

CPU 2.0

Application Programming Interfaces (API) 64-Bit For Win64

Parameters

Args	Param	If lesser units required
Arg1	RCX/XMM0	/ECX/CX/CL
Arg2	RDX/XMM1	/EDX/DX/DL
Arg3	R8	/R8D/R8W/R8B
Arg4	R9	/R9D/R9W/R9B
Arg5	Stack above shadow space	

Return (s)

RAX(1) RDX(2)
XMM0(1) XMM1(2)

If Arg1 is float, Arg2 is integer
=> Arg1 in XMM0, Arg2 in RDX
If Arg1 is integer, Arg2 is float
=> Arg1 in RCX, Arg2 in XMM1

Use MOVQ for double/REAL8 (QWORD)
Use MOVD for single/REAL4 (DWORD)
Pass a pointer to a DT for REAL10

LISTING**

prnreg	prnregdu	prnregd
prnregou	prnrego	dumpreg
dumpregdu	dumpregd	dumpregou
dumpregou	dumpregb	dumpseg
flags**	opsize	opcode
stackview	memview	memviewn
memviewc	memviewb	mem_reset
mem_set	mem_swap	mem_insertr
mem_insert	mem_copy	mem_lcopy
mem_tofile	prntu	prnto
prnhexu	prnhex	prnoct
prnoctu	prnbinu	prnbin
prnbinb	prnt128u	prnoct128u
prnhex128u	fpbin	fpbind
fpbinx		
prndbl	prndble	prnflt
prnflte	prndblx	dblsplit
decodeBASE4	encodeBASE64	int2str
dec2str	dec2stru	oct2str
oct2stru	hex2str	hex2stru
bin2str	bin2stru	dbl2str
dbl2str	flt2str	flte2str
dbl2str	dumpmmxrf	dumpmmxf
dumpmmxr	dumpmmx	dumpxmmrf
dumpxmmf	dumpxmmr	dumpxmm
dumpymmrf	dumpymm	dumpymmr
dumpymm	prndblf	prnfltf
prnhexbf	prntuf	prntf
prntd	prntdf	prntw
prntwf	prntb	prntbf
clearxmm	clearymm	sse_flags
sse_round	rstmxcsr	fpdinfo
fpinfo	fpinfo	fpu_stack
fpu_sflag		
fpu_cflag	fpu_tag	fpu_reg
fpu_copy	fpu_precision	fpu_round
rndint	randm	randq
factorial	add128	mul128

powint	rad2deg	deg2rad
fpow	g_ratio	lg10
ln10	isintREAL4	isintREAL8
iseven	isodd	
bconv	bitfield	str2int
str2dbl	str2flt	dbl2int
dbl2int	flt2int	flt2int
int2dbl	readint	readdbl
readflt	chr_isdigit	chr_isalpha
chr_islower	chr_isupper	chr_change
chr_chcase	chr_tolower	chr_toupper
chr_find	chr_count	chr_shuffle
byte_count	ascii	str_trim
str_wordcnt	str_token	str_copy
str_cmpz	str_cmp	str_find
str_findz	str_append	str_appendz
str_reverse	str_reverse2	str_tolower
str_toupper	str_length	str_lengthd
str_length2	scan_qword	scan_dword
scan_word	scan_byte	
sort_byte	sort_int	sort_intu
sort_dbl	sort_dblx	sortflt
aprnbyte	aprnbyte	aprnbyter
aprnbyteu	aprnbyteu	aprnwordu
aprnwordr	aprnwordr	aprnintd
aprnintdu	aprnintdr	aprnintdur
aprnint	aprnintu	aprnintr
aprnintur	aprnflt	aprnfltr
aprndbl	aprndblr	aprndblx
aprndblxr	aprnbytef	aprnbyterf
aprnbytef	aprnbyteuf	aprnworduf
aprnwordrf	aprnwordrf	aprnintdf
aprnintdf	aprnintdrf	aprnintdurf
aprnintf	aprnintuf	aprnintrf
aprninturf	aprnfltf	aprnfltrf
aprndblf	aprndblrf	

Kernel Specifics

prnchrp**	prnspaces**	prnstreg
prnlines**	prnline	prnspc
prnchr	prnchr	prnchr
prnstrd	prnstrz	prnstr
readchr	readch	readstr
mem_alloc	mem_free	mem_alloc2
mem_free2	mem_load	file_new
file_open	file_read	file_write
file_close	file_size	file_copy
file_delete	timer_start	timer_stop
delay	get_ticks	halt
exitp	exitx	

** Arguments on stack (callee cleanup)

prnreg(1)

Display 64-bit register
Display formatted hex
RCX | Register to display
Ret | -
Note | 16-digit Hex format
| Unsigned

prnregdu(1)

Display unsigned 64-bit Register
RCX | Register to display
Ret | -
Note | 20-digit decimal format

prnregd(1)

Display signed 64-bit Register
RCX | Register to display
Ret | -
Note | 19-digit decimal format

prnregou(1)

Display Register in unsigned octal
RCX | Register to display
Ret | -
Note | -

prnrego(1)

Display Register in signed octal
RCX | Register to display
Ret | -
Note | -

dumpreg

Display registers dump (unsigned hex)
Arg |
Ret | -
Note | -

dumpregdu

Display registers dump (decimal)
arg |
Ret | -
Note | -

dumpregd

Display registers dump (signed int)
arg |
Ret | -
Note | -

dumprego

Display registers dump (signed oct)
arg |
Ret | -
Note | -

dumpregou

Display registers dump (unsigned oct)
Arg |
Ret | -
Note | -

dumpregb

Display registers dump (unsigned binary)
Arg |
Ret | -
Note | -

dumpseg

Display segment registers
Arg | -
Ret | -
Note | -

flags(1)*

Display RFLAG
Arg | pushfq
Ret | -
Note | -

opsize(2)/1

Get size between 2 labels
RDX | Label (reg64,add64)
RCX | Label (reg64,add64)
Ret | Size
Note | Order of label is irrelevant

opcode(2)

Encode instrns between 2 labels
RDX | Label 2
RCX | Label 1 (first/lead label)
Ret | -
Note | Label 1 must be the first

stackview(1)

Display stack
RCX | Number of items to view (+up,-down)
Ret | -
Note | content |hex_address
| 1 item = 8 bytes implied
| If +ve, last line is TOS
| If -ve, First line is TOS

memview(2)

View memory dump + offsets
RDX | size (+up,-down)
RCX | label, address
Ret | -
Note | Leftmost is MSB
| Content |Addr| HexOffs|DecOffs

memviewn(2)

View memory dump in Little-Endian layout
RDX | size (+up,-down)
RCX | label, address
Ret | -
Note | -

memviewc(2)

View memory dump (strings) + offsets
RDX | size (+up,-down)
RCX | label, address
Ret | -
Note | Leftmost is MSB
| Content |Addr| HexOffs|DecOffs

memviewb(3)

View memory dump in flat layout
R8 | Type: 0=Hex, 1=Decimal
RDX | +size
RCX | label, address
Ret | -
Note | -

mem_reset(2)

Clear memory of RDX bytes
RDX | number of bytes to clear
RCX | offset / location
Ret | -
Note | Clear by bytes

mem_set(3)

Set memory with *arg3* byte
R8 | Set byte to use
RDX | size of bytes to set
RCX | offset / pointer
Ret | -
Note| Set by bytes

mem_swap(2)

Swap two memory content
Swap to array elements
RDX | Address of item
RCX | Address of item
Ret | -
Note| Could use LEA / OFFSET

mem_insertr(2)

