SECTION IV

IBM System/36 Machine Interface

4.1. Introduction

The functions described in this section fall into two general categories. The first category is the System/36 Machine Library **libm**. This library contains functions which are called by code produced by the code generator, primarily to perform operations too lengthy to be expanded inline. These functions are not C-callable. This list appears below in section 4.2, under the heading "Machine Library Functions".

The object form of the Machine Library is normally found in all standard compiler packages under the name **libmxx.36**, where **xx** denotes configuration dependencies.

The second category consists of C-callable routines that provide access to special System/36 instructions. They are packaged in the Base Library, **libb**, and are documented on the following pages in manual page form.

The object form of the Base Library is normally found in all standard compiler packages under the name **libxx.36**, where **xx** denotes configuration dependencies.

4.2. Machine Library Functions

The following is a list of the functions in the System/36 Machine Library libm. Note that you cannot call these functions directly from a C program.

cent - enter C function

cents - enter C function and check stack

cret -^_ return from C function

dadd - add double into double

dcmp - compare two doubles

ddiv - divide double into double

dmul - multiply double into double

dneg - negate double

dsub - subtract double from double

dti - convert double to int

dtf - convert double to float

dtl - convert double to long

dtui - convert double to unsigned int

fadd - add float into float

fcmp - compare two floats

fdiv - divide float into float

fmul - multiply float into float

fneg - negate float

fsub - subtract float from float

ftd - convert float to double

fti - convert float to int

ftl - convert float to long

ftui - convert float to unsigned int

iand - logical and int by int

idiv - divide signed int by int

ilsh - left shift int by count

imod - remainder signed int by int

imul - multiply int by int

ior - logical or int by int

irsh - shift signed int right by count

itd - convert signed integer to double

itf - convert signed integer to float

ixor - logical exclusive or int by int

jltab - perform C long switch statement

jtab - perform C integer switch statement

land - logical and long by long

ldiv - divide signed long by long

llsh - shift long left by count

lmod - remainder signed long by long

lmul - multiply long by long

lor - logical or long by long

lrsh - shift signed long right by count

ltd - convert signed long to double

ltf - convert signed long to float

lxor - logical exclusive or long by long

uidiv - divide unsigned int by int

uimod - remainder unsigned int by int

uirsh - shift unsigned int right by count

uitd - convert unsigned integer to double

uitf - convert unsigned integer to float

uldiv - divide unsigned long by long

ulmod - remainder unsigned long by long

ulrsh - shift unsigned long right by count

ultd - convert unsigned long to double

4.3. Base Library Manual Pages

The following manual pages document C-callable routines from the Base Library libb. For an explanation of the format and effective use of manual pages, see section 1.2.6, entitled "Manual Page Conventions".

NAME

@iand - and integer into integer

SYNOPSIS

```
        mvc ac0+7(2),left
        la  right-1,2      (xr2 points to first byte of right)
        b   @iand
    *   result in ac0+7
```

FUNCTION

@iand "ands" the integer in ac0 by the integer right pointed at by xr2, to obtain the integer intersection.

RETURNS

The value returned is the integer intersection left&right in ac0.

SEE ALSO

@ior, @ixor, @land, @lor, @lxor

NAME

@idiv - divide integer by integer

SYNOPSIS

```
        mvc ac0+7(2),left
        la  right-1,2      (xr2 points to first byte of right)
        b   @idiv
    *   result in ac0+7
```

FUNCTION

@idiv divides the integer in ac0 by the integer right pointed at by xr2, to obtain the integer quotient. No check is made for division by zero, which currently gives a quotient of -1.

RETURNS

The value returned is the integer quotient in ac0.

SEE ALSO

@imod, @uidiv, @uimod, @imul

NAME

@ilsh - integer left shift

SYNOPSIS

```
        mvc ac0+7(2),val
        la  count-1,2      (xr2 points to first byte of right)
        b   @ilsh
    *   result in ac0+7
```

FUNCTION

@ilsh shifts the integer in ac0 by the integer count pointed at by xr2, to obtain the integer result.

RETURNS

The value returned is the integer **val<<count** in ac0.

SEE ALSO

@irsh, @uirsh

NOTES

Negative shift counts give the result 0.

NAME

@imod - remainder integer by integer

SYNOPSIS

```
        mvc ac0+7(2),left
        la  right-1,2      (xr2 points to first byte of right)
        b   @imod
    *   result in ac0+7
```

FUNCTION

@imod divides the integer in ac0 by the integer right pointed at by xr2, to obtain the integer remainder. No check is made for division by zero, which currently gives a quotient of -1.

