

**0x**

**A 32-Bit VM written in Rust powered by a custom instruction set**

0xffset

# Contents

<b>1</b>	<b>Specs</b>	<b>3</b>
<b>2</b>	<b>Glossary</b>	<b>4</b>
2.1	Specialized registers . . . . .	4
2.2	Operands . . . . .	4
2.3	Opcodes . . . . .	5
<b>3</b>	<b>Status register</b>	<b>6</b>
<b>4</b>	<b>Instructions</b>	<b>7</b>
4.1	HALT - Halt . . . . .	7
4.2	NOP - No operation . . . . .	8
4.3	MOVR - Move to register . . . . .	9
4.4	MOVM - Move to memory . . . . .	10
4.5	MOVRR - Move register to register . . . . .	11
4.6	MOVRRM - Move register to memory . . . . .	12
4.7	MOVRRM - Move memory to register . . . . .	13
4.8	MOVRRPR - Move register pointer to register . . . . .	14
4.9	MOVRROR - Move register pointer + offset to register . . . . .	15
4.10	LOAD - Load buffer . . . . .	16
4.11	LOADR - Load buffer . . . . .	17
4.12	LOADM - Load buffer . . . . .	18
4.13	STORE - Store buffer . . . . .	19
4.14	STORER - Store buffer . . . . .	20
4.15	STOREM - Store buffer . . . . .	21
4.16	POP - Pop . . . . .	22
4.17	PUSH - Push . . . . .	23
4.18	PUSHR - Push register . . . . .	24

# 1 Specs

- 32-bit architecture
- 8 32-bit general purpose registers
- Variable sized memory
- Variable sized display
- Variable sized hard drive

## 2 Glossary

### 2.1 Specialized registers

- **PC** (32-Bit): Program Counter
- **SP** (32-Bit): Stack pointer
- **FP** (32-Bit): Frame pointer
- **ACC** (32-Bit): Accumulator
- **SR** (32-Bit): Status register

### 2.2 Operands

- **S**: Stack
- **R** (32-Bit): Register
- **Ro** (32-Bit): Origin register
- **Rd** (32-Bit): Destination register
- **R0** (32-Bit): Lowest general purpose register
- **Rx** (32-Bit): Highest general purpose register
- **Rs** (32-Bit): Status register
- **Sm**: Bitmaskt for status register
- **Sx**: Highest bit of status register
- **M** (32-Bit): Memory address
- **M0** (32-Bit): Lowest memory address
- **Mx** (32-Bit): Highest memory address
- **Mo** (32-Bit): Origin memory address
- **Md** (32-Bit): Destination memory address
- **k** (32-Bit): Constant memory address
- **K** (32-Bit): Constant

## 2.3 Opcodes

<i>Instruction</i>	<i>Parameter 1</i>	<i>Parameter 2</i>	<i>Parameter n</i>
xxxx xxxx	aaaa aaaa	bbbb bbbb	nnnn nnnn

### 3 Status register

						<b>O</b>	<b>Z</b>
--	--	--	--	--	--	----------	----------

#### **Z - Zero flag:**

- If the result of an operation is zero, the zero flag is set.

#### **O - Overflow flag:**

- If the result of an operation is too large to fit in 32-Bit, the overflow flag is set.

## 4 Instructions

### 4.1 HALT - Halt

**Description:**

Halts the program.

**Operation:**

None

**Syntax**

HALT

**Operands**

None

**Program counter**

None

**Opcode:**

1111 1111			
-----------	--	--	--

**Status register:**

						-	-
--	--	--	--	--	--	---	---

## 4.2 NOP - No operation

### Description:

Does nothing.

### Operation:

None

### Syntax

NOP

### Operands

None

### Program counter

PC + 1 → PC

### Opcode:

0000 0000			
-----------	--	--	--

### Status register:

						-	-
--	--	--	--	--	--	---	---



### 4.3 MOVR - Move to register

**Description:**

Moves value  $K$  into register  $Rd$ .

**Operation:**

$K \rightarrow Rd$

**Syntax**

MOVR  $K$ ,  $Rd$

**Operands**

$0 \leq K \leq 2^{32} - 1$

$R0 \leq Rd \leq Rx$

**Program counter**

$PC + 1 \rightarrow PC$

**Opcode:**

0001 0000	KKKK KKKK	dddd dddd	
-----------	-----------	-----------	--

**Status register:**

						-	-
--	--	--	--	--	--	---	---

## 4.4 MOVM - Move to memory

### Description:

Moves value  $K$  into memory location  $k$ .