Insert register content into memory
RDX | Value to insert
RCX | Memory location
Ret | -
Note| -

mem_insert(4)

Insert bytes into memory
R9 | Num of source bytes to write
R8 | Source Address
RDX | Destn offsets to write
RCX | Destn address
Ret | -
Note| Overwrites

mem_copy(3)

Copy memory content of *R8* bytes
R8 | Number of bytes to copy
RDX | Source pointer
RCX | Destination pointer
Ret | -
Note| -

mem_lcopy(3)

Copy memory content of *R8* bytes
R8 | Number of bytes to copy (Large n)
RDX | Source pointer
RCX | Destination pointer
Ret | -
Note| -

mem_tofile(3)

Copy memory content to a new file
R8 | Address of new filename
RDX | Size to copy
RCX | Address of memory source
Ret | A new filled file
Note| Must be a new file or else...

prntu(1)

Display unsigned Decimal
RCX | Value to display
Ret | -
Note| -

prnt(1)

Display Signed Decimal Integer
RCX | Value to display
Ret | -
Note| -

prnhexu(1)

Display Unsigned Hexadecimal
RCX | Value to display
Ret | -
Note| Imm. must be in valid format

prnhex(1)

Display Signed Hexadecimal
RCX | Value to display
Ret | -
Note| Imm must be in valid format

prnoct(1)

Display Octal (signed)
RCX | Value to display
Ret | -
Note| Imm. must be in valid format

prnoctu(1)

Display Octal (unsigned)
RCX | Value to display.
Ret | -
Note| Imm. must be in valid format

prnbinu(1)

Display unsigned Binary
RCX | Value to display
Ret | -
Note| Imm. must be in valid format

prnbin(1)

Display Signed Binary
RCX | Value to display.
Ret | -
Note| Imm. must be in valid format

prnbinb(1)

Display Unsigned Binary with separators
RCX | Value to display
Ret | -
Note| Imm must be in valid format
| Full display

prnt128u(1)

Display Unsigned 128-bit integer
RCX | Addr of 128-bit value to display
Ret | -
Note| Two consecutive QWORDS in memory

prnoct128u(1)

Display Unsigned 128-bit octal integer
RCX | Addr of 128-bit value to display
Ret | -
Note| Two consecutive QWORDS in memory

prnhex128u(1)

Display Unsigned 128-bit Hex
RCX | Addr of 128-bit value to display
Ret | -
Note| Two consecutive QWORDS in memory

fpbin(1)

Display REAL8 Floating Point Binary
XMM0| REAL8 value to display
Ret | -
Note| -

fpbind(1)

Display REAL4 Floating Point Binary
XMM0| REAL4 value to display
Ret | -
Note| -

fpbinx(1)

Display REAL10 Floating Point Binary
RCX | Address of the REAL10
Ret | -
Note| DT type.

prndbl(1)

Display double precision (REAL8)
XMM0| REAL8 value to display
Ret | -
Note| Var should be init as FP (0.0)
| Var must be of type DQ.

prndble(2)

Display REAL8 with decimal places
RDX | Decimal places (1-15)
XMM0| REAL8 value to display
Ret | -
Note| Var should be init as FP (0.0)
| Variable must be of type DQ.

prnflt(1)

Display single precision (REAL4)
XMM0| REAL4 value to display
Ret | -
Note| Var should be initd to FP (0.0)

prnflte(2)

Display single precision (REAL4) with decimal places
RDX | Decimal places (1-7)
XMM0| REAL4 value to display
Ret | -
Note| Var should be initd to FP (0.0)
| XMM0 is of type DD.

prndblx(1)

Display extended precision (REAL10)
RCX | The address of a DT in memory
Ret | -
Note| Var should be initd to FP (0.0)
| Variable must be of type DT.
| Displays unrounded precision

dblsplit(1)/2

Split a REAL8 into parts
XMM0| REAL8 value
Ret | Integral part in RAX/XMM0
| Fraction part in XMM1
Note| Deals normal FP value only

decodeBASE64(2)

Decode 0-ended BASE64 to ASCII
RDX | Address of destination buffer
RCX | Address of string to decode
Ret | -
Note| RDX at least 2/3 the size of RCX
| Returned string are null-ended
| RFC 4648

encodeBASE64(2)

Encode 0-ended ASCII to BASE64 (RFC 4648)
RDX | Address of destination buffer
RCX | Address of string to encode
Ret | -
Note| RDX twice the size of RCX
| Both strings are null-ended

int2str(4)

Convert integer to string
R9 | Signness. 0=unsigned. 1=signed
R8 | Target base (2 to 36 only)
RDX | Buffer's address
RCX | Value
Ret | String in the sent buffer
Note| Buffer size must reflect # of digits
| String will be 0-ended

dec2str(2)

Convert signed int to 0-ended decimal string
RDX | Buffer's address
RCX | Integer to convert
Ret | String in buffer
Note| Buffer must be large enough

dec2stru(2)

Convert unsigned int to 0-ended dec string
RDX | Buffer's address
RCX | Integer to convert
Ret | String in buffer
Note| Buffer must be large enough

oct2str(2)

Convert signed octal to 0-ended string
RDX | Buffer's address
RCX | Integer to convert
Ret | String in buffer
Note| Buffer must be large enough

oct2stru(2)

Convert unsigned octal to 0-ended octal string
RDX | Buffer's address
RCX | Integer to convert
Ret | String in buffer
Note| Buffer must be large enough

hex2str(2)

Convert signed int to 0-ended hex string
RDX | Buffer's address
RCX | Integer to convert
Ret | String in buffer
Note| Buffer must be large enough

hex2stru(2)

Convert unsigned int to 0-ended hex string
RDX | Buffer's address
RCX | Integer to convert
Ret | String in buffer
Note| Buffer must be large enough

bin2str(2)

Convert signed bin to 0-ended bin string
RDX | Buffer's address
RCX | Integer to convert
Ret | String in buffer
Note| Buffer must be large enough (>66)

bin2stru(2)

Convert unsigned bin to 0-ended bin string
RDX | Buffer's address
RCX | Integer to convert
Ret | String in buffer
Note| Buffer must be large enough (>64)

dbl2str(2)

Convert REAL8 to 0-ended string
RDX | Buffer's address to save to
XMM0 | REAL8 value
Ret | 0-ended string in sent buffer.
Note | XMM0 must be of type DQ/REAL8.
| Buffer > 24

double2str(3)

Convert REAL8 with decimal places to 0-ended string
R8 | Buffer's address to save to
RDX | Decimal places (1 to 15)
XMM0 | REAL8 value
Ret | 0-ended string in sent buffer.
Note | XMM0 must be of type DQ/REAL8
| Buffer > 24

flt2str(2)

Convert REAL4 to 0-ended string
RDX | Buffer's address to save to
XMM0 | REAL4 value
Ret | 0-ended string in sent buffer.
Note | XMM0 must be of type DD/REAL4
| Buffer > 15

flte2str(3)

Convert REAL4 with decimal places to 0-ended string
R8 | Buffer's address to save to
RDX | Decimal places. (1-7)
XMM0 | REAL4 value
Ret | 0-ended string in sent buffer.
Note | XMM0 must be of type DD/REAL4
| Buffer > 24

dblx2str(2)

Convert REAL10 to 0-ended string
RDX | Buffer address to save to
RCX | Address of a DT to convert
Ret | 0-ended string in sent buffer.
Note | RCX must point to type DT.
| Buffer >= 32

dumpmmxrf(1)

Dump MMX registers, reversed view.
Formatted.

