|  |
| --- |
| 0xVM |
| 0xVM datasheet |

|  |
| --- |
|  |

[Glossary 4](#_Toc81312140)

[Registers and Operands 4](#_Toc81312141)

[Status Register 5](#_Toc81312142)

[Instruction set 6](#_Toc81312143)

[HALT – Halt 6](#_Toc81312144)

[NOP – No Operation 7](#_Toc81312145)

[MOVR – Move to register 8](#_Toc81312146)

[MOVM – Move to memory 9](#_Toc81312147)

[MOVRR – Move register to register 10](#_Toc81312148)

[MOVRM – Move register to memory 11](#_Toc81312149)

[MOVMR – Move memory to register 12](#_Toc81312150)

[MOVRPR – Move register pointer to register 13](#_Toc81312151)

[MOVROR – Move register plus offset pointer to register 14](#_Toc81312152)

[LOAD – Load buffer 15](#_Toc81312153)

[LOADR – Load buffer 16](#_Toc81312154)

[LOADM – Load buffer 17](#_Toc81312155)

[STORE – Store buffer 18](#_Toc81312156)

[STORER – Store buffer 19](#_Toc81312157)

[STOREM – Store buffer 20](#_Toc81312158)

[POP – Pop 21](#_Toc81312159)

[PUSH – Push 22](#_Toc81312160)

[PUSHR – Push register 23](#_Toc81312161)

[ADD – Add 24](#_Toc81312162)

[ADDR – Add register 25](#_Toc81312163)

[SUB – Subtract 26](#_Toc81312164)

[SUBWR – Subtract register from word 27](#_Toc81312165)

[SUBR – Subtract two registers 28](#_Toc81312166)

[MULT – Multiply 29](#_Toc81312167)

[MULTR – Multiply registers 30](#_Toc81312168)

[DIV – Divide 31](#_Toc81312169)

[DIVWR – Divide word by register 32](#_Toc81312170)

[DIVR – Divide registers 33](#_Toc81312171)

[INC – Increment 34](#_Toc81312172)

[DEC – Decrement 35](#_Toc81312173)

[LSF – Left shift 36](#_Toc81312174)

[LSFR – Left shift by register 37](#_Toc81312175)

[RSF – Right shift 38](#_Toc81312176)

[RSFR – Right shift by register 39](#_Toc81312177)

[WLSF – Wrapping left shift 40](#_Toc81312178)

[WLSFR – Wrapping left shift by register 41](#_Toc81312179)

[WRSF – Wrapping right shift 42](#_Toc81312180)

[WRSFR – Wrapping right shift by register 43](#_Toc81312181)

[AND – Bitwise and 44](#_Toc81312182)

[ANDR – Left shift by register 45](#_Toc81312183)

[OR – Bitwise or 46](#_Toc81312184)

[ORR – Bitwise or by register 47](#_Toc81312185)

[XOR – Exclusive or 48](#_Toc81312186)

[XORR – Exclusive or by register 49](#_Toc81312187)

[NOT – Not 50](#_Toc81312188)

[BRBS – Branch if bit is set 51](#_Toc81312189)

[BRBC – Branch if bit is clear 52](#_Toc81312190)

[BREQ – Branch if equal 53](#_Toc81312191)

[BREQR – Branch if equal register 54](#_Toc81312192)

[BREQRW– Branch if equal register and word 55](#_Toc81312193)

[BREQRR – Branch if equal registers 56](#_Toc81312194)

[BRNQ – Branch if not equal 57](#_Toc81312195)

[BRNQR – Branch if not equal register 58](#_Toc81312196)

[BRNQRW– Branch if not equal register and word 59](#_Toc81312197)

[BRNQRR – Branch if not equal registers 60](#_Toc81312198)

[BRLT – Branch if less than 61](#_Toc81312199)

[BRLTR – Branch if less than register 62](#_Toc81312200)

[BRLTRW– Branch if register less than word 63](#_Toc81312201)

[BRLTRR – Branch if register less than register 64](#_Toc81312202)

[BRGT – Branch if greater than 65](#_Toc81312203)

[BRGTR – Branch if greater than register 66](#_Toc81312204)

[BRGTRW– Branch if register greater than word 67](#_Toc81312205)

[BRGTRR – Branch if register greater than register 68](#_Toc81312206)

[BRLTE – Branch if less than or equals 69](#_Toc81312207)

[BRLTER – Branch if less than or equals register 70](#_Toc81312208)

[BRLTERW– Branch if register less than or equals word 71](#_Toc81312209)

[BRLTERR – Branch if register less than or equal register 72](#_Toc81312210)

[BRGTE – Branch if greater than or equals 73](#_Toc81312211)

[BRGTER – Branch if greater than or equals register 74](#_Toc81312212)

[BRGTERW– Branch if register greater than or equals word 75](#_Toc81312213)

[BRGTERR – Branch if register greater than or equal register 76](#_Toc81312214)

[JMP – Jump 77](#_Toc81312215)

[CALL – Call subroutine 78](#_Toc81312216)

[CALLR – Call subroutine from register 79](#_Toc81312217)

[RET – Return from subroutine 80](#_Toc81312218)

Glossary

Registers and Operands

**PC** – Program counter (u32)

Stores the next instruction to execute.

**ACC** – Accumulator (u32)

Stores the result of the last arithmetic operation.

**SP** – Stack pointer (u32)

**FP** – Frame pointer (u32)

**SF** – Stack frame

**Rd** – Destination (and source) register (u32)

**Rr** – Source register (u32)

**K** – Constant data (u32)

**k** – Constant address (u32)

**R** – Value from register (u32)

**mX** – Maximum memory address (u32)

**rX** – Highest register (u32)

**s** – Bitmask for bit in status register (u8)

**sX** – Highest bit in status register (u8)

Status Register

**SR** – Status register

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
|  |  |  |  |  |  | **O** | **Z** |

**Z –** Zero flag

1 if last operation resulted in zero.

