0x

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

0xffset

Contents

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

Instruction	*Parameter 1*	*Parameter 2*	*Parameter n*
xxxx xxxx	aaaa aaaa	bbbb bbbb	nnnn nnnn

3 Status register

						O	**Z**

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	**Operands**	**Program counter**
`HALT`	None	None

Opcode:

1111 1111			

Status register:

						O	Z
						-	-

4.2 NOP - No operation

Description:

Does nothing.

Operation:

None

Syntax	Operands	Program counter
`NOP`	None	PC + 1 → PC

Opcode:

0000 0000			

Status register:

						O	Z
						-	-

4.3 MOVR - Move to register

Description:

Moves value `K` into register `Rd`.

Operation:

K → Rd

Syntax	Operands	Program counter
`MOVR K, Rd`	$0 \le K \le 2^{32} - 1$ $R0 \le Rd \le Rx$	PC + 1 → PC

Opcode:

0001 0000	KKKK KKKK	dddd dddd	

Status register:

						O	Z
						-	-

4.4 MOVM - Move to memory

Description:

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

Operation:

K → k

Syntax	Operands	Program counter
`MOVM K, k`	$0 \leq K \leq 2^{32} - 1$ $M0 \leq k \leq Mx$	PC + 1 → PC

Opcode:

0001 0001	KKKK KKKK	kkkk kkkk	

Status register:

						O	Z
						-	-

4.5 MOVRR - Move register to register

Description:

Moves value from register `Ro` into register `Rd`.

Operation:

Ro → Rd

Syntax	Operands	Program counter
`MOVRR Ro, Rd`	$R0 \leq Ro, Rd \leq Rx$	PC + 1 → PC

Opcode:

0001 0010	oooo oooo	dddd dddd	

Status register:

						O	Z
						-	-

4.6 MOVRM - Move register to memory

Description:

Moves value from a register `Ro` into memory location `k`.

Operation:

Ro → k

Syntax	Operands	Program counter
`MOVRM Ro, k`	$M0 \leq k \leq Mx$ $R0 \leq Ro \leq Rx$	PC + 1 → PC

Opcode:

0001 0011	oooo oooo	kkkk kkkk	

Status register:

						O	Z
						-	-

4.7 MOVMR - Move memory to register

Description:

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

Operation:

k → Rd

Syntax	Operands	Program counter
`MOVMR k, Rd`	$M0 \leq k \leq Mx$ $R0 \leq Rd \leq Rx$	PC + 1 → PC

Opcode:

0001 0100	kkkk kkkk	dddd dddd	

Status register:

						O	Z
						-	-

4.8 MOVRPR - Move register pointer to register

Description:

Moves a value from memory location `Ro*` into register `Rd`.

Operation:

Ro* → Rd

Syntax	**Operands**	**Program counter**
`MOVRPR Ro, Rd`	$R0 \leq Ro, Rd \leq Rx$	PC + 1 → PC

Opcode:

0001 0111	oooo oooo	dddd dddd	

Status register:

						O	**Z**
						-	-

4.9 MOVROR - Move register pointer + offset to register

Description:

Moves a value from memory location `Ro* + K` into register `Rd`.

Operation:

Ro* + K → Rd

Syntax	Operands	Program counter
`MOVROR Ro, K, Rd`	$0 \leq K \leq 2^{32} - 1$ $R0 \leq Ro, Rd \leq Rx$	PC + 1 → PC

Opcode:

0001 1000	oooo oooo	KKKK KKKK	dddd dddd

Status register:

						O	Z
						-	-

4.10 LOAD - Load buffer

Description:

Copys a byte buffer from device at `Ro*` to memory range `k to k + R`.