RETURNS

The value returned is the integer remainder in ac0.

SEE ALSO

@idiv, @uidiv, @uimod, @imul

NAME

@imul - multiply integer by integer

SYNOPSIS

```
        mvc ac0+7(2),left
        la  right-1,2      (xr2 points to first byte of right)
        b   @imul
    *   result in ac0+7
```

FUNCTION

@imul multiplies the integer in ac0 by the integer right pointed at by xr2, to obtain the integer product.

RETURNS

The value returned is the integer product in ac0.

SEE ALSO

@idiv, @uidiv, @imod, @uimod

NAME

@ior - or integer into integer

SYNOPSIS

```
        mvc ac0+7(2),left
        la  right-1,2     (xr2 points to first byte of right)
        b   @ior
    *   result in ac0+7
```

FUNCTION

@ior "ors" the integer in **ac0** by the integer right pointed at by **xr2**, to obtain the integer union.

RETURNS

The value returned is the integer union **left|right** in **ac0**.

SEE ALSO

@iand, @ixor, @land, @lor, @lxor

NAME

@irsh - integer right shift

SYNOPSIS

```
        mvc ac0+7(2),val
        la  count-1,2      (xr2 points to first byte of right)
        b   @irsh
    *   result in ac0+7
```

FUNCTION

@irsh shifts the integer in **ac0** right by the integer count pointed at by **xr2**, to obtain the integer result.

RETURNS

The value returned is the integer **val>>count** in **ac0**.

SEE ALSO

@ilsh, @uirsh

NOTES

Negative shift counts give the result 0 or -1.

NAME

@ixor - xor integer into integer

SYNOPSIS

```
        mvc ac0+7(2),left
        la  right-1,2      (xr2 points to first byte of right)
        b   @ixor
    *   result in ac0+7
```

FUNCTION

@ixor "xors" the integer in **ac0** by the integer right pointed at by **xr2**, to obtain the integer symmetric difference.

RETURNS

The value returned is the integer symmetric difference left^right in ac0.

SEE ALSO

@iand, @ior, @land, @lor, @lxor

NAME

@land - and long into long

SYNOPSIS

```
        mvc ac0+7(4),left
        la  right-3,2      (xr2 points to first byte of right)
        b   @land
    *   result in ac0+7
```

FUNCTION

@land "ands" the long in **ac0** by the long right pointed at by **xr2**, to obtain the long intersection.

RETURNS

The value returned is the long intersection **left&right** in **ac0**.

SEE ALSO

@iand, @ior, @ixor, @lor, @lxor

NAME

@ldiv - divide long by long

SYNOPSIS

```
        mvc ac0+7(4),left
        la  right-3,2      (xr2 points to first byte of right)
        b   @ldiv
    *   result in ac0+7
```

FUNCTION

@ldiv divides the long in ac0 by the long right pointed at by xr2, to obtain the long quotient. No check is made for division by zero, which currently gives a quotient of -1.

RETURNS

The value returned is the long quotient in ac0.

SEE ALSO

@uldiv, @lmod, @lmul, @ulmod

NAME

@llsh - long left shift

SYNOPSIS

```
        mvc ac0+7(4),val
        la  count-1,2      (xr2 points to first byte of right)
        b   @llsh
    *   result in ac0+7
```

FUNCTION

@llsh shifts the long in **ac0** by the integer count pointed at by **xr2**, to obtain the long result.

RETURNS

The value returned is the long **val<<count** in **ac0**.

SEE ALSO

@lrsh, @ulrsh

NOTES

Negative shift counts gives the result 0.

NAME

@lmod - reminder long by long

SYNOPSIS

```
        mvc ac0+7(4),left
        la  right-3,2      (xr2 points to first byte of right)
        b   @lmod
    *   result in ac0+7
```

FUNCTION

@lmod divides the long in ac0 by the long right pointed at by xr2, to obtain the long reminder. No check is made for division by zero, which currently gives an undefined result.

RETURNS

The value returned is the long reminder in ac0.

SEE ALSO

@uldiv, @ldiv, @lmul, @ulmod

NAME

@lmul - multiply long by long

SYNOPSIS

```
        mvc ac0+7(4),left
        la  right-3,2      (xr2 points to first byte of right)
        b   @lmul
    *   result in ac0+7
```

FUNCTION

@lmul multiplies the long in ac0 by the long right pointed at by xr2, to obtain the long product.