**O** – Overflow flag

1 if last operation resulted in an overflow.

Instruction set

HALT – Halt

**Description:**

Halts the program.

**Operation:** None

|  |  |  |
| --- | --- | --- |
| **Syntax** | **Operands** | **Program counter** |
| HALT | None | None |

**Opcode:** 1111 1111

**Status Register:**

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
|  |  |  |  |  |  | **O** | **Z** |
|  |  |  |  |  |  | **-** | **-** |

NOP – No Operation

**Description:**

Does nothing.

|  |  |  |
| --- | --- | --- |
| **Syntax** | **Operands** | **Program counter** |
| NOP | None | PC 🡨 PC + 1 |

**Opcode:** 0000 0000

**Status Register:**

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
|  |  |  |  |  |  | **O** | **Z** |
|  |  |  |  |  |  | **-** | **-** |

MOVR – Move to register

**Description:**

Moves a value into a register.

**Operation:** Rd 🡨 K

|  |  |  |
| --- | --- | --- |
| **Syntax** | **Operands** | **Program counter** |
| MOVR K, Rd |  | PC 🡨 PC + 1 |

**Opcode:** 0001 0000

**Status Register:**

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
|  |  |  |  |  |  | **O** | **Z** |
|  |  |  |  |  |  | - | - |

MOVM – Move to memory

**Description:**

Moves a value into memory.

**Operation:** k 🡨 K

|  |  |  |
| --- | --- | --- |
| **Syntax** | **Operands** | **Program counter** |
| MOVM K, k |  | PC 🡨 PC + 1 |

**Opcode:** 0001 0001

**Status Register:**

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
|  |  |  |  |  |  | **O** | **Z** |
|  |  |  |  |  |  | - | - |

MOVRR – Move register to register

**Description:**

Moves a register into another register.

**Operation:** Rd 🡨 Rr

|  |  |  |
| --- | --- | --- |
| **Syntax** | **Operands** | **Program counter** |
| MOVRR Rr, Rd |  | PC 🡨 PC + 1 |

**Opcode:** 0001 0010

**Status Register:**

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
|  |  |  |  |  |  | **O** | **Z** |
|  |  |  |  |  |  | - | - |

MOVRM – Move register to memory

**Description:**

Moves a register into memory.

**Operation:** k 🡨 Rr

|  |  |  |
| --- | --- | --- |
| **Syntax** | **Operands** | **Program counter** |
| MOVRM Rr, k |  | PC 🡨 PC + 1 |

**Opcode:** 0001 0011

**Status Register:**

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
|  |  |  |  |  |  | **O** | **Z** |
|  |  |  |  |  |  | - | - |

MOVMR – Move memory to register

**Description:**

Moves a value from memory into a register.

**Operation:** Rd 🡨 k

|  |  |  |
| --- | --- | --- |
| **Syntax** | **Operands** | **Program counter** |
| MOVMR k, Rd |  | PC 🡨 PC + 1 |

**Opcode:** 0001 0100

**Status Register:**

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
|  |  |  |  |  |  | **O** | **Z** |
|  |  |  |  |  |  | - | - |

MOVRPR – Move register pointer to register

**Description:**

Moves a value from memory, pointed at by a register to a register.

**Operation:** Rd 🡨 Rr\*

|  |  |  |
| --- | --- | --- |
| **Syntax** | **Operands** | **Program counter** |
| MOVRPR Rr, Rd |  | PC 🡨 PC + 1 |

**Opcode:** 0001 0111

**Status Register:**

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
|  |  |  |  |  |  | **O** | **Z** |
|  |  |  |  |  |  | - | - |

MOVROR – Move register plus offset pointer to register

**Description:**

Moves a value from memory, pointed at by a register plus an offset to a register.

**Operation:** Rd 🡨 Rr\* + K

|  |  |  |
| --- | --- | --- |
| **Syntax** | **Operands** | **Program counter** |
| MOVROR Rr, K, Rd |  | PC 🡨 PC + 1 |

**Opcode:** 0001 1000

**Status Register:**

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
|  |  |  |  |  |  | **O** | **Z** |
|  |  |  |  |  |  | - | - |

LOAD – Load buffer

**Description:**

Loads a byte buffer from an external device into a specified memory area.

**Operation:** k to k + R 🡨 Rr\*

|  |  |  |
| --- | --- | --- |
| **Syntax** | **Operands** | **Program counter** |
| LOAD Rr, R, k |  | PC 🡨 PC + 1 |

**Opcode:** 0001 1001

**Status Register:**

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
|  |  |  |  |  |  | **O** | **Z** |
|  |  |  |  |  |  | - | - |

LOADR – Load buffer

**Description:**

Loads a byte buffer from an external device into a by a register pointed at memory area.

**Operation:** Rd\* to Rd\* + R 🡨 R\*

|  |  |  |
| --- | --- | --- |
| **Syntax** | **Operands** | **Program counter** |
| LOAD Rr, R, Rd |  | PC 🡨 PC + 1 |

**Opcode:** 0001 1010

**Status Register:**

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
|  |  |  |  |  |  | **O** | **Z** |
|  |  |  |  |  |  | - | - |

LOADM – Load buffer

**Description:**

Loads a byte buffer from an external device into a by memory pointed at memory area.

**Operation:** Rd\* to Rd\* + R 🡨 D 🡨 Mr\*

|  |  |  |
| --- | --- | --- |
| **Syntax** | **Operands** | **Program counter** |
| LOAD Rd, R, Mr |  | PC 🡨 PC + 1 |

**Opcode:** 0001 1011

**Status Register:**

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
|  |  |  |  |  |  | **O** | **Z** |
|  |  |  |  |  |  | - | - |

STORE – Store buffer

**Description:**

Stores a byte buffer from a specified memory area into an external device.