RCX | Type of display:
0 - Unsigned integer BYTE
1 - Signed integer BYTE
2 - Unsigned integer WORD
3 - Signed integer WORD
4 - Unsigned integer DWORD
5 - Signed integer DWORD
6 - Unsigned integer QWORD
7 - Signed integer QWORD
8 - Unsigned hex BYTES

Note | -

dumpmmxf(1)

Dump MMX registers. Formatted.

RCX | Type of display:
0 - Unsigned integer BYTE
1 - Signed integer BYTE
2 - Unsigned integer WORD
3 - Signed integer WORD
4 - Unsigned integer DWORD
5 - Signed integer DWORD
6 - Unsigned integer QWORD
7 - Signed integer QWORD
8 - Unsigned hex BYTES

Note | -

dumpmmxr(1)

Dump MMX registers, reversed view

RCX | Type of display:
0 - Unsigned integer BYTE
1 - Signed integer BYTE
2 - Unsigned integer WORD
3 - Signed integer WORD
4 - Unsigned integer DWORD
5 - Signed integer DWORD
6 - Unsigned integer QWORD
7 - Signed integer QWORD
8 - Unsigned hex BYTES

Note | -

dumpmmx(1)

Dump MMX registers

RCX | Type of display:
0 - Unsigned integer BYTE
1 - Signed integer BYTE
2 - Unsigned integer WORD
3 - Signed integer WORD
4 - Unsigned integer DWORD
5 - Signed integer DWORD
6 - Unsigned integer QWORD
7 - Signed integer QWORD
8 - Unsigned hex BYTES

Note | -

dumpxmmrf(1)

Dump XMM registers, reversed view.
Formatted.

RCX | Type of display:
0 - Unsigned integer BYTE
1 - Signed integer BYTE
2 - Unsigned integer WORD
3 - Signed integer WORD
4 - Unsigned integer DWORD
5 - Signed integer DWORD
6 - Unsigned integer QWORD
7 - Signed integer QWORD
8 - Singles
9 - Doubles
10 - Unsigned hex BYTES

Note | -

dumpxmmf(1)

Dump XMM registers. Formatted.

RCX | Type of display:
0 - Unsigned integer BYTE
1 - Signed integer BYTE
2 - Unsigned integer WORD
3 - Signed integer WORD
4 - Unsigned integer DWORD
5 - Signed integer DWORD
6 - Unsigned integer QWORD
7 - Signed integer QWORD
8 - Singles
9 - Doubles
10- Unsigned hex BYTES

Note| -

dumpxmmr(1)

Dump XMM registers, reversed view

RCX | Type of display:
0 - Unsigned integer BYTE
1 - Signed integer BYTE
2 - Unsigned integer WORD
3 - Signed integer WORD
4 - Unsigned integer DWORD
5 - Signed integer DWORD
6 - Unsigned integer QWORD
7 - Signed integer QWORD
8 - Singles
9 - Doubles
10- Unsigned hex BYTES

Note| -

dumpxmm(1)

Dump XMM registers

RCX | Type of display:
0 - Unsigned integer BYTE
1 - Signed integer BYTE
2 - Unsigned integer WORD
3 - Signed integer WORD
4 - Unsigned integer DWORD
5 - Signed integer DWORD
6 - Unsigned integer QWORD
7 - Signed integer QWORD
8 - Singles
9 - Doubles
10- Unsigned hex BYTES

Note| -

dumpymmrf(1)

Dump YMM registers, reversed view. Formatted.

RCX | Type of display:
0 - Unsigned integer BYTE
1 - Signed integer BYTE
2 - Unsigned integer WORD
3 - Signed integer WORD
4 - Unsigned integer DWORD
5 - Signed integer DWORD
6 - Unsigned integer QWORD
7 - Signed integer QWORD
8 - Singles
9 - Doubles
10- Unsigned hex BYTES

Note| Expects AVX CPU.

Note| Byte options need wide display

dumpymm(1)

Dump YMM registers. Formatted.

RCX | Type of display:
0 - Unsigned integer BYTE
1 - Signed integer BYTE
2 - Unsigned integer WORD
3 - Signed integer WORD
4 - Unsigned integer DWORD
5 - Signed integer DWORD
6 - Unsigned integer QWORD
7 - Signed integer QWORD
8 - Singles
9 - Doubles
10- Unsigned hex BYTES

Note| Expects AVX CPU

| Byte options need wide display

dumpymmr(1)

Dump YMM registers, reversed view

RCX | Type of display:
0 - Unsigned integer BYTE
1 - Signed integer BYTE
2 - Unsigned integer WORD
3 - Signed integer WORD
4 - Unsigned integer DWORD
5 - Signed integer DWORD
6 - Unsigned integer QWORD
7 - Signed integer QWORD
8 - Singles
9 - Doubles
10- Unsigned hex BYTES

Note| Expects AVX CPU

| Byte options need wide display

dumpymm(1)

Dump YMM registers

RCX | Type of display:
0 - Unsigned integer BYTE
1 - Signed integer BYTE
2 - Unsigned integer WORD
3 - Signed integer WORD
4 - Unsigned integer DWORD
5 - Signed integer DWORD
6 - Unsigned integer QWORD
7 - Signed integer QWORD
8 - Singles
9 - Doubles
10- Unsigned hex BYTES

Note| Expects AVX CPU

| Byte options need wide display

prndblf(1)

Display Double-precision. Formatted

XMM0| Double value to display.

Ret | -

Note| Internal use

prnflt(1)

Display Single-precision. Formatted

XMM0| Single value to display.

Ret | -

Note| Internal use

prnhexbf(1)

Display Unsigned Hex Byte (formatted)

RCX | Byte value to display.

Ret | -

prnintuf(1)

Display Unsigned 64-bit Decimal. Formatted
RCX | Qword Value to display.
Ret | -
Note| Internal use

prnintf(1)

Display Signed 64-bit Decimal. Formatted
RCX | Qword Value to display.
Ret | -
Note| Internal use

prnintd(2)

Display 32-bit Decimal
RDX | 0-unsigned. 1-signed
RCX | Dword Value to display.
Ret | -
Note| Internal use

prnintdf(2)

Display 32-bit Decimal. Formatted
RDX | 0-unsigned. 1-signed
RCX | Dword Value to display.
Ret | -
Note| Internal use

prnintw(2)

Display 16-bit Decimal
RDX | 0-unsigned. 1-signed
RCX | Word value to display.
Ret | -
Note| Internal use

prnintwf(2)

Display 16-bit Decimal (formatted)
RDX | 0-unsigned. 1-signed
RCX | Word value to display
Ret | -
Note| Internal use

prnintb(2)

Display Byte Decimal
RDX | 0-unsigned. 1-signed
RCX | Byte value to display.
Ret | -
Note| Internal use

prnintbf(2)

Display Byte Decimal. Formatted)
RDX | 0-unsigned. 1-signed
RCX | Byte value to display.
Ret | -
Note| Internal use

clearxmm

Clear all XMM registers
Arg | -
Ret | -
Note| -

clearymm

Clear all YMM registers
Arg | -
Ret | -
Note| Expects AVX CPU

sse_flags

Display MXCSR register
Arg | -
Ret | -
Note| -

sse_round(1)

