

: Mjølner BETA Low Level Primitives

Table of Contents

<u>Copyright Notice</u>	1
<u>13. MjølnerBETA Low Level Primitives</u>	3
<u>Introduction</u>	3
<u>Low Level Operations</u>	3
<u>Syntax</u>	3
<u>Addressing Conventions</u>	3
<u>Operations</u>	4
<u>Bitwise logical complement (one's complement)</u>	4
<u>Bitwise logical and, or, exclusive or</u>	4
<u>Shift of a long</u>	4
<u>Get byte/short from a long</u>	4
<u>Put byte/short into a long</u>	5
<u>Get bits from a long</u>	5
<u>Put bits into a long</u>	5
<u>This object</u>	5

Copyright Notice

Mjølner Informatics Report August 1999

Copyright © 1999 [Mjølner Informatics](#).

All rights reserved.

No part of this document may be copied or distributed
without the prior written permission of Mjølner Informatics

BETA Language Modifications

13. Mjølner BETA Low Level Primitives

- [Introduction](#)
- [Low Level Operations](#)
- [Syntax](#)
- [Addressing Conventions](#)
- [Operations](#)
 - ◆ [Bitwise logical complement \(one's complement\)](#)
 - ◆ [Bitwise logical and, or, exclusive or](#)
 - ◆ [Shift of a long](#)
 - ◆ [Get byte/short from a long](#)
 - ◆ [Put byte/short into a long](#)
 - ◆ [Get bits from a long](#)
 - ◆ [Put bits into a long](#)
 - ◆ [This object](#)

Introduction

This document describes the semantics of the low-level primitives available in the Mjølner implementation of the BETA language. There are currently some syntactic inconveniences. These may be fixed with a grammar change in a future version.

Low Level Operations

Low level operations on bits, bytes and words are available as described below. Use of these operations may in general be platform dependent.

Syntax

The syntax is as follows

`%op`

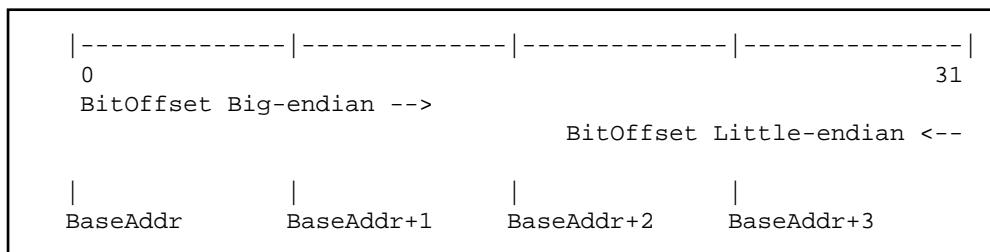
i.e., the `%` indicates, that `op` is a special low-level operation.

In the following, `E`, `val`, and `inx` are assumed to be integer evaluations, `A` is an integer object, and `R` is a repetition object.

Addressing Conventions

The addressing conventions of bytes, words, longs and bitfields follow the big-endian (Motorola, SPARC etc.) conventions:

31		long[0]		0
15	word[0]	0 15	word[1]	0
7	byte[0]	0 7	byte[1]	0 7
	byte[2]	0 7	byte[3]	0



Notice that a BitOffset is addressed from the most significant bit on big-endian architectures, and from the the least significant bit on little-endian architectures (nti, linux).

Operations

The following operations are available.

Bitwise logical complement (one's complement)

```
OP:      %Bnot
usage:  %Bnot E
```

Bitwise logical and, or, exclusive or

```
OP:      %Band, %Bor, %Bxor
usage:  E1 OP E2
ex:     E1 %Band E2
```

Note the `B` in these operations - B stands for bitwise. A future version may use the syntax `%and`.

Shift of a long

```
OP:      %srl      shift right logical
        %sll      shift left logical
        %sra      shift right arithmetic
        %sla      shift left arithmetic
        %rор      rotate right
        %rol      rotate left
usage:  E1 OP E2
ex:    E1 %sll E2
```

Get byte/short from a long

```
OP:      byteNo  -> A.%getByte
        shortNo -> A.%getShort
        longNo  -> A.%getLong
        byteNo  -> A.%getSignedByte
        shortNo -> A.%getSignedShort
```

where `byteNo` is an integer-evaluation in [0,3], `shortNo` in [0,1] and `longNo` in [0].

Usage: E1 -> A1.%getByte -> A2
 Ex: 1 -> A.%getByte -> B

Note: byteNo -> A.%getLong is the same as A.

Put byte/short into a long

OP: (val,byteNo) -> A.%putByte
 (val,shortNo) -> A.%putShort
 (val,longNo) -> A.%putLong

The same restrictions for byteNo etc. as in [Get byte/short from a long](#) apply here.

usage: (val,E) -> A.OP
 ex.: (val,3) -> A.%putByte

The same restrictions for byteNo etc. as in [Get byte/short from a long](#) apply here.

Note: (val,E)->A.%putLong is the same as val->A.

Get bits from a long

OP: (pos,width) -> A.%getBits
 (pos,width) -> A.%getSignedBits

where pos, width in [0,31] are integer-evaluations.

usage: (pos,width) -> A.%getBits -> V

Put bits into a long

OP: (val,pos,width) -> A.%putBits

where pos, width in [0,31] are integer-evaluations.

usage: (V,12,4) -> A.%putBits

This object

Note: This operation is needed in some cases where THIS(P) cannot be used. E.g. inside singular objects in the do-part.

Notice that THIS(Object) will NOT work, you must use the operation below:

OP: %thiss object

A reference to the current object is returned.

Usage: %thiss object -> S[]

where s is declared as s: ^Object.

[Mjølner Informatics](#)

BETA Language Modifications