**Operation:** Rd\* 🡨 k to k + R

|  |  |  |
| --- | --- | --- |
| **Syntax** | **Operands** | **Program counter** |
| STORE k, R, Rd |  | PC 🡨 PC + 1 |

**Opcode:** 0001 1100

**Status Register:**

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
|  |  |  |  |  |  | **O** | **Z** |
|  |  |  |  |  |  | - | - |

STORER – Store buffer

**Description:**

Stores a byte buffer from a register pointed at memory area into an external device.

**Operation:** Rd\* 🡨 Rr\* to Rr\* + R

|  |  |  |
| --- | --- | --- |
| **Syntax** | **Operands** | **Program counter** |
| STORE Rr, R, Rd |  | PC 🡨 PC + 1 |

**Opcode:** 0001 1101

**Status Register:**

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
|  |  |  |  |  |  | **O** | **Z** |
|  |  |  |  |  |  | - | - |

STOREM – Store buffer

**Description:**

Stores a byte buffer from a memory pointed at memory area into an external device.

**Operation:** Rd\* 🡨 Mr\* to Mr\* + R

|  |  |  |
| --- | --- | --- |
| **Syntax** | **Operands** | **Program counter** |
| STORE Mr, R, Rd |  | PC 🡨 PC + 1 |

**Opcode:** 0001 1110

**Status Register:**

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
|  |  |  |  |  |  | **O** | **Z** |
|  |  |  |  |  |  | - | - |

POP – Pop

**Description:**

Pops a value from the stack into a register and increments SP.

**Operation:** Rd 🡨 k, SP 🡨 SP + 4

|  |  |  |
| --- | --- | --- |
| **Syntax** | **Operands** | **Program counter** |
| POP Rd |  | PC 🡨 PC + 1 |

**Opcode:** 0000 0101

**Status Register:**

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
|  |  |  |  |  |  | **O** | **Z** |
|  |  |  |  |  |  | - | - |

PUSH – Push

**Description:**

Pushes a value onto the stack and decrements SP.

**Operation:** k 🡨 K, SP 🡨 SP - 4

|  |  |  |
| --- | --- | --- |
| **Syntax** | **Operands** | **Program counter** |
| PUSH K |  | PC 🡨 PC + 1 |

**Opcode:** 0001 0101

**Status Register:**

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
|  |  |  |  |  |  | **O** | **Z** |
|  |  |  |  |  |  | - | - |

PUSHR – Push register

**Description:**

Pushes a register onto the stack and decrements SP.

**Operation:** k 🡨 Rr, SP 🡨 SP - 4

|  |  |  |
| --- | --- | --- |
| **Syntax** | **Operands** | **Program counter** |
| PUSHR Rr |  | PC 🡨 PC + 1 |

**Opcode:** 0001 0110

**Status Register:**

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
|  |  |  |  |  |  | **O** | **Z** |
|  |  |  |  |  |  | - | - |

ADD – Add

**Description:**

Adds a value and a register and stores it in ACC.

**Operation:** ACC 🡨 Rr + K

|  |  |  |
| --- | --- | --- |
| **Syntax** | **Operands** | **Program counter** |
| ADD K, Rr |  | PC 🡨 PC + 1 |

**Opcode:** 0010 0000

**Status Register:**

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
|  |  |  |  |  |  | **O** | **Z** |
|  |  |  |  |  |  | x | x |

**Z** – 1 if the operation results in 0

**O** – 1 if the operation overflows

ADDR – Add register

**Description:**

Adds two registers together and stores the result in ACC.

**Operation:** ACC 🡨 Rr1 + Rr2

|  |  |  |
| --- | --- | --- |
| **Syntax** | **Operands** | **Program counter** |
| ADDR Rr1, Rr2 |  | PC 🡨 PC + 1 |

**Opcode:** 0010 0001

**Status Register:**

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
|  |  |  |  |  |  | **O** | **Z** |
|  |  |  |  |  |  | x | x |

**Z** – 1 if the operation results in 0

**O** – 1 if the operation overflows

SUB – Subtract

**Description:**

Subtracts a word from a register.

**Operation:** ACC 🡨 Rr - K

|  |  |  |
| --- | --- | --- |
| **Syntax** | **Operands** | **Program counter** |
| SUB Rr, K |  | PC 🡨 PC + 1 |

**Opcode:** 0010 0010

**Status Register:**

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
|  |  |  |  |  |  | **O** | **Z** |
|  |  |  |  |  |  | x | x |

**Z** – 1 if the operation results in 0

**O** – 1 if the operation overflows

SUBWR – Subtract register from word

**Description:**

Subtracts a register from a word.

**Operation:** ACC 🡨 K - Rr

|  |  |  |
| --- | --- | --- |
| **Syntax** | **Operands** | **Program counter** |
| SUBWR K, Rr |  | PC 🡨 PC + 1 |

**Opcode:** 0010 0011

**Status Register:**

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
|  |  |  |  |  |  | **O** | **Z** |
|  |  |  |  |  |  | x | x |

**Z** – 1 if the operation results in 0

**O** – 1 if the operation overflows

SUBR – Subtract two registers

**Description:**

Subtracts a word from a register.

