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
- **Sb** (8-Bit): Bit in status register
- **S0**: Lowest bit in 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`	$0 \leq K \leq 2^{32} - 1$ $R0 \leq Rd \leq Rx$	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`	$0 \leq K \leq 2^{32} - 1$	PC + 1 → PC
	$M0 \leq k \leq Mx$	

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 ÷ R → ACC

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

4.29 INC - Increment

Description:

Increments register `Rd` by one.

Operation:

Rd + 1 → Rd

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

Opcode:

0010 1010	dddd dddd		

Status register:

						O	Z
						x	x

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

4.30 DEC - Decrement

Description:

Decrements register `Rd` by one.

Operation:

Rd - 1 → Rd

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

Opcode:

0010 1011	dddd dddd		

Status register:

						O	Z
						x	x

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

4.31 LSF - Left shift

Description:

Shifts register `Rd` left by `K` bits.

Operation:

Rd $\ll$ K $\rightarrow$ Rd

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

Opcode:

0101 0000	dddd dddd	KKKK KKKK	

Status register:

						O	Z
						-	x

Z - Set if the operation results in 0

4.32 LSFR - Left shift by register

Description:

Shifts register `Rd` left by `R` bits.

Operation:

Rd $\ll$ R → Rd

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

Opcode:

0101 0001	dddd dddd	RRRR RRRR	

Status register:

						O	Z
						-	x

Z - Set if the operation results in 0

4.33 RSF - Right shift

Description:

Shifts register `Rd` right by `K` bits.

Operation:

Rd $\gg$ K → Rd

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

Opcode:

0101 0010	dddd dddd	KKKK KKKK	

Status register:

						O	Z
						-	x

Z - Set if the operation results in 0

4.34 RSFR - Right shift by register

Description:

Shifts register `Rd` right by `R` bits.

Operation:

Rd $\gg$ R → Rd

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

Opcode:

0101 0011	dddd dddd	RRRR RRRR	

Status register:

						O	Z
						-	x

Z - Set if the operation results in 0

4.35 WLSF - Wrapping left shift

Description:

Shifts register `Rd` left by `K` bits and wraps the bits around.

Operation:

Rd $\ll$ K → Rd

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

Opcode:

0101 0100	dddd dddd	KKKK KKKK	

Status register:

						O	**Z**
						-	x

Z - Set if the operation results in 0

4.36 WLSFR - Wrapping left shift by register

Description:

Shifts register `Rd` left by `R` bits and wraps the bits around.

Operation:

Rd $\ll$ R → Rd

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

Opcode:

0101 0101	dddd dddd	RRRR RRRR	

Status register:

						O	Z
						-	x

Z - Set if the operation results in 0

4.37 WRSF - Wrapping right shift

Description:

Shifts register `Rd` right by `K` bits and wraps the bits around.

Operation:

Rd $\gg$ K → Rd

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

Opcode:

0101 0110	dddd dddd	KKKK KKKK	

Status register:

						O	Z
						-	x

Z - Set if the operation results in 0

4.38 WRSFR - wrapping right shift by register

Description:

Shifts register `Rd` right by `R` bits and wraps the bits around.

Operation:

Rd $\gg$ R $\rightarrow$ Rd

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

Opcode:

0101 0111	dddd dddd	RRRR RRRR	

Status register:

						O	Z
						-	x

Z - Set if the operation results in 0

4.39 AND - Bitwise AND

Description:

Performs a bitwise AND operation on register `Rd` with value `K` and stores the result in `Rd`.

Operation:

Rd & K → Rd

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

Opcode:

0101 1000	dddd dddd	KKKK KKKK	

Status register:

						O	Z
						-	x

Z - Set if the operation results in 0

4.40 ANDR - Bitwise AND by register

Description:

Performs a bitwise AND operation on register `Rd` with register `R` and stores the result in `Rd`.

Operation:

Rd & R → Rd

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

Opcode:

0101 1001	dddd dddd	KKKK KKKK	

Status register:

						O	Z
						-	x

Z - Set if the operation results in 0

4.41 OR - Bitwise OR

Description:

Performs a bitwise OR operation on register `Rd` with value `K` and stores the result in `Rd`.

Operation:

Rd | K → Rd

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

Opcode:

0101 1010	dddd dddd	KKKK KKKK	

Status register:

						O	Z
						-	x

Z - Set if the operation results in 0

4.42 ORR - Bitwise OR by register

Description:

Performs a bitwise OR operation on register `Rd` with register `R` and stores the result in `Rd`.