Set SSE rounding control
RCX | Rounding mode
0 = near
1 = down
2 = up
3 = zero
Ret | -
Note| -

rstmxcscr

Reset SIMD Flags register (mxcscr) to CPU default
Arg | -
Ret | -
Note| -

fpdinfo(1)

Display Double FP information
XMM0| REAL8 value
Note| -
Ret | -

fpfinfo(1)

Display Float FP information
XMM0| REAL4 value
Ret | -
Note| -

fpxinfo(1)

Display REAL10 FP information
RCX | Address of a REAL10
Ret | -
Note| -

fpu_stack

Display FPU stack
Arg | -
Ret | -
Note| Display is real10

fpu_sflag

Display FPU Status Flag
Arg | -
Ret | -
Note| -

fpu_cflag

Display FPU Control Flag
Arg | -
Ret | -
Note| -

fpu_tag

Display FPU Tag Register
Arg | -
Ret | -
Note| 00 = valid
| 01 = zero
| 10 = invalid
| 11 = empty

fpu_reg(1)

Display one FPU stack register
RCX | Stack # to display (0-7)
Ret | -
Note| Special values show nothing

fpu_copy(1)/1

Copy a FPU register

RCX | Stack # to copy (0-7)
Ret | Copied value in RAX
Note| -

fpu_precision(1)

Set FPU precision control

RCX | Precision mode
0 = real8
1 = real10
2 = real4

Ret | -

Note| FSAVE/FINIT will reset it

fpu_round(1)

Set FPU rounding control

RCX | Rounding mode
0 = near
1 = down
2 = up
3 = zero

Ret | -

Note| FSAVE/FINIT will reset it

rndint/1

Generate 1 pseudorandom integer

Arg | -
Ret | Random integer in RAX
Note| -

randm(1)/1

Get 1 unsigned random integer

RCX | MAX value (0-MAX)
Ret | Random integer in RAX
Note| 0, MAX inclusive

randq(2)

Generate unique random integers from 0 to MAX-1 and save in array

RDX | Addr of array initialized to -1
RCX | MAX value
Ret | Random elements in array
Note| Example: MAX = 50
elems dq 50 dup(-1)
elems: TIMES 50 dq -1

factorial(1)/1

Get factorial

RCX | +Value
Ret | Factorial in RAX
Note| Limit to 20! Only

add128(3)

Add two 64-bit values and keep result in buffer. Unsigned

R8 | Address of buffer (16 bytes)
RDX | Value
RCX | Value
Ret | -
Note| Use printf128u to display it

mul128(3)

Multiply two 64-bit values and keep result in buffer. Unsigned

R8 | Address of buffer (32 bytes)
RDX | Multiplier
RCX | Multiplicand
Ret | -
Note| Use printf128u to display it

powint(2)/1

Calculate 64-bit integral power

RDX | +Exponent/power (int)
RCX | Base (int)
Ret | Result in RAX
| # if overflow
Note| Var must be of type DQ
| Takes +exponent only

rad2deg(1)/1

Convert Radian to Degree

XMM0| REAL8 value in RADIAN
Ret | DEGREE value in XMM0
Note| Var should be init as FP (0.0)

deg2rad(1)/1

Convert Degree to Radian

XMM0| REAL8 value in DEGREE
Ret | RADIAN value in XMM0
Note| Var should be init as FP (0.0)

fpow(2)/1

Calculate x^n

XMM1| REAL8 power
0.3333333333333333 for cubic root
or 3FD5555555555555h
XMM0| +REAL8 Base
Ret | Raised value in XMM0 (Double)
Note| Args. must be in doubles
| base must be positive

g_ratio(2)/1

Find golden ratio for multi ops

RDX | 0 - mul
1 - div
2 - pow
XMM0| REAL8 value
Ret | golden ratio in XMM0 (double)
Note| 1.618033988749894;848
| XMM0 = XMM0 <op> G.ratio

lg10(1)/1

Find Common Log of Base 10

XMM0| +REAL8 value
Ret | Value in XMM0
Note| -

ln10(1)/1

Find Natural Log

XMM0| +REAL8 value
Ret | Value in XMM0
Note| -

isintREAL4(1)/1

Check if a REAL4 is a qualified integer

XMM0| REAL4 value
Ret | RAX: 0-no,1-yes
Note| Test normal FP values only

isintREAL8(1)/1

Check if a REAL8 is a qualified integer

XMM0| REAL8 value
Ret | RAX: 0-no,1-yes
Note| Test normal FP values only

iseven(1)/1

Check if even number

RCX | Integer to check
Ret | 0 if no. 1 if yes
Note|

isodd(1)/1

Check if odd number

RCX | Integer to check
Ret | RAX - 0 if no. 1 if yes

bconv(2)

Convert and display integer

RDX | Display base to use (2 to 36)
RCX | Integer to convert
Ret | -
Note | Imm value is limited to 32-bit

bitfield(3)

Display a bitfield from a 64-bit data

R8 | lower bit
RDX | higher bit
RCX | Value to be extracted from
Ret | -
Note | Display bit range

str2int(1)/1

Convert 0-ended string to integer

RCX | Address of the 0-ended string
Ret | Integer in RAX
Note | Expected Suffix- b(bin), d(dec)
| h(hex), o,q(octal). 'd' is optional
| String must be of type DB
| Digits string must be valid

str2dbl(1)/1

Convert string to double

RCX | Address of a 0-ended string
Ret | Double-precision XMM0
Note | String must be of type DB

str2flt(1)/1

Convert string to float

RCX | Address of a 0-ended string
Ret | Single-precision in XMM0
Note | Limit digits 7
| String must be of type DB

dbl2int(2)/1

Convert REAL8 to integer

RDX | Rounding mode:
0-near
1-down
2-up
3-zero
XMM0 | REAL8 value
Ret | Integer in RAX
Note | Input must be in FP format

dbl2int(1)/1

Convert REAL8 to integer, truncate to 0

XMM0 | REAL8 value
Ret | Integer in RAX
Note | Input must be in FP format

flt2int(2)/1

Convert REAL4 to integer

RDX | Rounding mode:
0-Near
1-Down
2-Up
3-Zero
XMM0 | REAL4 value
Ret | Integer in EAX
Note | Input must be in FP format

flt2int(1)/1

Convert REAL4 to integer, truncate to 0

XMM0 | REAL4 value
Ret | Integer in EAX
Note | Input must be in FP format

int2dbl(1)/1

Convert integer to real

RCX | Integer value to convert
Ret | Real in XMM0
Note | -

readint/1

Get multi suffix integer from keyboard

Arg | -
Ret | Integer in RAX
Note | Expected Suffix- b(binary), d(dec)
| h(hex), o,q(octal). d is optional
| Digits must be in valid format

readdbl/1

Get double-precision from stdin

Arg | -
Ret | Double precision in XMM0
Note | Doesn't take E format
| Must use FP format for input

readflt/1

Get single-precision from stdin

Arg | -
Ret | Single precision in XMM0
Note | Doesn't take E format
| Must use FP format for input

chr_isdigit(1)/1

Check if alphanumeric digit

RCX | Byte char to check
Ret | RAX - 0 if no. 1 if yes
Note | Digit '0' to '9' only.

chr_isalpha(1)/1

Check if alphabet

RCX | Byte char to check
Ret | RAX - 0 if no. 1 if yes
Note | 'A'-'Z' and 'a'-'z'

chr_islower(1)/1

Check if lowercase

RCX | Byte char to check
Ret | RAX - 0 if no. 1 if yes
Note |

chr_isupper(1)/1

Check if uppercase

RCX | Byte char to check
Ret | RAX - 0 if no. 1 if yes
Note | -

chr_change(3)