**Operation:** ACC 🡨 Rr1 – Rr2

|  |  |  |
| --- | --- | --- |
| **Syntax** | **Operands** | **Program counter** |
| SUBR Rr1, Rr2 |  | PC 🡨 PC + 1 |

**Opcode:** 0010 0100

**Status Register:**

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
|  |  |  |  |  |  | **O** | **Z** |
|  |  |  |  |  |  | x | x |

**Z** – 1 if the operation results in 0

**O** – 1 if the operation overflows

MULT – Multiply

**Description:**

Multiplies a word with a register

**Operation:** ACC 🡨 K \* Rr

|  |  |  |
| --- | --- | --- |
| **Syntax** | **Operands** | **Program counter** |
| MULT K, Rr |  | PC 🡨 PC + 1 |

**Opcode:** 0010 0101

**Status Register:**

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
|  |  |  |  |  |  | **O** | **Z** |
|  |  |  |  |  |  | x | x |

**Z** – 1 if the operation results in 0

**O** – 1 if the operation overflows

MULTR – Multiply registers

**Description:**

Multiplies two registers.

**Operation:** ACC 🡨 Rr1 \* Rr2

|  |  |  |
| --- | --- | --- |
| **Syntax** | **Operands** | **Program counter** |
| MULTR Rr1, Rr2 |  | PC 🡨 PC + 1 |

**Opcode:** 0010 0110

**Status Register:**

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
|  |  |  |  |  |  | **O** | **Z** |
|  |  |  |  |  |  | x | x |

**Z** – 1 if the operation results in 0

**O** – 1 if the operation overflows

DIV – Divide

**Description:**

Divides a register by a word.

**Operation:** ACC 🡨 Rr / K

|  |  |  |
| --- | --- | --- |
| **Syntax** | **Operands** | **Program counter** |
| DIV Rr, K |  | PC 🡨 PC + 1 |

**Opcode:** 0010 0111

**Status Register:**

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
|  |  |  |  |  |  | **O** | **Z** |
|  |  |  |  |  |  | x | x |

**Z** – 1 if the operation results in 0

**O** – 1 if the operation overflows

DIVWR – Divide word by register

**Description:**

Divides a word by a register.

**Operation:** ACC 🡨 K / Rr

|  |  |  |
| --- | --- | --- |
| **Syntax** | **Operands** | **Program counter** |
| DIVWR K, Rr |  | PC 🡨 PC + 1 |

**Opcode:** 0010 1000

**Status Register:**

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
|  |  |  |  |  |  | **O** | **Z** |
|  |  |  |  |  |  | x | x |

**Z** – 1 if the operation results in 0

**O** – 1 if the operation overflows

DIVR – Divide registers

**Description:**

Divides two registers.

**Operation:** ACC 🡨 Rr1 – Rr2

|  |  |  |
| --- | --- | --- |
| **Syntax** | **Operands** | **Program counter** |
| DIVR Rr1, Rr2 |  | PC 🡨 PC + 1 |

**Opcode:** 0010 1001

**Status Register:**

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
|  |  |  |  |  |  | **O** | **Z** |
|  |  |  |  |  |  | x | x |

**Z** – 1 if the operation results in 0

**O** – 1 if the operation overflows

INC – Increment

**Description:**

Increments a register.

**Operation:** Rd 🡨 Rd + 1

|  |  |  |
| --- | --- | --- |
| **Syntax** | **Operands** | **Program counter** |
| INC Rd |  | PC 🡨 PC + 1 |

**Opcode:** 0010 1010

**Status Register:**

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
|  |  |  |  |  |  | **O** | **Z** |
|  |  |  |  |  |  | x | x |

**Z** – 1 if the operation results in 0

**O** – 1 if the operation overflows

DEC – Decrement

**Description:**

Decrements a register.

**Operation:** Rd 🡨 Rd - 1

|  |  |  |
| --- | --- | --- |
| **Syntax** | **Operands** | **Program counter** |
| DEC Rd |  | PC 🡨 PC + 1 |

**Opcode:** 0010 1011

**Status Register:**

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
|  |  |  |  |  |  | **O** | **Z** |
|  |  |  |  |  |  | x | x |

**Z** – 1 if the operation results in 0

**O** – 1 if the operation overflows

LSF – Left shift

**Description:**

Shifts a register to the left n times.

**Operation:** Rd 🡨 Rd << K

|  |  |  |
| --- | --- | --- |
| **Syntax** | **Operands** | **Program counter** |
| LSF Rd, K |  | PC 🡨 PC + 1 |

**Opcode:** 0101 0000

**Status Register:**

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
|  |  |  |  |  |  | **O** | **Z** |
|  |  |  |  |  |  | - | x |

**Z** – 1 if the operation results in 0

LSFR – Left shift by register

**Description:**

Shifts a register to the left n times.

**Operation:** Rd 🡨 Rd << Rr

|  |  |  |
| --- | --- | --- |
| **Syntax** | **Operands** | **Program counter** |
| LSFR Rd, Rr |  | PC 🡨 PC + 1 |

**Opcode:** 0101 0001

**Status Register:**

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
|  |  |  |  |  |  | **O** | **Z** |
|  |  |  |  |  |  | - | x |

**Z** – 1 if the operation results in 0

RSF – Right shift

**Description:**

Shifts a register to the right n times.

**Operation:** Rd 🡨 Rd >> K

|  |  |  |
| --- | --- | --- |
| **Syntax** | **Operands** | **Program counter** |
| RSF Rd, K |  | PC 🡨 PC + 1 |

**Opcode:** 0101 0010

**Status Register:**

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
|  |  |  |  |  |  | **O** | **Z** |
|  |  |  |  |  |  | - | x |

**Z** – 1 if the operation results in 0

RSFR – Right shift by register

**Description:**

Shifts a register to the right n times.

**Operation:** Rd 🡨 Rd >> Rr

|  |  |  |
| --- | --- | --- |
| **Syntax** | **Operands** | **Program counter** |
| RSFR Rd, Rr |  | PC 🡨 PC + 1 |

**Opcode:** 0101 0011

**Status Register:**

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
|  |  |  |  |  |  | **O** | **Z** |
|  |  |  |  |  |  | - | x |

**Z** – 1 if the operation results in 0

WLSF – Wrapping left shift

**Description:**

Shifts a register to the left n times and wraps the bits.