Operation:

Rd | R → Rd

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

Opcode:

0101 1011	dddd dddd	KKKK KKKK	

Status register:

						O	Z
						-	x

Z - Set if the operation results in 0

4.43 XOR - Bitwise XOR

Description:

Performs a bitwise XOR operation on register `Rd` with value `K` and stores the result in `Rd`.

Operation:

Rd ^ K → Rd

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

Opcode:

0101 1100	dddd dddd	KKKK KKKK	

Status register:

						O	Z
						-	x

Z - Set if the operation results in 0

4.44 XORR - Bitwise XOR by register

Description:

Performs a bitwise XOR operation on register `Rd` with register `R` and stores the result in `Rd`.

Operation:

Rd ^ R → Rd

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

Opcode:

0101 1101	dddd dddd	KKKK KKKK	

Status register:

						O	Z
						-	x

Z - Set if the operation results in 0

4.45 NOT - Not

Description:

Flips the bits of register `Rd`.

Operation:

~Rd → Rd

Syntax	Operands	Program counter
`NOT Rd`	$R0 \leq Rd \leq Rx$	PC + 1 → PC

Opcode:

0101 1110	dddd dddd		

Status register:

						O	Z
						-	x

Z - Set if the operation results in 0

4.46 BRBS - Branch if bit set

Description:

If the `Sb` bit in the SR is set, branch to absolute address `k`.

Operation:

If SR(Sb) = 1 then k → PC else PC + 1 → PC

Syntax	Operands	Program counter
`BRBS Sb, k`	$M0 \leq k \leq Mx$	k → PC
	$S0 \leq Sb \leq Sx$	PC + 1 → PC

Opcode:

0011 0000	bbbb bbbb	kkkk kkkk	

Status register:

						O	Z
						-	-

4.47 BRBC - Branch if bit clear

Description:

If the `Sb` bit in the SR is clear, branch to absolute address `k`.

Operation:

If SR(Sb) = 0 then k → PC else PC + 1 → PC

Syntax	Operands	Program counter
`BRBC Sb, k`	$M0 \leq k \leq Mx$	k → PC
	$S0 \leq Sb \leq Sx$	PC + 1 → PC

Opcode:

0011 0001	bbbb bbbb	kkkk kkkk	

Status register:

						O	Z
						-	-

4.48 BREQ - Branch if equal

Description:

If `K` is equal to `ACC`, branch to absolute address `k`.

Operation:

If ACC = K then k → PC else PC + 1 → PC

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

Opcode:

0011 0010	KKKK KKKK	kkkk kkkk	

Status register:

						O	Z
						-	-

4.49 BREQR - Branch if equal register

Description:

If `R` is equal to `ACC`, branch to absolute address `k`.

Operation:

If ACC = R then k → PC else PC + 1 → PC

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

Opcode:

0011 0011	RRRR RRRR	kkkk kkkk	

Status register:

						O	Z
						-	-

4.50 BREQRW - Branch if equal register and word

Description:

If `K` is equal to `R`, branch to absolute address `k`.

Operation:

If R = K then k → PC else PC + 1 → PC

Syntax	Operands	Program counter
`BREQRW R, K, k`	$0 \leq K \leq 2^{32} - 1$	k → PC
	$M0 \leq k \leq Mx$	PC + 1 → PC
	$R0 \leq R \leq Rx$	

Opcode:

0011 0100	RRRR RRRR	KKKK KKKK	kkkk kkkk

Status register:

						O	Z
						-	-

4.51 BREQRR - Branch if equal registers

Description:

If $\mathtt{R}_1$ is equal to $\mathtt{R}_2$, branch to absolute address `k`.

Operation:

If $R_1 = R_2$ then k → PC else PC + 1 → PC

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

Opcode:

0011 0101	$R_1R_1\ R_1R_1$	$R_2R_2\ R_2R_2$	kkkk kkkk

Status register:

						O	Z
						-	-

4.52 BRNQ - Branch if not equal

Description:

If `K` is not equal to `ACC`, branch to absolute address `k`.

Operation:

If ACC $\neq$ K then k → PC else PC + 1 → PC

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

Opcode:

0011 0110	KKKK KKKK	kkkk kkkk	

Status register:

						O	Z
						-	-

4.53 BRNQR - Branch if not equal register

Description:

If `R` is not equal to `ACC`, branch to absolute address `k`.