RETURNS

The value returned is the long product in ac0.

SEE ALSO

@ldiv, @uldiv, @lmod, @ulmod

NAME

@lor - or long into long

SYNOPSIS

```
        mvc ac0+7(4),left
        la  right-3,2      (xr2 points to first byte of right)
        b   @lor
    *   result in ac0+7
```

FUNCTION

@lor "ors" the long in ac0 by the long right pointed at by xr2, to obtain the long union.

RETURNS

The value returned is the long union left|right in ac0.

SEE ALSO

@iand, @ior, @ixor, @land, @lxor

NAME

@lrsh - long right shift

SYNOPSIS

```
        mvc ac0+7(4),val
        la  count-1,2       (xr2 points to first byte of right)
        b   @lrsh
    *   result in ac0+7
```

FUNCTION

@lrsh shifts the long in ac0 right by the integer count pointed at by **xr2**, to obtain the long result.

RETURNS

The value returned is the long **val>>count** in ac0.

SEE ALSO

@llsh, @ulrsh

NOTES

Negative shift counts gives the result 0 or -1.

NAME

@lxor - xor long into long

SYNOPSIS

```
        mvc ac0+7(4),left
        la  right-3,2      (xr2 points to first byte of right)
        b   @lxor
    *   result in ac0+7
```

FUNCTION

@lxor "xors" the long in ac0 by the long right pointed at by xr2, to obtain the long symmetric difference.

RETURNS

The value returned is the long symmetric difference left^right in ac0.

SEE ALSO

@iand, @ior, @ixor, @land, @lor

NAME

@uidiv - divide unsigned by unsigned

SYNOPSIS

```
        mvc ac0+7(2),left
        la  right-1,2      (xr2 points to first byte of right)
        b   @uidiv
    *   result in ac0+7
```

FUNCTION

@uidiv divides the unsigned in **ac0** by the unsigned right pointed at by **xr2**, to obtain the unsigned quotient. No check is made for division by zero, which currently gives a quotient of -1.

RETURNS

The value returned is the unsigned quotient in **ac0**.

SEE ALSO

@imod, @idiv, @uimod, @imul

NAME

@uimod - remainder unsigned by unsigned

SYNOPSIS

```
        mvc ac0+7(2),left
        la  right-1,2      (xr2 points to first byte of right)
        b   @uimod
    *   result in ac0+7
```

FUNCTION

@uimod divides the unsigned in ac0 by the unsigned right pointed at by xr2, to obtain the unsigned remainder. No check is made for division by zero, which currently gives a quotient of -1.

RETURNS

The value returned is the unsigned remainder in ac0.

SEE ALSO

@idiv, @uidiv, @imod, @imul

NAME

@uldiv - divide unsigned long by unsigned long

SYNOPSIS

```
        mvc ac0+7(4),left
        la  right-3,2      (xr2 points to first byte of right)
        b   @uldiv
*       result in ac0+7
```

FUNCTION

@uldiv divides the unsigned long in ac0 by the unsigned long right pointed at by xr2, to obtain the unsigned long quotient. No check is made for division by zero, which currently gives a quotient of -1.

RETURNS

The value returned is the unsigned long quotient in ac0.

SEE ALSO

@ldiv, @lmod, @lmul, @ulmod

NAME

@ulmod - remainder unsigned long by unsigned long

SYNOPSIS

```
        mvc ac0+7(4),left
        la  right-3,2      (xr2 points to first byte of right)
        b   @ulmod
    *   result in ac0+7
```

FUNCTION

@ulmod divides the unsigned long in ac0 by the unsigned long right pointed at by xr2, to obtain the unsigned long remainder. No check is made for division by zero, which currently gives an undefined result.

RETURNS

The value returned is the unsigned long remainder in ac0.

SEE ALSO

@uldiv, @ldiv, @lmul, @lmod

NAME

@ulrsh - unsigned long right shift

SYNOPSIS

```
        mvc ac0+7(4),val
        la  count-1,2      (xr2 points to first byte of right)
        b   @ulrsh
    *   result in ac0+7
```

FUNCTION

@ulrsh shifts the unsigned long in **ac0** right by the integer count pointed at by **xr2**, to obtain the unsigned long result.

RETURNS

The value returned is the unsigned long **val>>count** in **ac0**.

SEE ALSO

@llsh, @lrsh

NOTES

Negative shift counts gives the result 0.