**Operation:** Rd 🡨 Rd << K

|  |  |  |
| --- | --- | --- |
| **Syntax** | **Operands** | **Program counter** |
| WLSF Rd, K |  | PC 🡨 PC + 1 |

**Opcode:** 0101 0100

**Status Register:**

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
|  |  |  |  |  |  | **O** | **Z** |
|  |  |  |  |  |  | - | x |

**Z** – 1 if the operation results in 0

WLSFR – Wrapping left shift by register

**Description:**

Shifts a register to the left n times and wraps the bits.

**Operation:** Rd 🡨 Rd << Rr

|  |  |  |
| --- | --- | --- |
| **Syntax** | **Operands** | **Program counter** |
| WLSFR Rd, Rr |  | PC 🡨 PC + 1 |

**Opcode:** 0101 0101

**Status Register:**

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
|  |  |  |  |  |  | **O** | **Z** |
|  |  |  |  |  |  | - | x |

**Z** – 1 if the operation results in 0

WRSF – Wrapping right shift

**Description:**

Shifts a register to the right n times and wraps the bits.

**Operation:** Rd 🡨 Rd >> K

|  |  |  |
| --- | --- | --- |
| **Syntax** | **Operands** | **Program counter** |
| WRSF Rd, K |  | PC 🡨 PC + 1 |

**Opcode:** 0101 0110

**Status Register:**

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
|  |  |  |  |  |  | **O** | **Z** |
|  |  |  |  |  |  | - | x |

**Z** – 1 if the operation results in 0

WRSFR – Wrapping right shift by register

**Description:**

Shifts a register to the right n times and wraps the bits.

**Operation:** Rd 🡨 Rd >> Rr

|  |  |  |
| --- | --- | --- |
| **Syntax** | **Operands** | **Program counter** |
| WRSFR Rd, Rr |  | PC 🡨 PC + 1 |

**Opcode:** 0101 0111

**Status Register:**

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
|  |  |  |  |  |  | **O** | **Z** |
|  |  |  |  |  |  | - | x |

**Z** – 1 if the operation results in 0

AND – Bitwise and

**Description:**

Masks a register with a bitwise AND-mask.

**Operation:** Rd 🡨 Rd & K

|  |  |  |
| --- | --- | --- |
| **Syntax** | **Operands** | **Program counter** |
| AND Rd, K |  | PC 🡨 PC + 1 |

**Opcode:** 0101 1000

**Status Register:**

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
|  |  |  |  |  |  | **O** | **Z** |
|  |  |  |  |  |  | - | x |

**Z** – 1 if the operation results in 0

ANDR – Left shift by register

**Description:**

Masks a register with a bitwise AND-mask.

**Operation:** Rd 🡨 Rd & Rr

|  |  |  |
| --- | --- | --- |
| **Syntax** | **Operands** | **Program counter** |
| ANDR Rd, Rr |  | PC 🡨 PC + 1 |

**Opcode:** 0101 1001

**Status Register:**

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
|  |  |  |  |  |  | **O** | **Z** |
|  |  |  |  |  |  | - | x |

**Z** – 1 if the operation results in 0

OR – Bitwise or

**Description:**

Masks a register with a bitwise OR-mask.

**Operation:** Rd 🡨 Rd | K

|  |  |  |
| --- | --- | --- |
| **Syntax** | **Operands** | **Program counter** |
| OR Rd, K |  | PC 🡨 PC + 1 |

**Opcode:** 0101 1010

**Status Register:**

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
|  |  |  |  |  |  | **O** | **Z** |
|  |  |  |  |  |  | - | x |

**Z** – 1 if the operation results in 0

ORR – Bitwise or by register

**Description:**

Masks a register with a bitwise OR-mask.

**Operation:** Rd 🡨 Rd | Rr

|  |  |  |
| --- | --- | --- |
| **Syntax** | **Operands** | **Program counter** |
| ORR Rd, Rr |  | PC 🡨 PC + 1 |

**Opcode:** 0101 1011

**Status Register:**

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
|  |  |  |  |  |  | **O** | **Z** |
|  |  |  |  |  |  | - | x |

**Z** – 1 if the operation results in 0

XOR – Exclusive or

**Description:**

Masks a register with a bitwise XOR-mask.

**Operation:** Rd 🡨 Rd ^ K

|  |  |  |
| --- | --- | --- |
| **Syntax** | **Operands** | **Program counter** |
| XOR Rd, K |  | PC 🡨 PC + 1 |

**Opcode:** 0101 1100

**Status Register:**

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
|  |  |  |  |  |  | **O** | **Z** |
|  |  |  |  |  |  | - | x |

**Z** – 1 if the operation results in 0

XORR – Exclusive or by register

**Description:**

Masks a register with a bitwise XOR-mask.

**Operation:** Rd 🡨 Rd ^ Rr

|  |  |  |
| --- | --- | --- |
| **Syntax** | **Operands** | **Program counter** |
| XORR Rd, Rr |  | PC 🡨 PC + 1 |

**Opcode:** 0101 1101

**Status Register:**

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
|  |  |  |  |  |  | **O** | **Z** |
|  |  |  |  |  |  | - | x |

**Z** – 1 if the operation results in 0

NOT – Not

**Description:**

Flips all bits in a register.

**Operation:** Rd 🡨 !Rd

|  |  |  |
| --- | --- | --- |
| **Syntax** | **Operands** | **Program counter** |
| NOT Rd |  | PC 🡨 PC + 1 |

**Opcode:** 0101 1110

**Status Register:**

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
|  |  |  |  |  |  | **O** | **Z** |
|  |  |  |  |  |  | - | x |

**Z** – 1 if the operation results in 0

BRBS – Branch if bit is set

**Description:**

Conditional branch. Checks if a bit in the SR is set and branches to the absolute address if true.