Operation:

If ACC $\neq$ R then k → PC else PC + 1 → PC

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

Opcode:

0011 0111	RRRR RRRR	kkkk kkkk	

Status register:

						O	Z
						-	-

4.54 BRNQRW - Branch if not equal register and word

Description:

If `K` is not equal to `R`, branch to absolute address `k`.

Operation:

If R $\neq$ K then k → PC else PC + 1 → PC

Syntax	Operands	Program counter
`BRNQRW R, K, k`	$0 \leq K \leq 2^{32} - 1$ $M0 \leq k \leq Mx$ $R0 \leq R \leq Rx$	k → PC PC + 1 → PC

Opcode:

0011 1000	RRRR RRRR	KKKK KKKK	kkkk kkkk

Status register:

						O	Z
						-	-

4.55 BREQRR - Branch if not equal registers

Description:

If $\mathtt{R}_1$ is not equal to $\mathtt{R}_2$, branch to absolute address $\mathtt{k}$.

Operation:

If $R_1 \neq R_2$ then k → PC else PC + 1 → PC

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

Opcode:

0011 1001	$R_1R_1\ R_1R_1$	$R_2R_2\ R_2R_2$	kkkk kkkk

Status register:

						O	Z
						-	-

4.56 BRLT - Branch if less than

Description:

If `ACC` is less than `K`, branch to absolute address `k`.

Operation:

If ACC < K then k → PC else PC + 1 → PC

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

Opcode:

0011 1010	KKKK KKKK	kkkk kkkk	

Status register:

						O	Z
						-	-

4.57 BRLTR - Branch if less than register

Description:

If `ACC` is less than `R`, branch to absolute address `k`.

Operation:

If ACC < R then k → PC else PC + 1 → PC

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

Opcode:

0011 1011	RRRR RRRR	kkkk kkkk	

Status register:

						O	Z
						-	-

4.58 BRLTRW - Branch if less than register and word

Description:

If `R` is less than `K`, branch to absolute address `k`.

Operation:

If R < K then k → PC else PC + 1 → PC

Syntax	Operands	Program counter
`BRLTRW R, K, k`	$0 \leq K \leq 2^{32} - 1$	k → PC
	$M0 \leq k \leq Mx$	PC + 1 → PC
	$R0 \leq R \leq Rx$	

Opcode:

0011 1100	RRRR RRRR	KKKK KKKK	kkkk kkkk

Status register:

						O	Z
						-	-

4.59 BRLTRR - Branch if less than registers

Description:

If R_1 is less than R_2, branch to absolute address k.

Operation:

If $R_1 < R_2$ then k → PC else PC + 1 → PC

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

Opcode:

0011 1101	R_1R_1 R_1R_1	R_2R_2 R_2R_2	kkkk kkkk

Status register:

						O	Z
						-	-

4.60 BRGT - Branch if greater than

Description:

If `ACC` is greater than `K`, branch to absolute address `k`.

Operation:

If ACC > K then k → PC else PC + 1 → PC

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

Opcode:

0011 1110	KKKK KKKK	kkkk kkkk	

Status register:

						O	Z
						-	-

4.61 BRGTR - Branch if greater than register

Description:

If `ACC` is greater than `R`, branch to absolute address `k`.

Operation:

If ACC > R then k → PC else PC + 1 → PC

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

Opcode:

0011 1111	RRRR RRRR	kkkk kkkk	

Status register:

						O	Z
						-	-

4.62 BRGTRW - Branch if greater than register and word

Description:

If `R` is greater than `K`, branch to absolute address `k`.

Operation:

If R > K then k → PC else PC + 1 → PC

Syntax	Operands	Program counter
`BRGTRW R, K, k`	$0 \leq K \leq 2^{32} - 1$ $M0 \leq k \leq Mx$ $R0 \leq R \leq Rx$	k → PC PC + 1 → PC

Opcode:

0100 0000	RRRR RRRR	KKKK KKKK	kkkk kkkk

Status register:

						O	Z
						-	-

4.63 BRGTRR - Branch if greater than registers

Description:

If R_1 is greater than R_2, branch to absolute address k.

Operation:

If $R_1 > R_2$ then k → PC else PC + 1 → PC

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

Opcode:

0100 0001	R_1R_1 R_1R_1	R_2R_2 R_2R_2	kkkk kkkk

Status register:

						O	Z
						-	-

4.64 BRLTE - Branch if less than or equals

Description:

If `ACC` is less than or equals `K`, branch to absolute address `k`.