Change a char from a 0-ended string

R8 | Char byte to use
RDX | Char byte to change
RCX | Address of string
Ret | -
Note | -

chr_chcase(1)/1

Alternate character case

RCX | Byte char to change
Ret | Modified char
Note | -

chr_tolower(1)/1

Change to lowercase

RCX | Byte to change
Ret | Modified char
Note| -

chr_toupper(1)/1

Change to uppercase

RCX | Character to change
Ret | Modified char
Note| -

chr_find(2)/1

Find byte from a 0-ended string

RDX | Character to find
RCX | String address to look from
Ret | Index in RAX. -1 if not found
Note| Will stop at first one found

chr_count(2)/1

Count a char from a 0-ended string

RDX | Character to count
RCX | String's address
Ret | Count in RAX. 0 if none
Note| -

chr_shuffle(1)

Shuffle a 0-ended string

RCX | String's address
Ret | Same string shuffled
Note| -

byte_count(3)/1

Count byte recurrences

R8 | The byte to count
RDX | Size of source buffer
RCX | Source address
Ret | Count in RAX. 0 if none
Note| -

ascii(1)

Display ASCII equivalences

RCX | Char or Hex value to display
Ret | -
Note| -

str_trim(2)

Trim a 0-ended string

RDX | # of chars to cut
RCX | Address of the 0-ended string
Ret | The same string trimmed
Note| -

str_wordcnt(1)/1

Count words of a 0-ended string

RCX | Address of string
Ret | Word count
Note| Single char is a word

str_token(2)

Display tokens off a 0-ended string

RDX | Addr of 0-ended delimiter string
RCX | Addr of the string
Ret | -
Note| eg, delimiter db '? .-',0

str_copy(3)

Copy a string to another array

R8 | Size of bytes to copy
RDX | Source string's address
RCX | Target buffer's address
Ret | Copied string at the sent address
Note| Target must fit the size to copy
| Target buffer is 0-ended

str_cmpz(2)/1

Compare two 0-ended strings

RDX | String 1 address
RCX | String 2 address
Ret | RAX = 1 if Equal, 0 if not
Note| The two strings must be 0-ended

str_cmp(3)/1

Compare strings with size

R8 | Size of bytes to compare
RDX | String 1 address
RCX | String 2 address
Ret | RAX = 1 if Equal, 0 if not
Note| -

str_find(3)/1

Search for a sub-string

R8 | Size of key string
RDX | Addr of key string
RCX | Addr of 0-ended source string
Ret | Address. -1 -Not found
Note| Case-sensitive

str_findz(2)/1

Search for a sub-string from a C string

RDX | Addr of 0-ended search/key string
RCX | Addr of 0-ended source string
Ret | Location. -1 = Not found
Note| Case-sensitive

str_append(5)/1

Combine two 0-ended strings with size

Arg5| Separator byte. 0-if none
R9 | # of bytes to copy from str2
R8 | Addr of str2
RDX | Addr of str1 (0-ended)
RCX | Addr of buffer
Ret | 0-ended combined string
| RAX - Combined string length
Note| 1st string + 2nd string
| Buffer must be large enough

str_appendz(4)/1

Combine two 0-ended strings

R9 | Separator byte. 0-if none
R8 | Addr of second string
RDX | Addr of first string
RCX | Addr of buffer
Ret | 0-ended combined string
| RAX - Combined string length
Note| 1st string + 2nd string
| Buffer must be large enough

str_reverse(1)

Reverse 0-ended string

RCX | Address of the 0-ended string
Ret | The same string reversed
Note| -

str_reverse2(1)

Reverse 0-ended aligned-16 string
RCX | Address of the 0-ended string
Ret | The same string reversed
Note | String must be align16

str_tolower(1)

Change 0-ended string to lower case
RCX | Address of the 0-ended string
Ret | The same string converted
Note | -

str_toupper(1)

Change 0-ended string to upper case
RCX | Address of the 0-ended string
Ret | The same string converted
Note | -

str_length(1)/1

Find length of a 0-ended string
RCX | String's address
Ret | size in RAX.
Note | -

str_lengthd(2)/1

Get length of a 0-ended string with
delimiter
RDX | Delimiter byte
RCX | String's address
Ret | size in RAX
Note | -

str_length2(1)/1

Find length of a 0-ended string (aligned)
RCX | String's address
Ret | size in RAX.
Note | String must be aligned 16

scan_qword(3)/1

Find first matching quadword
R8 | Size of bytes to scan
RDX | The qword immediate
RCX | Starting location
Ret | Offset. -1 if not found
Note | -

scan_dword(3)/1

Find first matching dword
R8 | Size of bytes to scan
RDX | The dword immediate (DWORD)
RCX | Starting location
Ret | Offset. -1 if not found
Note | -

scan_word(3)/1

Find first matching two-byte
R8 | Size of bytes to scan
RDX | The word immediate (WORD)
RCX | Starting location
Ret | Offset. -1 if not found
Note | -

scan_byte(3)/1

Find first matching byte
R8 | Size of bytes to scan
RDX | The byte immediate (BYTE)
RCX | Starting location
Ret | Offset. -1 if not found
Note | -

sort_byte(3)

Sort char/byte array (signed)
R8 | 0-ascending, 1-descending
RDX | Number of array elements
RCX | Address of array (DB)
Ret | Return the sorted array
Note | Elements are of type DB

sort_int(3)

Sort array of signed int (bubble sort)
R8 | 0-ascending, 1-descending
RDX | Number of array elements
RCX | Address of array (DQ)
Ret | Return the sorted array
Note | Elements are of type DQ

sort_intu(3)

Sort array of unsigned int (bubble sort)
R8 | 0-ascending, 1-descending
RDX | Number of array elements
RCX | Address of array (DQ)
Ret | Return the sorted array
Note | Elements are of type DQ

sort_dbl(3)

Sort array of doubles (bubble sort)
R8 | 0-ascending, 1-descending
RDX | Number of array elements
RCX | Address of array (DQ)
Ret | Return the sorted array
Note | Elements are of type DQ

sort_dblx(3)

Sort array of REAL10 (bubble sort)
R8 | 0-ascending, 1-descending
RDX | Number of array elements
RCX | Address of array (DT)
Ret | Return the sorted array
Note | Elements are of type DT

sortflt(2)

Sort array of singles (bubble sort)
R8 | 0-ascending, 1-descending
RDX | Number of array elements
RCX | Address of array (DD)
Ret | Return the sorted array
Note | Elements are of type DD

aprnbyte(3)

Display array of unsigned hex bytes.
Little-Endian
R8 | Separator byte. 0 if none
RDX | Number of elements
RCX | Address of array
Ret | -
Note | Elements are of type DB

aprnbyter(3)

Display array of unsigned bytes. Reversed-
view
R8 | Separator byte. 0 if none
RDX | Number of elements
RCX | Address of array
Ret | -
Note | Elements are of type DB

aprnbyte(3)

Display array of signed bytes. Little-Endian

R8 | Separator byte. 0 if none
RDX | Number of elements
RCX | Address of array
Ret | -
Note| Elements are of type DB

aprnbyteu(3)

Display array of unsigned bytes. Little-Endian

R8 | Separator byte. 0 if none
RDX | Number of elements
RCX | Address of array
Ret | -
Note| Elements are of type DB

aprnbyter(3)