**Operation:** If SR(s) = 1 then PC 🡨 k else PC 🡨 PC + 1

|  |  |  |
| --- | --- | --- |
| **Syntax** | **Operands** | **Program counter** |
| BRBS s, k |  | PC 🡨 k  PC 🡨 PC + 1 |

**Opcode:** 0011 0000

**Status Register:**

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
|  |  |  |  |  |  | **O** | **Z** |
|  |  |  |  |  |  | - | - |

BRBC – Branch if bit is clear

**Description:**

Conditional branch. Checks if a bit in the SR is clear and branches to the absolute address if true.

**Operation:** If SR(s) = 0 then PC 🡨 k else PC 🡨 PC + 1

|  |  |  |
| --- | --- | --- |
| **Syntax** | **Operands** | **Program counter** |
| BRBC s, k |  | PC 🡨 k  PC 🡨 PC + 1 |

**Opcode:** 0011 0001

**Status Register:**

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
|  |  |  |  |  |  | **O** | **Z** |
|  |  |  |  |  |  | - | - |

BREQ – Branch if equal

**Description:**

Conditional branch. Checks if the ACC is equal to a value and branches to the absolute if true.

**Operation:** If ACC = K then PC 🡨 k else PC 🡨 PC + 1

|  |  |  |
| --- | --- | --- |
| **Syntax** | **Operands** | **Program counter** |
| BREQ K, k |  | PC 🡨 k  PC 🡨 PC + 1 |

**Opcode:** 0011 0010

**Status Register:**

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
|  |  |  |  |  |  | **O** | **Z** |
|  |  |  |  |  |  | - | - |

BREQR – Branch if equal register

**Description:**

Conditional branch. Checks if the ACC is equal to a register and branches to the absolute if true.

**Operation:** If ACC = Rr then PC 🡨 k else PC 🡨 PC + 1

|  |  |  |
| --- | --- | --- |
| **Syntax** | **Operands** | **Program counter** |
| BREQR Rr, k |  | PC 🡨 k  PC 🡨 PC + 1 |

**Opcode:** 0011 0011

**Status Register:**

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
|  |  |  |  |  |  | **O** | **Z** |
|  |  |  |  |  |  | - | - |

BREQRW– Branch if equal register and word

**Description:**

Conditional branch. Checks if a register and an immediate value are equal and branches to the absolute if true.

**Operation:** If Rr = K then PC 🡨 k else PC 🡨 PC + 1

|  |  |  |
| --- | --- | --- |
| **Syntax** | **Operands** | **Program counter** |
| BREQRW Rr, K, k |  | PC 🡨 k  PC 🡨 PC + 1 |

**Opcode:** 0011 0100

**Status Register:**

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
|  |  |  |  |  |  | **O** | **Z** |
|  |  |  |  |  |  | - | - |

BREQRR – Branch if equal registers

**Description:**

Conditional branch. Checks if two registers are equal and branches to the absolute if true.

**Operation:** If Rr1 = Rr2 then PC 🡨 k else PC 🡨 PC + 1

|  |  |  |
| --- | --- | --- |
| **Syntax** | **Operands** | **Program counter** |
| BREQRR Rr1, Rr2, k |  | PC 🡨 k  PC 🡨 PC + 1 |

**Opcode:** 0011 0101

**Status Register:**

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
|  |  |  |  |  |  | **O** | **Z** |
|  |  |  |  |  |  | - | - |

BRNQ – Branch if not equal

**Description:**

Conditional branch. Checks if the ACC is not equal to a value and branches to the absolute if true.

**Operation:** If ACC ≠ K then PC 🡨 k else PC 🡨 PC + 1

|  |  |  |
| --- | --- | --- |
| **Syntax** | **Operands** | **Program counter** |
| BREQ K, k |  | PC 🡨 k  PC 🡨 PC + 1 |

**Opcode:** 0011 0110

**Status Register:**

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
|  |  |  |  |  |  | **O** | **Z** |
|  |  |  |  |  |  | - | - |

BRNQR – Branch if not equal register

**Description:**

Conditional branch. Checks if the ACC is not equal to a register and branches to the absolute if true.

**Operation:** If ACC ≠ Rr then PC 🡨 k else PC 🡨 PC + 1

|  |  |  |
| --- | --- | --- |
| **Syntax** | **Operands** | **Program counter** |
| BRNQR Rr, k |  | PC 🡨 k  PC 🡨 PC + 1 |

**Opcode:** 0011 0111

**Status Register:**

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
|  |  |  |  |  |  | **O** | **Z** |
|  |  |  |  |  |  | - | - |

BRNQRW– Branch if not equal register and word

**Description:**

Conditional branch. Checks if a register and an immediate value are not equal and branches to the absolute if true.

**Operation:** If Rr ≠ K then PC 🡨 k else PC 🡨 PC + 1

|  |  |  |
| --- | --- | --- |
| **Syntax** | **Operands** | **Program counter** |
| BRNQRW Rr, K, k |  | PC 🡨 k  PC 🡨 PC + 1 |

**Opcode:** 0011 1000

**Status Register:**

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
|  |  |  |  |  |  | **O** | **Z** |
|  |  |  |  |  |  | - | - |

BRNQRR – Branch if not equal registers

**Description:**

Conditional branch. Checks if two registers are not equal and branches to the absolute if true.

**Operation:** If Rr1 ≠ Rr2 then PC 🡨 k else PC 🡨 PC + 1

|  |  |  |
| --- | --- | --- |
| **Syntax** | **Operands** | **Program counter** |
| BRNQRR Rr1, Rr2, k |  | PC 🡨 k  PC 🡨 PC + 1 |

**Opcode:** 0011 1001

**Status Register:**

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
|  |  |  |  |  |  | **O** | **Z** |
|  |  |  |  |  |  | - | - |

BRLT – Branch if less than

**Description:**

Conditional branch. Checks if the ACC is less than a value and branches to the absolute if true.