Operation:

If ACC $\leq$ K then k → PC else PC + 1 → PC

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

Opcode:

0100 0010	KKKK KKKK	kkkk kkkk	

Status register:

						O	Z
						-	-

4.65 BRLTER - Branch if less than or equals register

Description:

If `ACC` is less than or equals `R`, branch to absolute address `k`.

Operation:

If ACC $\leq$ R then k → PC else PC + 1 → PC

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

Opcode:

0100 0011	RRRR RRRR	kkkk kkkk	

Status register:

						O	Z
						-	-

4.66 BRLTERW - Branch if less than or equals register and word

Description:

If `R` is less than or equals `K`, branch to absolute address `k`.

Operation:

If R $\leq$ K then k → PC else PC + 1 → PC

Syntax	Operands	Program counter
`BRLTERW R, K, k`	$0 \leq K \leq 2^{32} - 1$ $M0 \leq k \leq Mx$ $R0 \leq R \leq Rx$	k → PC PC + 1 → PC

Opcode:

0100 0100	RRRR RRRR	KKKK KKKK	kkkk kkkk

Status register:

						O	Z
						-	-

4.67 BRLTERR - Branch if less than or equals registers

Description:

If R_1 is less than or equals R_2, branch to absolute address k.

Operation:

If $R_1 \leq R_2$ then k → PC else PC + 1 → PC

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

Opcode:

0100 0101	R_1R_1 R_1R_1	R_2R_2 R_2R_2	kkkk kkkk

Status register:

						O	Z
						-	-

4.68 BRGTE - Branch if greater than or equals

Description:

If `ACC` is greater than or equals `K`, branch to absolute address `k`.

Operation:

If ACC $\geq$ K then k → PC else PC + 1 → PC

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

Opcode:

0100 0110	KKKK KKKK	kkkk kkkk	

Status register:

						O	Z
						-	-

4.69 BRGTER - Branch if greater than or equals register

Description:

If `ACC` is greater than or equals `R`, branch to absolute address `k`.

Operation:

If ACC $\geq$ R then k → PC else PC + 1 → PC

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

Opcode:

0100 0111	RRRR RRRR	kkkk kkkk	

Status register:

						O	Z
						-	-

4.70 BRGTERW - Branch if greater than or equals register and word

Description:

If `R` is greater than or equals `K`, branch to absolute address `k`.

Operation:

If R $\geq$ K then k → PC else PC + 1 → PC

Syntax	Operands	Program counter
`BRGTERW R, K, k`	$0 \leq K \leq 2^{32} - 1$ $M0 \leq k \leq Mx$ $R0 \leq R \leq Rx$	k → PC PC + 1 → PC

Opcode:

0100 1000	RRRR RRRR	KKKK KKKK	kkkk kkkk

Status register:

						O	Z
						-	-

4.71 BRGTERR - Branch if greater than or equals registers

Description:

If R_1 is greater than or equals R_2, branch to absolute address k.

Operation:

If $R_1 \geq R_2$ then k → PC else PC + 1 → PC

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

Opcode:

0100 1001	R_1R_1 R_1R_1	R_2R_2 R_2R_2	kkkk kkkk

Status register:

						O	Z
						-	-

4.72 JMP - Jump

Description:

Jump to absolute address `k`.

Operation:

k → PC

Syntax	Operands	Program counter
`JMP k`	$M0 \leq k \leq Mx$	k → PC

Opcode:

0000 0001	kkkk kkkk		

Status register:

						O	Z
						-	-

4.73 CALL - Call subroutine

Description:

Push `SF` onto the stack and jump to absolute address `k`.

Operation:

SF → PC, k → PC

Syntax	Operands	Program counter
`CALL k`	$M0 \leq k \leq Mx$	k → PC

Opcode:

0000 0010	kkkk kkkk		

Status register:

						O	Z
						-	-

4.74 CALLR - Call subroutine from register

Description:

Push `SF` onto the stack and jump to absolute address `R`.

Operation:

SF → PC, R → PC

Syntax	Operands	Program counter
`CALLR R`	$R0 \leq R \leq Rx$	k → PC

Opcode:

0000 0011	RRRR RRRR		

Status register:

						O	Z
						-	-

4.75 RET - Return from subroutine

Description:

Pop `SF` from stack and return from subroutine.

Operation:

SF → R0 - Rx, SF → PC

Syntax	Operands	Program counter
`RET`	None	SF → PC

Opcode:

0000 0100			

Status register:

						O	Z
						-	-