Display array of signed bytes. Reversed-view

R8 | Separator byte. 0 if none
RDX | Number of elements
RCX | Address of array
Ret | -
Note| Elements are of type DB

aprnbyteur(3)

Display array of unsigned bytes. Reversed-view

R8 | Separator byte. 0 if none
RDX | Number of elements
RCX | Address of array
Ret | -
Note| Elements are of type DB

aprnword(3)

Display array of signed words. Little-Endian

R8 | Separator byte. 0 if none
RDX | Number of elements
RCX | Address of array
Ret | -
Note| Elements are of type DW

aprnwordu(3)

Display array of unsigned words. Little-Endian

R8 | Separator byte. 0 if none
RDX | Number of elements
RCX | Address of array
Ret | -
Note| Elements are of type DW

aprnwordr(3)

Display array of signed words. Reversed-view

R8 | Separator byte. 0 if none
RDX | Number of elements
RCX | Address of array
Ret | -
Note| Elements are of type DW

aprnwordur(3)

Display array of unsigned words. Reversed-view

R8 | Separator byte. 0 if none
RDX | Number of elements
RCX | Address of array
Ret | -
Note| Elements are of type DW

aprintd(3)

Display array of signed ints. Little-Endian

R8 | Separator byte. 0 if none
RDX | Number of elements
RCX | Address of array
Ret | -
Note| Elements are of type DD

aprintdu(3)

Display array of unsigned ints. Little-Endian

R8 | Separator byte. 0 if none
RDX | Number of elements
RCX | Address of array
Ret | -
Note| Elements are of type DD

aprintdr(3)

Display array of signed ints. Reversed-view

R8 | Separator byte. 0 if none
RDX | Number of elements
RCX | Address of array
Ret | -
Note| Elements are of type DD

aprintdur(3)

Display array of unsigned ints. Reversed-view

R8 | Separator byte. 0 if none
RDX | Number of elements
RCX | Address of array
Ret | -
Note| Elements are of type DD

aprint(3)

Display array of signed longs. Little-Endian

R8 | Separator byte. 0 if none
RDX | Number of elements
RCX | Address of array
Ret | -
Note| Elements are of type DQ

aprintu(3)

Display array of unsigned longs. Little-Endian

R8 | Separator byte. 0 if none
RDX | Number of elements
RCX | Address of array
Ret | -
Note| Elements are of type DQ

aprnintr(3)

Display array of signed longs. Reversed view

R8 | Separator byte. 0 if none
RDX | Number of elements
RCX | Address of array
Ret | -
Note| Elements are of type DQ

aprnintur(3)

Display array of unsigned longs. Reversed-view

R8 | Separator byte. 0 if none
RDX | Number of elements
RCX | Address of array
Ret | -
Note| Elements are of type DQ

aprnflt(3)

Display array of REAL4s. Little-Endian

R8 | Separator byte. 0 if none
RDX | Number of elements
RCX | Address of array
Ret | -
Note| Elements are of type DD/REAL4

aprnfltr(3)

Display array of REAL4s. Reversed view

R8 | Separator byte. 0 if none
RDX | Number of elements
RCX | Address of array
Ret | -
Note| Elements are of type DD/REAL4

aprndbl(3)

Display array of REAL8s. Little-Endian

R8 | Separator byte. 0 if none
RDX | Number of elements
RCX | Address of array
Ret | -
Note| Elements are of type DQ/REAL8

aprndblr(3)

Display array of REAL8s. Reversed view

R8 | Separator byte. 0 if none
RDX | Number of elements
RCX | Address of array
Ret | -
Note| Elements are of type DQ/REAL8

aprndblx(3)

Display array of REAL10s. Little-Endian

R8 | Separator byte. 0 if none
RDX | Number of elements
RCX | Address of array
Ret | -
Note| Elements are of type DT/REAL10

aprndblxr(3)

Display array of REAL10s. Reversed view

R8 | Separator byte. 0 if none
RDX | Number of elements
RCX | Address of array
Ret | -
Note| Elements are of type DT/REAL10

aprnbytef(3)

Display array of unsigned hex bytes. Little-Endian. Formatted.

R8 | Separator byte. 0 if none
RDX | Number of elements
RCX | Address of array
Ret | -
Note| Elements are of type DB

aprnbyterf(3)

Display array of unsigned bytes. Reversed-view. Formatted.

R8 | Separator byte. 0 if none
RDX | Number of elements
RCX | Address of array
Ret | -
Note| Elements are of type DB

aprnbytef(3)

Display array of signed bytes. Little-Endian. Formatted.

R8 | Separator byte. 0 if none
RDX | Number of elements
RCX | Address of array
Ret | -
Note| Elements are of type DB

aprnbyteuf(3)

Display array of unsigned bytes. Little-Endian. Formatted.

R8 | Separator byte. 0 if none
RDX | Number of elements
RCX | Address of array
Ret | -
Note| Elements are of type DB

aprnbyterf(3)

Display array of signed bytes. Reversed-view. Formatted.

R8 | Separator byte. 0 if none
RDX | Number of elements
RCX | Address of array
Ret | -
Note| Elements are of type DB

aprnbyteurf(3)

Display array of unsigned bytes. Reversed-view. Formatted.

R8 | Separator byte. 0 if none
RDX | Number of elements
RCX | Address of array
Ret | -
Note| Elements are of type DB

aprnwordf(3)

Display array of signed words. Little-Endian. Formatted.

R8 | Separator byte. 0 if none
RDX | Number of elements
RCX | Address of array
Ret | -
Note| Elements are of type DW

aprnworduf(3)

Display array of unsigned words. Little-Endian. Formatted.

R8 | Separator byte. 0 if none
 RDX | Number of elements
 RCX | Address of array
 Ret | -
 Note| Elements are of type DW

aprnwordrf(3)

Display array of signed words. Reversed-view. Formatted.

R8 | Separator byte. 0 if none
 RDX | Number of elements
 RCX | Address of array
 Ret | -
 Note| Elements are of type DW

aprnwordurf(3)

Display array of unsigned words. Reversed-view. Formatted.

R8 | Separator byte. 0 if none
 RDX | Number of elements
 RCX | Address of array
 Ret | -
 Note| Elements are of type DW

aprnintdf(3)

Display array of signed ints. Little-Endian. Formatted.

R8 | Separator byte. 0 if none
 RDX | Number of elements
 RCX | Address of array
 Ret | -
 Note| Elements are of type DD

aprnintduf(3)

Display array of unsigned ints. Little-Endian. Formatted.

R8 | Separator byte. 0 if none
 RDX | Number of elements
 RCX | Address of array
 Ret | -
 Note| Elements are of type DD

aprnintdrf(3)

Display array of signed ints. Reversed-view. Formatted.

R8 | Separator byte. 0 if none
 RDX | Number of elements
 RCX | Address of array
 Ret | -
 Note| Elements are of type DD

aprnintdurf(3)

Display array of unsigned ints. Reversed-view. Formatted.

R8 | Separator byte. 0 if none
 RDX | Number of elements
 RCX | Address of array
 Ret | -
 Note| Elements are of type DD

aprnintf(3)

Display array of signed longs. Little-Endian. Formatted.

R8 | Separator byte. 0 if none
 RDX | Number of elements
 RCX | Address of array
 Ret | -
 Note| Elements are of type DQ

aprnintuf(3)

Display array of unsigned longs. Little-Endian. Formatted.

R8 | Separator byte. 0 if none
 RDX | Number of elements
 RCX | Address of array
 Ret | -
 Note| Elements are of type DQ

aprnintrf(3)

Display array of signed longs. Reversed-view. Formatted.