**Operation:** If ACC < K then PC 🡨 k else PC 🡨 PC + 1

|  |  |  |
| --- | --- | --- |
| **Syntax** | **Operands** | **Program counter** |
| BRLT K, k |  | PC 🡨 k  PC 🡨 PC + 1 |

**Opcode:** 0011 1010

**Status Register:**

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
|  |  |  |  |  |  | **O** | **Z** |
|  |  |  |  |  |  | - | - |

BRLTR – Branch if less than register

**Description:**

Conditional branch. Checks if the ACC is less than a register and branches to the absolute if true.

**Operation:** If ACC < Rr then PC 🡨 k else PC 🡨 PC + 1

|  |  |  |
| --- | --- | --- |
| **Syntax** | **Operands** | **Program counter** |
| BRLTR Rr, k |  | PC 🡨 k  PC 🡨 PC + 1 |

**Opcode:** 0011 1011

**Status Register:**

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
|  |  |  |  |  |  | **O** | **Z** |
|  |  |  |  |  |  | - | - |

BRLTRW– Branch if register less than word

**Description:**

Conditional branch. Checks if a register is less than an immediate value and branches to the absolute if true.

**Operation:** If Rr < K then PC 🡨 k else PC 🡨 PC + 1

|  |  |  |
| --- | --- | --- |
| **Syntax** | **Operands** | **Program counter** |
| BRLTRW Rr, K, k |  | PC 🡨 k  PC 🡨 PC + 1 |

**Opcode:** 0011 1100

**Status Register:**

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
|  |  |  |  |  |  | **O** | **Z** |
|  |  |  |  |  |  | - | - |

BRLTRR – Branch if register less than register

**Description:**

Conditional branch. Checks if register is less than register and branches to the absolute if true.

**Operation:** If Rr1 < Rr2 then PC 🡨 k else PC 🡨 PC + 1

|  |  |  |
| --- | --- | --- |
| **Syntax** | **Operands** | **Program counter** |
| BRLTRR Rr1, Rr2, k |  | PC 🡨 k  PC 🡨 PC + 1 |

**Opcode:** 0011 1101

**Status Register:**

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
|  |  |  |  |  |  | **O** | **Z** |
|  |  |  |  |  |  | - | - |

BRGT – Branch if greater than

**Description:**

Conditional branch. Checks if the ACC is greater than a value and branches to the absolute if true.

**Operation:** If ACC > K then PC 🡨 k else PC 🡨 PC + 1

|  |  |  |
| --- | --- | --- |
| **Syntax** | **Operands** | **Program counter** |
| BRGT K, k |  | PC 🡨 k  PC 🡨 PC + 1 |

**Opcode:** 0011 1110

**Status Register:**

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
|  |  |  |  |  |  | **O** | **Z** |
|  |  |  |  |  |  | - | - |

BRGTR – Branch if greater than register

**Description:**

Conditional branch. Checks if the ACC is greater than a register and branches to the absolute if true.

**Operation:** If ACC > Rr then PC 🡨 k else PC 🡨 PC + 1

|  |  |  |
| --- | --- | --- |
| **Syntax** | **Operands** | **Program counter** |
| BRGTR Rr, k |  | PC 🡨 k  PC 🡨 PC + 1 |

**Opcode:** 0011 1111

**Status Register:**

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
|  |  |  |  |  |  | **O** | **Z** |
|  |  |  |  |  |  | - | - |

BRGTRW– Branch if register greater than word

**Description:**

Conditional branch. Checks if a register is greater than an immediate value and branches to the absolute if true.

**Operation:** If Rr > K then PC 🡨 k else PC 🡨 PC + 1

|  |  |  |
| --- | --- | --- |
| **Syntax** | **Operands** | **Program counter** |
| BRGTRW Rr, K, k |  | PC 🡨 k  PC 🡨 PC + 1 |

**Opcode:** 0100 0000

**Status Register:**

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
|  |  |  |  |  |  | **O** | **Z** |
|  |  |  |  |  |  | - | - |

BRGTRR – Branch if register greater than register

**Description:**

Conditional branch. Checks if register is greater than register and branches to the absolute if true.

**Operation:** If Rr1 > Rr2 then PC 🡨 k else PC 🡨 PC + 1

|  |  |  |
| --- | --- | --- |
| **Syntax** | **Operands** | **Program counter** |
| BRGTRR Rr1, Rr2, k |  | PC 🡨 k  PC 🡨 PC + 1 |

**Opcode:** 0100 0001

**Status Register:**

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
|  |  |  |  |  |  | **O** | **Z** |
|  |  |  |  |  |  | - | - |

BRLTE – Branch if less than or equals

**Description:**

Conditional branch. Checks if the ACC is less than or equals a value and branches to the absolute if true.

**Operation:** If ACC ≤ K then PC 🡨 k else PC 🡨 PC + 1

|  |  |  |
| --- | --- | --- |
| **Syntax** | **Operands** | **Program counter** |
| BRLTE K, k |  | PC 🡨 k  PC 🡨 PC + 1 |

**Opcode:** 0100 0010

**Status Register:**

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
|  |  |  |  |  |  | **O** | **Z** |
|  |  |  |  |  |  | - | - |

BRLTER – Branch if less than or equals register

**Description:**

Conditional branch. Checks if the ACC is less than or equals a register and branches to the absolute if true.