### Operation:

$K \rightarrow k$

### Syntax

MOVM  $K$ ,  $k$

### Operands

$0 \leq K \leq 2^{32} - 1$

$M0 \leq k \leq Mx$

### Program counter

$PC + 1 \rightarrow PC$

### Opcode:

0001 0001	KKKK KKKK	kkkk kkkk	
-----------	-----------	-----------	--

### Status register:

						-	-
--	--	--	--	--	--	---	---

## 4.5 MOVRR - Move register to register

### Description:

Moves value from register  $R_o$  into register  $R_d$ .

### Operation:

$R_o \rightarrow R_d$

### Syntax

MOVRR  $R_o$ ,  $R_d$

### Operands

$R0 \leq R_o, R_d \leq Rx$

### Program counter

$PC + 1 \rightarrow PC$

### Opcode:

0001 0010	0000 0000	dddd dddd	
-----------	-----------	-----------	--

### Status register:

						-	-
--	--	--	--	--	--	---	---

## 4.6 MOVRM - Move register to memory

### Description:

Moves value from a register  $R_0$  into memory location  $k$ .

### Operation:

$R_0 \rightarrow k$

### Syntax

MOVRM  $R_0$ ,  $k$

### Operands

$M_0 \leq k \leq M_x$

$R_0 \leq R_o \leq R_x$

### Program counter

$PC + 1 \rightarrow PC$

### Opcode:

0001 0011	0000 0000	kkkk kkkk	
-----------	-----------	-----------	--

### Status register:

						-	-
--	--	--	--	--	--	---	---

## 4.7 MOVMR - Move memory to register

### Description:

Moves value from memory location  $k$  into register  $Rd$ .

### Operation:

$k \rightarrow Rd$

### Syntax

MOVMR  $k, Rd$

### Operands

$M0 \leq k \leq Mx$

$R0 \leq Rd \leq Rx$

### Program counter

$PC + 1 \rightarrow PC$

### Opcode:

0001 0100	kkkk kkkk	dddd dddd	
-----------	-----------	-----------	--

### Status register:

						-	-
--	--	--	--	--	--	---	---

## 4.8 MOVRPR - Move register pointer to register

### Description:

Moves a value from memory location  $R_0^*$  into register  $R_d$ .

### Operation:

$R_0^* \rightarrow R_d$

### Syntax

MOVRPR  $R_0, R_d$

### Operands

$R_0 \leq R_0, R_d \leq R_x$

### Program counter

$PC + 1 \rightarrow PC$

### Opcode:

0001 0111	0000 0000	dddd dddd	
-----------	-----------	-----------	--

### Status register:

						-	-
--	--	--	--	--	--	---	---

## 4.9 MOVROR - Move register pointer + offset to register

### Description:

Moves a value from memory location  $R_0^* + K$  into register  $R_d$ .

### Operation:

$R_0^* + K \rightarrow R_d$

### Syntax

MOVROR  $R_0$ ,  $K$ ,  $R_d$

### Operands

$0 \leq K \leq 2^{32} - 1$   
 $R_0 \leq R_0, R_d \leq R_x$

### Program counter

$PC + 1 \rightarrow PC$

### Opcode:

0001 1000	0000 0000	KKKK KKKK	dddd dddd
-----------	-----------	-----------	-----------

### Status register:

						-	-
--	--	--	--	--	--	---	---

## 4.10 LOAD - Load buffer

### Description:

Copys a byte buffer from device at  $R_o^*$  to memory range  $k$  to  $k + R$ .

### Operation:

$R_o^* \rightarrow k$  to  $k + R$

### Syntax

LOAD  $R_o, R, k$

### Operands

$M0 \leq k \leq Mx$

$R0 \leq R_o, R \leq Rx$

### Program counter

$PC + 1 \rightarrow PC$

### Opcode:

0001 1001	0000 0000	RRRR RRRR	kkkk kkkk
-----------	-----------	-----------	-----------

### Status register:

						-	-
--	--	--	--	--	--	---	---



## 4.11 LOADR - Load buffer

### Description:

Copys a byte buffer from device at  $R_o^*$  to memory range  $R_d^*$  to  $R_d^* + R$ .

