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:

						-	-

4.2 NOP - No operation

Description:

Does nothing.

Operation:

None

Syntax	Operands	Program counter
`NOP`	None	PC + 1 $\rightarrow$ PC

Opcode:

0000 0000			

Status register:

						-	-

4.3 MOVR - Move to register

Description:

Moves value `K` into register `Rd`.

Operation:

K $\rightarrow$ Rd

Syntax	**Operands**	**Program counter**
`MOVR K, Rd`	$0 \leq K \leq 2^{32} - 1$ $R0 \leq Rd \leq Rx$	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	**Operands**	**Program counter**
`MOVM K, k`	$0 \leq K \leq 2^{32} - 1$ $M0 \leq k \leq Mx$	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 `Ro` into register `Rd`.

Operation:

Ro $\rightarrow$ Rd

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

Opcode:

0001 0010	oooo oooo	dddd dddd	

Status register:

						-	-

4.6 MOVRM - Move register to memory

Description:

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

Operation:

Ro $\rightarrow$ k

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

Opcode:

0001 0011	oooo oooo	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	Operands	Program counter
`MOVMR k, Rd`	$M0 \leq k \leq Mx$ $R0 \leq Rd \leq Rx$	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 `Ro*` into register `Rd`.

Operation:

Ro* $\rightarrow$ Rd

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

Opcode:

0001 0111	oooo oooo	dddd dddd	

Status register:

						-	-

4.9 MOVROR - Move register pointer + offset to register

Description:

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

Operation:

Ro* + K $\rightarrow$ Rd

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

Opcode:

0001 1000	oooo oooo	KKKK KKKK	dddd dddd

Status register:

						-	-

4.10 LOAD - Load buffer

Description:

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

Operation:

Ro* $\rightarrow$ 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 $\rightarrow$ PC

Opcode:

0001 1001	oooo oooo	RRRR RRRR	kkkk kkkk

Status register:

						-	-

4.11 LOADR - Load buffer

Description:

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

Operation:

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

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

Opcode:

0001 1010	oooo oooo	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	Operands	Program counter
`LOADM Ro, R, Md`	$M0 \leq Md \leq Mx$ $R0 \leq Ro, R \leq Rx$	PC + 1 $\rightarrow$ PC

Opcode:

0001 1011	oooo oooo	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	Operands	Program counter
`STORE k, R, Rd`	$M0 \leq k \leq Mx$ $R0 \leq Ro, R \leq Rx$	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	**Operands**	**Program counter**
`STORER Ro, R, Rd`	$R0 \leq Ro, R, Rd \leq Rx$	PC + 1 $\rightarrow$ PC

Opcode:

0001 1101	oooo oooo	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	Operands	Program counter
`STOREM Mo, R, Rd`	$M0 \leq k \leq Mx$ $R0 \leq R, Rd \leq Rx$	PC + 1 $\rightarrow$ PC

Opcode:

0001 1110	oooo oooo	RRRR RRRR	dddd dddd

Status register:

						-	-

4.16 POP - Pop

Description:

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

Operation:

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

Syntax	**Operands**	**Program counter**
`POP Rd`	$R0 \leq Rd \leq Rx$	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	Operands	Program counter
`PUSH K`	$0 \leq K \leq 2^{32} - 1$	PC + 1 $\rightarrow$ PC

Opcode:

0001 0101	KKKK KKKK		

Status register:

						-	-

4.18 PUSHR - Push register

Description:

Pushes value `Ro` onto the stack.

Operation:

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

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

Opcode:

0001 0110	oooo oooo		

Status register:

						-	-