**Operation:** If ACC ≤ Rr then PC 🡨 k else PC 🡨 PC + 1

|  |  |  |
| --- | --- | --- |
| **Syntax** | **Operands** | **Program counter** |
| BRLTER Rr, k |  | PC 🡨 k  PC 🡨 PC + 1 |

**Opcode:** 0100 0011

**Status Register:**

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
|  |  |  |  |  |  | **O** | **Z** |
|  |  |  |  |  |  | - | - |

BRLTERW– Branch if register less than or equals word

**Description:**

Conditional branch. Checks if a register is less than or equals an immediate value and branches to the absolute if true.

**Operation:** If Rr ≤ K then PC 🡨 k else PC 🡨 PC + 1

|  |  |  |
| --- | --- | --- |
| **Syntax** | **Operands** | **Program counter** |
| BRLTERW Rr, K, k |  | PC 🡨 k  PC 🡨 PC + 1 |

**Opcode:** 0100 0100

**Status Register:**

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
|  |  |  |  |  |  | **O** | **Z** |
|  |  |  |  |  |  | - | - |

BRLTERR – Branch if register less than or equal register

**Description:**

Conditional branch. Checks if register is less than or equal register and branches to the absolute if true.

**Operation:** If Rr1 ≤ Rr2 then PC 🡨 k else PC 🡨 PC + 1

|  |  |  |
| --- | --- | --- |
| **Syntax** | **Operands** | **Program counter** |
| BRLTERR Rr1, Rr2, k |  | PC 🡨 k  PC 🡨 PC + 1 |

**Opcode:** 0100 0101

**Status Register:**

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
|  |  |  |  |  |  | **O** | **Z** |
|  |  |  |  |  |  | - | - |

BRGTE – Branch if greater than or equals

**Description:**

Conditional branch. Checks if the ACC is greater than or equals a value and branches to the absolute if true.

**Operation:** If ACC ≥ K then PC 🡨 k else PC 🡨 PC + 1

|  |  |  |
| --- | --- | --- |
| **Syntax** | **Operands** | **Program counter** |
| BRGTE K, k |  | PC 🡨 k  PC 🡨 PC + 1 |

**Opcode:** 0100 0110

**Status Register:**

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
|  |  |  |  |  |  | **O** | **Z** |
|  |  |  |  |  |  | - | - |

BRGTER – Branch if greater than or equals register

**Description:**

Conditional branch. Checks if the ACC is greater than or equals a register and branches to the absolute if true.

**Operation:** If ACC ≥ Rr then PC 🡨 k else PC 🡨 PC + 1

|  |  |  |
| --- | --- | --- |
| **Syntax** | **Operands** | **Program counter** |
| BRGTER Rr, k |  | PC 🡨 k  PC 🡨 PC + 1 |

**Opcode:** 0100 0111

**Status Register:**

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
|  |  |  |  |  |  | **O** | **Z** |
|  |  |  |  |  |  | - | - |

BRGTERW– Branch if register greater than or equals word

**Description:**

Conditional branch. Checks if a register is greater than or equals an immediate value and branches to the absolute if true.

**Operation:** If Rr ≥ K then PC 🡨 k else PC 🡨 PC + 1

|  |  |  |
| --- | --- | --- |
| **Syntax** | **Operands** | **Program counter** |
| BRGTERW Rr, K, k |  | PC 🡨 k  PC 🡨 PC + 1 |

**Opcode:** 0100 1000

**Status Register:**

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
|  |  |  |  |  |  | **O** | **Z** |
|  |  |  |  |  |  | - | - |

BRGTERR – Branch if register greater than or equal register

**Description:**

Conditional branch. Checks if register is greater than or equals register and branches to the absolute if true.

**Operation:** If Rr1 ≥ Rr2 then PC 🡨 k else PC 🡨 PC + 1

|  |  |  |
| --- | --- | --- |
| **Syntax** | **Operands** | **Program counter** |
| BRGTERR Rr1, Rr2, k |  | PC 🡨 k  PC 🡨 PC + 1 |

**Opcode:** 0100 1001

**Status Register:**

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
|  |  |  |  |  |  | **O** | **Z** |
|  |  |  |  |  |  | - | - |

JMP – Jump

**Description:**

Jumps to address without manipulating stack or setting return address.

**Operation:** PC 🡨 k

|  |  |  |
| --- | --- | --- |
| **Syntax** | **Operands** | **Program counter** |
| CALL k |  | PC 🡨 k |

**Opcode:** 0000 0001

**Status Register:**

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
|  |  |  |  |  |  | **O** | **Z** |
|  |  |  |  |  |  | - | - |

CALL – Call subroutine

**Description:**

Calls a subroutine at a specific memory address and creates a stack frame.

**Operation:** SF, PC 🡨 k

|  |  |  |
| --- | --- | --- |
| **Syntax** | **Operands** | **Program counter** |
| CALL k |  | PC 🡨 k |

**Opcode:** 0000 0010

**Status Register:**

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
|  |  |  |  |  |  | **O** | **Z** |
|  |  |  |  |  |  | - | - |

CALLR – Call subroutine from register

**Description:**

Calls a subroutine at a specific memory address from a register and creates a stack frame.

**Operation:** SF, PC 🡨 k

|  |  |  |
| --- | --- | --- |
| **Syntax** | **Operands** | **Program counter** |
| CALLR k |  | PC 🡨 k |

**Opcode:** 0000 0011

**Status Register:**

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
|  |  |  |  |  |  | **O** | **Z** |
|  |  |  |  |  |  | - | - |

RET – Return from subroutine

**Description:**

Recovers the stack frame and returns from the subroutine.

**Operation:** Rd 🡨 SF, PC 🡨 SF

|  |  |  |
| --- | --- | --- |
| **Syntax** | **Operands** | **Program counter** |
| RET | None | PC 🡨 SF |

**Opcode:** 0000 0100

**Status Register:**

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
|  |  |  |  |  |  | **O** | **Z** |
|  |  |  |  |  |  | - | - |