |

Divide Registers (float): fdivr reg1, reg2

| OPCODE | Dividend REGISTER | Divisor  REGISTER |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| 26 | 0-31 | 0-31 | 0 | 0 | 0 | 0 | 0 |

**Vector Math**

Vector Add (int): ivaddr reg1(v1.x), reg2(v1.y), reg3(v1.z), reg4(v2.x), reg5(v2.y), reg6(v2.z)

HLR: v1 = v1 + v2

| OPCODE | REGISTER 1 | REGISTER 2 | REGISTER 3 | REGISTER 4 | REGISTER 5 | REGISTER 6 |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| 30 | 0-31 | 0-31 | 0-31 | 0-31 | 0-31 | 0-31 | 0 |

Vector Subract (int): ivsubr reg1(v1.x), reg2(v1.y), reg3(v1.z), reg4(v2.x), reg5(v2.y), reg6(v2.z)

HLR: v1 = v1 - v2

| OPCODE | REGISTER 1 | REGISTER 2 | REGISTER 3 | REGISTER 4 | REGISTER 5 | REGISTER 6 |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| 31 | 0-31 | 0-31 | 0-31 | 0-31 | 0-31 | 0-31 | 0 |

Vector Multiply (int): ivmulr reg1(v1.x), reg2(v1.y), reg3(v1.z), reg4(v2.x), reg5(v2.y), reg6(v2.z)

HLR: v1 = v1 \* v2

| OPCODE | REGISTER 1 | REGISTER 2 | REGISTER 3 | REGISTER 4 | REGISTER 5 | REGISTER 6 |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| 32 | 0-31 | 0-31 | 0-31 | 0-31 | 0-31 | 0-31 | 0 |

Vector Divide (int): ivdivr reg1(v1.x), reg2(v1.y), reg3(v1.z), reg4(v2.x), reg5(v2.y), reg6(v2.z)

HLR: v1 = (v1 / v2)

| OPCODE | REGISTER 1 | REGISTER 2 | REGISTER 3 | REGISTER 4 | REGISTER 5 | REGISTER 6 |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| 33 | 0-31 | 0-31 | 0-31 | 0-31 | 0-31 | 0-31 | 0 |

Vector Add (float): fvaddr reg1(v1.x), reg2(v1.y), reg3(v1.z), reg4(v2.x), reg5(v2.y), reg6(v2.z)

HLR: v1 = v1 + v2

| OPCODE | REGISTER 1 | REGISTER 2 | REGISTER 3 | REGISTER 4 | REGISTER 5 | REGISTER 6 |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| 34 | 0-31 | 0-31 | 0-31 | 0-31 | 0-31 | 0-31 | 0 |

Vector Subract (float): fvsubr reg1(v1.x), reg2(v1.y), reg3(v1.z), reg4(v2.x), reg5(v2.y), reg6(v2.z)

HLR: v1 = v1 - v2

| OPCODE | REGISTER 1 | REGISTER 2 | REGISTER 3 | REGISTER 4 | REGISTER 5 | REGISTER 6 |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| 35 | 0-31 | 0-31 | 0-31 | 0-31 | 0-31 | 0-31 | 0 |

Vector Multiply (float): fvmulr reg1(v1.x), reg2(v1.y), reg3(v1.z), reg4(v2.x), reg5(v2.y), reg6(v2.z)

HLR: v1 = v1 \* v2

| OPCODE | REGISTER 1 | REGISTER 2 | REGISTER 3 | REGISTER 4 | REGISTER 5 | REGISTER 6 |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| 36 | 0-31 | 0-31 | 0-31 | 0-31 | 0-31 | 0-31 | 0 |

Vector Divide (float): fvdivr reg1(v1.x), reg2(v1.y), reg3(v1.z), reg4(v2.x), reg5(v2.y), reg6(v2.z)

HLR: v1 = (v1 / v2)

| OPCODE | REGISTER 1 | REGISTER 2 | REGISTER 3 | REGISTER 4 | REGISTER 5 | REGISTER 6 |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| 37 | 0-31 | 0-31 | 0-31 | 0-31 | 0-31 | 0-31 | 0 |

**Branching**

Jump Address: jmp address

First Word

| OPCODE |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| 38 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

Second Word

| ADDRESS |
| --- |

Jump Register: jmpr reg(address)

| OPCODE | REGISTER(Address) |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| 39 | 0-31 | 0 | 0 | 0 | 0 | 0 | 0 |

Return from Subroutine: ret

| OPCODE |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| 41 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

Call Function: call address

First Word

| OPCODE |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| 42 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

Second Word

| ADDRESS |
| --- |

Compare Registers: cmpr reg1, reg2

| OPCODE | REGISTER 1 | REGISTER 2 |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| 43 | 0-31 | 0-31 | 0 | 0 | 0 | 0 | 0 |

Branch if equal: bie address

First Word

| OPCODE | 0 | 0 |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| 44 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

Second Word

| ADDRESS |
| --- |

Branch if not equal: bin address

First Word

| OPCODE | 0 | 0 |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| 45 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

Second Word

| ADDRESS |
| --- |

Branch if greater: big address

First Word

| OPCODE | 0 | 0 |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| 46 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

Second Word

| ADDRESS |
| --- |

Branch if lesser: bil address

First Word

| OPCODE | 0 | 0 |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| 47 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

Second Word

| ADDRESS |
| --- |

Branch if greater than or equal: bige address

First Word

| OPCODE | 0 | 0 |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| 48 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

Second Word

| ADDRESS |
| --- |

Branch if lesser or equal: bile address

First Word

| OPCODE | 0 | 0 |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| 49 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

Second Word

| ADDRESS |
| --- |

**Bitwise**

AND Registers: andr reg1, reg2, reg3

R1 = R2 & R3

| OPCODE | REGISTER 1 | REGISTER 2 | REGISTER 3 |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| 50 | 0-31 | 0-31 | 0-31 | 0 | 0 | 0 | 0 |

OR Registers: orr reg1, reg2, reg3

R1 = R2 | R3

| OPCODE | REGISTER 1 | REGISTER 2 | REGISTER 3 |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| 51 | 0-31 | 0-31 | 0-31 | 0 | 0 | 0 | 0 |

XOR Registers: xorr reg1, reg2, reg3

R1 = R2 ^ R3

| OPCODE | REGISTER 1 | REGISTER 2 | REGISTER 3 |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| 52 | 0-31 | 0-31 | 0-31 | 0 | 0 | 0 | 0 |

NOT Registers: notr reg1, reg2

R1 = ~R2

| OPCODE | REGISTER 1 | REGISTER 2 |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| 53 | 0-31 | 0-31 | 0 | 0 | 0 | 0 | 0 |

Shift Left: shlr reg1, reg2, reg3

R1 = R2 << R3

| OPCODE | DEST  (reg) | SOURCE  (reg) | AMOUNT (reg) |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| 54 | 0-31 | 0-31 | 0-31 | 0 | 0 | 0 | 0 |

Shift Left: shrr reg1, reg2, reg3

R1 = R2 >> R3

| OPCODE | DEST  (reg) | SOURCE  (reg) | AMOUNT (reg) |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| 55 | 0-31 | 0-31 | 0-31 | 0 | 0 | 0 | 0 |

**Shift Right Arithmetic: sarr** reg1, reg2, reg3

**R1 = (int64\_t)R2 >> R3**

| OPCODE | DEST  (reg) | SOURCE  (reg) | AMOUNT (reg) |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| 56 | 0-31 | 0-31 | 0-31 | 0 | 0 | 0 | 0 |