Operation:

Ro* → k to k + R

Syntax	Operands	Program counter
`LOAD Ro, R, k`	$M0 \leq k \leq Mx$ $R0 \leq Ro, R \leq Rx$	PC + 1 → PC

Opcode:

0001 1001	oooo oooo	RRRR RRRR	kkkk kkkk

Status register:

						O	Z
						-	-

4.11 LOADR - Load buffer

Description:

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

Operation:

Ro* → Rd* to Rd* + R

Syntax	**Operands**	**Program counter**
`LOADR Ro, R, Rd`	$R0 \leq Ro, R, Rd \leq Rx$	PC + 1 → PC

Opcode:

0001 1010	oooo oooo	RRRR RRRR	dddd dddd

Status register:

						O	**Z**
						-	-

4.12 LOADM - Load buffer

Description:

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

Operation:

Ro* → Md* to Md* + R

Syntax	**Operands**	**Program counter**
`LOADM Ro, R, Md`	$M0 \leq Md \leq Mx$ $R0 \leq Ro, R \leq Rx$	PC + 1 → PC

Opcode:

0001 1011	oooo oooo	RRRR RRRR	dddd dddd

Status register:

						O	Z
						-	-

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 → Rd*

Syntax	Operands	Program counter
`STORE k, R, Rd`	$M0 \leq k \leq Mx$ $R0 \leq Ro, R \leq Rx$	PC + 1 → PC

Opcode:

0001 1100	kkkk kkkk	RRRR RRRR	dddd dddd

Status register:

						O	Z
						-	-

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 → Rd*

Syntax	Operands	Program counter
`STORER Ro, R, Rd`	$R0 \leq Ro, R, Rd \leq Rx$	PC + 1 → PC

Opcode:

0001 1101	oooo oooo	RRRR RRRR	dddd dddd

Status register:

						O	Z
						-	-

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 → Rd*

Syntax	Operands	Program counter
`STOREM Mo, R, Rd`	$M0 \leq k \leq Mx$ $R0 \leq R, Rd \leq Rx$	PC + 1 → PC

Opcode:

0001 1110	oooo oooo	RRRR RRRR	dddd dddd

Status register:

						O	Z
						-	-

4.16 POP - Pop

Description:

Pops a value from the stack into register `Rd`.

Operation:

S → Rd, SP - 4 → SP

Syntax	Operands	Program counter
`POP Rd`	$R0 \leq Rd \leq Rx$	PC + 1 → PC

Opcode:

0000 0101	dddd dddd		

Status register:

						O	Z
						-	-

4.17 PUSH - Push

Description:

Pushes value `K` onto the stack.

Operation:

SP + 4 → SP, K → S

Syntax	Operands	Program counter
`PUSH K`	$0 \leq K \leq 2^{32} - 1$	PC + 1 → PC

Opcode:

0001 0101	KKKK KKKK		

Status register:

						O	Z
						-	-

4.18 PUSHR - Push register

Description:

Pushes value `Ro` onto the stack.

Operation:

SP + 4 → SP, Ro → S

Syntax	Operands	Program counter
`PUSH Ro`	$R0 \leq Ro \leq Rx$	PC + 1 → PC

Opcode:

0001 0110	oooo oooo		

Status register:

						O	Z
						-	-

4.19 ADD - Add

Description:

Adds value `K` and register `R` together and stores the result in `ACC`.

Operation:

K + R → ACC

Syntax	Operands	Program counter
`ADD K, R`	$0 \leq K \leq 2^{32} - 1$ $R0 \leq R \leq Rx$	PC + 1 → PC

Opcode:

0010 0000	KKKK KKKK	RRRR RRRR	

Status register:

						O	Z
						x	x

Z - Set if the operation results in 0
O - Set if the operation overflows

4.20 ADDR - Add register

Description:

Adds register R_1 and register R_2 together and stores the result in ACC.

Operation:

$R_1 + R_2 \rightarrow ACC$

Syntax	Operands	Program counter
ADDR R_1, R_2	$R0 \leq R_1, R_2 \leq Rx$	PC + 1 → PC

Opcode:

0010 0001	R_1R_1 R_1R_1	R_2R_2 R_2R_2	

Status register:

						O	Z
						x	x

Z - Set if the operation results in 0
O - Set if the operation overflows

4.21 SUB - Subtract

Description:

Subtracts value `K` from register `R` and stores the result in `ACC`.