R8 | Separator byte. 0 if none
 RDX | Number of elements
 RCX | Address of array
 Ret | -
 Note| Elements are of type DQ

aprninturf(3)

Display array of unsigned longs. Reversed-view. Formatted.

R8 | Separator byte. 0 if none
 RDX | Number of elements
 RCX | Address of array
 Ret | -
 Note| Elements are of type DQ

aprnfltf(3)

Display array of REAL4s. Little-Endian. Formatted.

R8 | Separator byte. 0 if none
 RDX | Number of elements
 RCX | Address of array
 Ret | -
 Note| Elements are of type DD/REAL4

aprnfltrf(3)

Display array of REAL4s. Reversed view. Formatted.

R8 | Separator byte. 0 if none
 RDX | Number of elements
 RCX | Address of array
 Ret | -
 Note| Elements are of type DD/REAL4

aprndblrf(3)

Display array of REAL8s. Little-Endian. Formatted.

R8 | Separator byte. 0 if none
 RDX | Number of elements
 RCX | Address of array
 Ret | -
 Note| Elements are of type DQ/REAL8

aprndblrf(3)

Display array of REAL8s. Reversed view. Formatted.
R8 | Separator byte. 0 if none
RDX | Number of elements
RCX | Address of array
Ret | -
Note| Elements are of type DQ/REAL8

prnchrp(2)*

Display char pattern
push| Push the char to use
push| Push number of times to print
Ret | -
Note| -

prnspaces(1)*

Print multiple whitespace
push| Push number of whitespaces
Ret | -
Note| Arguments on stack

prnstreg(1)

Display short string off register RCX
RCX | The string
Ret | -
Note| -

prnlines(1)*

Print multiple lines
push| Push number of lines
Ret | -
Note| -

prnline

Print a newline
Arg | -
Ret | -
Note| -

prnspace

Print a whitespace
Arg | -
Ret | -
Note| -

prnchr(1)*

Display a char byte from stack
push| Push the char value
Ret | -
Note| Use "qword" for DB variable

prnchar(1)

Display a char byte from memory
RCX | The char's address
Ret | -
Note| Use "qword" for DB var

prnchr(1)

Display a char byte in RCX
RCX | The char value
Ret | -
Note| -

prnstrd(2)

Display a delimiter-ended string
RDX | Delimiter byte value
RCX | The string address
Ret | -
Note| -

prnstrz(1)

Display a 0-ended string
RCX | The string address
Ret | -
Note| -

prnstr(2)

Display string with size
RDX | Size of bytes to display
RCX | The string address
Ret | -
Note| -

readchr(1)

Get a char from keyb and save to memory
RCX | The address to save the char to
Ret | The char in the sent address
Note| Var should be of type DB/rb or resb

readch/1

Get a char from keyb into AL, with echo
Arg | -
Ret | The key in AL
Note| Internal use
| Use halt if needed

readstr(1)/1

Get string from keyboard (echoed)
RCX | Buffer's address to save string
Ret | 0-ended string in sent buffer
| RAX= # of actual bytes in
Note| Press ENTER to finish
| Buffer must be large enough
| Ret -1 signals error
| String will be 0-ended

mem_alloc(1)/1

Request memory of n bytes
RCX | Bytes requested
Ret | Pointer to the mem block. -1 = none
Note| -

mem_free(1)/1

Free memory allocated by mem_alloc
RCX | Pointer from mem_alloc
Ret | 1 = success. -1 = error
Note| -

mem_alloc2(1)/1

Request memory of n bytes
RCX | Bytes requested
Ret | Pointer to the mem block. -1 = none
Note| Uses HeapAlloc

mem_free2(1)/1

Free memory allocated by mem_alloc2
RCX | Pointer from mem_alloc
Ret | 1 = success. -1 = error
Note| Uses HeapFree

mem_load(1)/2

Load a file to memory
RCX | Filename path string
Ret | RAX - block pointer
| RDX - Size
Note| Should re-claim memory at RAX
| File name must be 0-ended

file_new(1)/1

Create a new file

RCX | File name path string
Ret | -1 = Error. 0 = success
Note| Filename string must be 0-ended

file_open(2)/1

Open an existing file for reading/writing

RDX | Operation:
0-Read
1-Write

RCX | File name path string
Ret | File descriptor. -1 if error
Note| Filename string must be 0-ended

file_read(3)/1

Read from an existing opened file

R8 | Number of bytes to read
RDX | Input buffer
RCX | File handle from file_open
Ret | Number of actual read
Note| Filename string must be 0-ended
| Input buffer must be >= RCX

file_write(3)/1

Write to an existing opened file

R8 | Number of bytes to write
RDX | Source buffer's address
RCX | File handle from file_open
Ret | Number of actual writes
Note| Filename string must be 0-ended
| Handle must be opened-write

file_close(1)

Close handle issued by file_open

RCX | File handle
Ret | -
Note| RAX will be zeroed

file_size(1)/1

Get filesize of an opened file

RCX | File handle from file_open
Ret | Size in bytes. -1 says error
Note| -

file_copy(2)

Copy a file to a new file

RDX | Address of new file name
RCX | Address of source file name
Ret | -
Note| Both must be 0-ended string

file_delete(1)

Delete a file

RCX | Address of Filename string
Ret | 0-Fail (Win64), -ve (Linux64)
Note| Handles must be closed

timer_start

Start timer

Arg | -
Ret | R14=freq,R15-timer (silent ret)
Note| Should be followed by timer_stop
| Destroys R14 and R15

timer_stop

Stop current timer

R14 | Freq (ignore if not used)
R15 | Timer (ignore if not used)
Ret | Time difference in seconds. XMM0
Note| Ignore arguments if you are not
using them in-between
| Should come after timer_start
| Destroys R14 and R15

delay(1)

Put execution at delay

RCX | Milliseconds
Ret | -
Note| 1000ms = 1s

get_ticks/1

Get tick count

- | -
Ret | Ticks (ms)
Note| 1000ms = 1s

halt

Pause screen

Arg | -
Ret | -
Note| Hit Enter to continue

exitp

Pause then exit to system

Arg | -
Ret | -
Note| Hit Enter to exit

exitx

Exit to system

Arg | -
Ret | -
Note| -

CPU 2.0

Application Programming Interfaces (API) 64-Bit For Linux64

Parameters

Args	Param	If lesser units required
Arg1	RDI/XMM0	/EDI/DI/DIL
Arg2	RSI/XMM1	/ESI/SI/SIL
Arg3	RDX	/EDX/DX/DL
Arg4	RCX	/ECX/CX/CL
Arg5	R8	/R8D/R8W/R8B

Return (s)

RAX(1) RDX(2)
XMM0(1) XMM1(2)

If Arg1 is float, Arg2 is integer
=> Arg1 in XMM0, Arg2 in RDI

If Arg1 is integer, Arg2 is float
=> Arg1 in RDI, Arg2 in XMM0

Use MOVQ for double/REAL8 (QWORD)

Use MOVD for single/REAL4 (DWORD)