### Operation:

$R_o^* \rightarrow R_d^*$  to  $R_d^* + R$

### Syntax

LOADR  $R_o$ ,  $R$ ,  $R_d$

### Operands

$R0 \leq R_o, R, R_d \leq Rx$

### Program counter

$PC + 1 \rightarrow PC$

### Opcode:

0001 1010	0000 0000	RRRR RRRR	dddd dddd
-----------	-----------	-----------	-----------

### Status register:

						-	-
--	--	--	--	--	--	---	---

## 4.12 LOADM - Load buffer

### Description:

Copys a byte buffer from device at  $Ro^*$  to memory range  $Md^*$  to  $Md^* + R$ .

### Operation:

$Ro^* \rightarrow Md^*$  to  $Md^* + R$

### Syntax

LOADM  $Ro$ ,  $R$ ,  $Md$

### Operands

$M0 \leq Md \leq Mx$

$R0 \leq Ro, R \leq Rx$

### Program counter

$PC + 1 \rightarrow PC$

### Opcode:

0001 1011	0000 0000	RRRR RRRR	dddd dddd
-----------	-----------	-----------	-----------

### Status register:

						-	-
--	--	--	--	--	--	---	---

### 4.13 STORE - Store buffer

**Description:**

Copys a byte buffer from memory range  $k$  to  $k + R$  to device at  $Rd^*$ .

**Operation:**

$k$  to  $k + R \rightarrow Rd^*$

**Syntax**

STORE  $k, R, Rd$

**Operands**

$M0 \leq k \leq Mx$

$R0 \leq Ro, R \leq Rx$

**Program counter**

$PC + 1 \rightarrow PC$

**Opcode:**

0001 1100	kkkk kkkk	RRRR RRRR	dddd dddd
-----------	-----------	-----------	-----------

**Status register:**

						-	-
--	--	--	--	--	--	---	---

#### 4.14 STORER - Store buffer

**Description:**

Copys a byte buffer from memory range  $Ro^*$  to  $Ro^* + R$  to device at  $Rd^*$ .

**Operation:**

$Ro^*$  to  $Ro^* + R \rightarrow Rd^*$

**Syntax**

STORER  $Ro, R, Rd$

**Operands**

$R0 \leq Ro, R, Rd \leq Rx$

**Program counter**

$PC + 1 \rightarrow PC$

**Opcode:**

0001 1101	0000 0000	RRRR RRRR	dddd dddd
-----------	-----------	-----------	-----------

**Status register:**

						-	-
--	--	--	--	--	--	---	---

## 4.15 STOREM - Store buffer

### Description:

Copys a byte buffer from memory range  $Mo^*$  to  $Mo^* + R$  to device at  $Rd^*$ .

### Operation:

$Mo^*$  to  $Mo^* + R \rightarrow Rd^*$

### Syntax

STOREM  $Mo$ ,  $R$ ,  $Rd$

### Operands

$M0 \leq k \leq Mx$

$R0 \leq R, Rd \leq Rx$

### Program counter

$PC + 1 \rightarrow PC$

### Opcode:

0001 1110	0000 0000	RRRR RRRR	dddd dddd
-----------	-----------	-----------	-----------

### Status register:

						-	-
--	--	--	--	--	--	---	---

## 4.16 POP - Pop

### Description:

Pops a value from the stack into register  $R_d$ .

### Operation:

$S \rightarrow R_d, SP - 4 \rightarrow SP$

### Syntax

POP  $R_d$

### Operands

$R_0 \leq R_d \leq R_x$

### Program counter

$PC + 1 \rightarrow PC$

### Opcode:

0000 0101	dddd dddd		
-----------	-----------	--	--

### Status register:

						-	-
--	--	--	--	--	--	---	---

## 4.17 PUSH - Push

### Description:

Pushes value  $K$  onto the stack.

### Operation:

$SP + 4 \rightarrow SP, K \rightarrow S$

### Syntax

PUSH  $K$

### Operands

$0 \leq K \leq 2^{32} - 1$

### Program counter

$PC + 1 \rightarrow PC$

### Opcode:

0001 0101	KKKK KKKK		
-----------	-----------	--	--

### Status register:

						-	-
--	--	--	--	--	--	---	---

## 4.18 PUSH - Push register

### Description:

Pushes value  $R_0$  onto the stack.

### Operation:

$SP + 4 \rightarrow SP, R_0 \rightarrow S$

### Syntax

PUSH  $R_0$

### Operands

$R_0 \leq R_0 \leq R_x$

### Program counter

$PC + 1 \rightarrow PC$

### Opcode:

0001 0110	0000 0000		
-----------	-----------	--	--

### Status register:

						-	-
--	--	--	--	--	--	---	---