Arithmatic and logic

UMINUS

Usage:
```
UMINUS PARAM1 RESULT
```

Effect:

RESULT = -PARAM1

ADD

Usage:
```
ADD PARAM1 PARAM2 RESULT
```

Effect:

RESULT = PARAM1 + PARAM2

Ef PARAM1 og PARAM2 eru af sitt hvoru tagi (t.d. int og real) þá kemur fram keyrsluvilla.

SUB

Usage:
```
SUB PARAM1 PARAM2 RESULT
```

Effect:
RESULT = PARAM1 – PARAM2

If PARAM1 og PARAM2 have different types (e.g. int and real) a runtime error is raised.

MULT

Usage:
```
MULT PARAM1 PARAM2 RESULT
```

Effect:
RESULT = PARAM1 * PARAM2

If PARAM1 og PARAM2 have different types (e.g. int and real) a runtime error is raised.

DIVIDE

Usage:
```
DIVIDE PARAM1 PARAM2 RESULT
```

Effect:
RESULT = PARAM1 / PARAM2

If PARAM1 og PARAM2 have different types (e.g. int and real) a runtime error is raised.
If PARAM2 is 0 (or 0.0) a runtime error is raised.

Note: The operation gives different results depending on whether the types are int or real.
For example 5.0 / 2.0 = 2.5 and 5 / 2 = 2.

DIV

Usage:
```
DIV PARAM1 PARAM2 RESULT
```
Effect:
Same as DIVIDE.

MOD

Usage:
```
MOD PARAM1 PARAM2 RESULT
```

Effect:
RESULT = PARAM1 % PARAM2
(RESULT gets the reminder of the division PARAM1 / PARAM2)

If PARAM1 og PARAM2 have different types (e.g. int and real) a runtime error is raised.
If PARAM2 is 0 (or 0.0) a runtime error is raised.

AND

Usage:
```
AND PARAM1 PARAM2 RESULT
```

Effect:
RESULT = PARAM1 AND PARAM2
("Bitwise and")

The operation is only defined for integer parameters.

OR

Usage:
```
OR PARAM1 PARAM2 RESULT
```

Effect:
RESULT = PARAM1 OR PARAM2
("Bitwise or")

The operation is only defined for integer parameters.

NOT

Usage:
```
NOT PARAM1 RESULT
```

Effect:
RESULT = NOT PARAM1
("Bitwise not")

The operation is only defined for integer parameters.

Call and branching

CALL

Usage:
```
CALL RESULT
```

Effect:
Calls a procedure with the name specified in RESULT. This name must be defined as a label somewhere in the code.

If the procedure accepts parameters then the parameters must be specified in the correct order using APARAM instructions, right befare the CALL instruction.
To use the parameters inside the procedure, it (that is the procedure) must start with as many FPARAM instructions as there are APARAM instructions preceding the CALL instruction.
Each APARAM instruction associated the procedures parameter with a variable name.

Inside the procedures code, the procedures return value can be set by assigning it to a variable with the same name as the procedure itself (excluding the colon ':' of the label). Note that this return value is write-only, it can not be read inside the procedure. The procedure must end with a RETURN statement.

After the procedure has been executed, it's return value (if any) is acessible in a new variable with the same name as the procedure. Note that this value is read-only it can not be set.

FPARAM

Usage:
```
FPARAM RESULT
```

Effect:
Defines a parameter inside a procedure.

See CALL for more details.

APARAM

Usage:
```
APARAM RESULT
```

Effect:
Defines a value of a parameter to a procedure call.

See CALL for more details.

RETURN

Usage:
```
RETURN
```

Effect:
Terminates a procedure.

Stökkskipanir

GOTO

Usage:
```
GOTO RESULT
```

Effect:
Jumps, unconditionally, to the label given by RESULT.

LT

Usage:
```
LT PARAM1 PARAM2 RESULT
```

Effect:
Jumps to the label given by RESULT if PARAM1 < PARAM2

If PARAM1 og PARAM2 have different types (e.g. int and real) a runtime error is raised.

LE

Usage:
```
LE PARAM1 PARAM2 RESULT
```

Effect:
Jumps to the label given by RESULT if PARAM1 <= PARAM2

If PARAM1 og PARAM2 have different types (e.g. int and real) a runtime error is raised.

GT

Usage:
```
GT PARAM1 PARAM2 RESULT
```

Effect:
Jumps to the label given by RESULT if PARAM1 > PARAM2

If PARAM1 og PARAM2 have different types (e.g. int and real) a runtime error is raised.

GE

Usage:
```
GE PARAM1 PARAM2 RESULT
```

Effect:
Jumps to the label given by RESULT if PARAM1 >= PARAM2

If PARAM1 og PARAM2 have different types (e.g. int and real) a runtime error is raised.

EQ

Usage:
```
EQ PARAM1 PARAM2 RESULT
```

Effect:
Jumps to the label given by RESULT if PARAM1 == PARAM2

If PARAM1 og PARAM2 have different types (e.g. int and real) a runtime error is raised.

NE

Usage:
```
NE PARAM1 PARAM2 RESULT
```

Effect:
Jumps to the label given by RESULT if PARAM1 not equal to PARAM2

If PARAM1 og PARAM2 have different types (e.g. int and real) a runtime error is raised.

Other instructions

VAR

Usage:
```
VAR RESULT
```

Effect:
Defines a variable with the given name in RESULT

ASSIGN

Usage:
```
ASSIGN PARAM1 RESULT
```

Effect:
Assigns the value of PARAM1 to the variable indicated by RESULT

NOOP

Usage:
```
NOOP
```

Effect:
No operation (does nothing).