Operation:

R - K → ACC

Syntax	Operands	Program counter
`SUB R, K`	$0 \leq K \leq 2^{32} - 1$ $R0 \leq R \leq Rx$	PC + 1 → PC

Opcode:

0010 0010	RRRR RRRR	KKKK KKKK	

Status register:

						O	Z
						x	x

Z - Set if the operation results in 0
O - Set if the operation overflows

4.22 SUBWR - Subtract register from word

Description:

Subtracts register `R` from value `K` and stores the result in `ACC`.

Operation:

K - R → ACC

Syntax	Operands	Program counter
`SUBWR K, R`	$0 \leq K \leq 2^{32} - 1$ $R0 \leq R \leq Rx$	PC + 1 → PC

Opcode:

0010 0010	KKKK KKKK	RRRR RRRR	

Status register:

						O	Z
						x	x

Z - Set if the operation results in 0
O - Set if the operation overflows

4.23 SUBR - Subtract register

Description:

Subtracts register R_2 from register R_1 and stores the result in ACC.

Operation:

R_1 - R_2 → ACC

Syntax	Operands	Program counter
SUBR R_1, R_2	$0 \leq K \leq 2^{32} - 1$ $R0 \leq R_1, R_2 \leq Rx$	PC + 1 → PC

Opcode:

0010 0011	R_1R_1 R_1R_1	R_2R_2 R_2R_2	

Status register:

						O	Z
						x	x

Z - Set if the operation results in 0
O - Set if the operation overflows

4.24 MULT - Multiply

Description:

Multiplies value `K` and register `R` together and stores the result in `ACC`.

Operation:

K × R → ACC

Syntax	Operands	Program counter
`MULT K, R`	$0 \leq K \leq 2^{32} - 1$ $R0 \leq R \leq Rx$	PC + 1 → PC

Opcode:

0010 0101	KKKK KKKK	RRRR RRRR	

Status register:

						O	Z
						x	x

Z - Set if the operation results in 0
O - Set if the operation overflows

4.25 MULTR - Multiply register

Description:

Multiplies register R_1 and register R_2 together and stores the result in ACC.

Operation:

$R_1 \times R_2 \rightarrow ACC$

Syntax	Operands	Program counter
MULTR R_1, R_2	$R0 \leq R_1, R_2 \leq Rx$	PC + 1 → PC

Opcode:

0010 0110	R_1R_1 R_1R_1	R_2R_2 R_2R_2	

Status register:

						O	Z
						x	x

Z - Set if the operation results in 0
O - Set if the operation overflows

4.26 DIV - Divide

Description:

Devides register `R` by value `K` and stores the result in `ACC`.

Operation:

R ÷ K → ACC

Syntax	Operands	Program counter
`DIV R, K`	$0 \leq K \leq 2^{32} - 1$ $R0 \leq R \leq Rx$	PC + 1 → PC

Opcode:

0010 0111	RRRR RRRR	KKKK KKKK	

Status register:

						O	Z
						x	x

Z - Set if the operation results in 0
O - Set if the operation overflows

4.27 DIVWR - Divide word by register

Description:

Devides value `K` by register `R` and stores the result in `ACC`.

Operation:

K $\div$ R $\rightarrow$ ACC

Syntax	Operands	Program counter
`DIVWR K, R`	$0 \leq K \leq 2^{32} - 1$ $R0 \leq R \leq Rx$	PC + 1 $\rightarrow$ PC

Opcode:

0010 1000	KKKK KKKK	RRRR RRRR	

Status register:

						O	Z
						x	x

Z - Set if the operation results in 0
O - Set if the operation overflows

4.28 DIVR - Divide registers

Description:

Divides register R_1 by register R_2 and stores the result in ACC.

Operation:

$R_1 \div R_2 \rightarrow$ ACC

Syntax	Operands	Program counter
DIVR R_1, R_2	$0 \leq K \leq 2^{32}-1$ $R0 \leq R_1, R_2 \leq Rx$	PC + 1 → PC

Opcode:

0010 1001	R_1R_1 R_1R_1	R_2R_2 R_2R_2	

Status register:

						O	Z
						x	x

Z - Set if the operation results in 0
O - Set if the operation overflows