Pass a pointer to a DT for REAL10

LISTING**

prnreg	prnregdu	prnregd
prnregou	prnrego	dumpreg
dumpregdu	dumpregd	dumpregou
dumpregou	dumpregb	dumpseg
flags**	opsize	opcode
stackview	memview	memviewn
memviewc	memviewb	mem_reset
mem_set	mem_swap	mem_insertr
mem_insert	mem_copy	mem_lcopy
mem_tofile	prntu	prnto
prnhexu	prnhex	prnoct
prnoctu	prnbinu	prnbin
prnbinb	prnt128u	prnoct128u
prnhex128u	fpbin	fpbind
fpbinx		
prndbl	prndble	prnflt
prnflte	prndblx	dblsplit
decodeBASE4	encodeBASE64	int2str
dec2str	dec2stru	oct2str
oct2stru	hex2str	hex2stru
bin2str	bin2stru	dbl2str
dbl2str	flt2str	flte2str
dbl2str	dumpmmxrf	dumpmmxf
dumpmmxr	dumpmmx	dumpxmmrf
dumpxmmf	dumpxmmr	dumpxmm
dumpymmrf	dumpymm	dumpymmr
dumpymm	prndblf	prnfltf
prnhexbf	prntuf	prntf
prntd	prntdf	prntw
prntwf	prntb	prntbf
clearxmm	clearymm	sse_flags
sse_round	rstmxcsr	fpdinfo
fpinfo	fpinfo	fpu_stack
fpu_sflag		
fpu_cflag	fpu_tag	fpu_reg
fpu_copy	fpu_precision	fpu_round
rndint	randm	randq
factorial	add128	mul128

powint	rad2deg	deg2rad
fpow	g_ratio	lg10
ln10	isintREAL4	isintREAL8
iseven	isodd	
bconv	bitfield	str2int
str2dbl	str2flt	dbl2int
dbl2int	flt2int	fltt2int
int2dbl	readint	readdbl
readflt	chr_isdigit	chr_isalpha
chr_islower	chr_isupper	chr_change
chr_chcase	chr_tolower	chr_toupper
chr_find	chr_count	chr_shuffle
byte_count	ascii	str_trim
str_wordcnt	str_token	str_copy
str_cmpz	str_cmp	str_find
str_findz	str_append	str_appendz
str_reverse	str_reverse2	str_tolower
str_toupper	str_length	str_lengthd
str_length2	scan_qword	scan_dword
scan_word	scan_byte	
sort_byte	sort_int	sort_intu
sort_dbl	sort_dblx	sortflt
aprnbyte	aprnbyteu	aprnbyter
aprnbyteu	aprnword	aprnwordu
aprnwordr	aprnwordur	aprnintd
aprnintdu	aprnintdr	aprnintdur
aprnint	aprnintu	aprnintr
aprnintur	aprnflt	aprnfltr
aprndbl	aprndblr	aprndblx
aprndblxr	aprnbytef	aprnbyterf
aprnbytef	aprnbyteuf	aprnbyterf
aprnbyteurf	aprnwordf	aprnworduf
aprnwordrf	aprnwordurf	aprnintdf
aprnintduf	aprnintdrf	aprnintdurf
aprnintf	aprnintuf	aprnintrf
aprninturf	aprnfltf	aprnfltrf
aprndblf	aprndblrf	

Kernel Specifics

prnchrp**	prnspaces**	prnstreg
prnlines**	prnline	prnspace
prnchr	prnchr	prnchr
prnstrd	prnstrz	prnstr
readchr	readch	readstr
mem_alloc	mem_free	mem_alloc2
mem_free2	mem_load	file_new
file_open	file_read	file_write
file_close	file_size	file_copy
file_delete	timer_start	timer_stop
delay	get_ticks	halt
exitp	exitx	

** Arguments on stack (callee cleanup)

prnreg(1)

Display 64-bit register
Display formatted hex
RDI | Register to display
Ret | -
Note | 16-digit Hex format
| Unsigned

prnregdu(1)

Display unsigned 64-bit Register
RDI | Register to display
Ret | -
Note | 20-digit decimal format

prnregd(1)

Display signed 64-bit Register
RDI | Register to display
Ret | -
Note | 19-digit decimal format

prnregou(1)

Display Register in unsigned octal
RDI | Register to display
Ret | -
Note | -

prnrego(1)

Display Register in signed octal
RDI | Register to display
Ret | -
Note | -

dumpreg

Display registers dump (unsigned hex)
Arg |
Ret | -
Note | -

dumpregdu

Display registers dump (decimal)
arg |
Ret | -
Note | -

dumpregd

Display registers dump (signed int)
arg |
Ret | -
Note | -

dumprego

Display registers dump (signed oct)
arg |
Ret | -
Note | -

dumpregou

Display registers dump (unsigned oct)
Arg |
Ret | -
Note | -

dumpregb

Display registers dump (unsigned binary)
Arg |
Ret | -
Note | -

dumpseg

Display segment registers
Arg | -
Ret | -
Note | -

flags(1)*

Display RFLAG
Arg | pushfq
Ret | -
Note | -

opsize(2)/1

Get size between 2 labels
RSI | Label (reg64,add64)
RDI | Label (reg64,add64)
Ret | Size
Note | Order of label is irrelevant

opcode(2)

Encode instrns between 2 labels
RSI | Label 2
RDI | Label 1 (first/lead label)
Ret | -
Note | Label 1 must be the first

stackview(1)

Display stack
RDI | Number of items to view (+up,-down)
Ret | -
Note | content |hex_address
| 1 item = 8 bytes implied
| If +ve, last line is TOS
| If -ve, First line is TOS

memview(2)

View memory dump + offsets
RSI | size (+up,-down)
RDI | label, address
Ret | -
Note | Leftmost is MSB
| Content |Addr| HexOffs|DecOffs

memviewn(2)

View memory dump in Little-Endian layout
RSI | size (+up,-down)
RDI | label, address
Ret | -
Note | -

memviewc(2)

View memory dump (strings) + offsets
RSI | size (+up,-down)
RDI | label, address
Ret | -
Note | Leftmost is MSB
| Content |Addr| HexOffs|DecOffs

memviewb(3)

View memory dump in flat layout
RDX | Type: 0=Hex, 1=Decimal
RSI | +size
RDI | label, address
Ret | -
Note | -

mem_reset(2)

Clear memory of RDX bytes
RSI | number of bytes to clear
RDI | offset / location
Ret | -
Note | Clear by bytes

mem_set(3)

Set memory with DL byte
RDX | Set byte to use
RSI | size of bytes to set
RDI | offset / pointer
Ret | -
Note| Set by bytes

mem_swap(2)

Swap two memory content
Swap to array elements
RSI | Address of item
RDI | Address of item
Ret | -
Note| Could use LEA / OFFSET

mem_insertr(2)

Insert register content into memory
RSI | Value to insert
RDI | Memory location
Ret | -
Note| -

mem_insert(4)

Insert bytes into memory
RCX | Num of source bytes to write
RDX | Source Address
RSI | Destn offsets to write
RDI | Destn address
Ret | -
Note| Overwrites

mem_copy(3)

Copy memory content of RDX bytes
RDX | Number of bytes to copy
RSI | Source pointer
RDI | Destination pointer
Ret | -
Note| -

mem_lcopy(3)

Copy memory content of RDX bytes
RDX | Number of bytes to copy (Large n)
RSI | Source pointer
RDI | Destination pointer
Ret | -
Note| -

mem_tofile(3)

Copy memory content to a new file
RDX | Address of new filename
RSI | Size to copy
RDI | Address of memory source
Ret | A new filled file
Note| Must be a new file or else...

prntu(1)

Display unsigned Decimal
RDI | Value to display
Ret | -
Note| -

prnt(1)

Display Signed Decimal Integer
RDI | Value to display
Ret | -
Note| -

prnhexu(1)

Display Unsigned Hexadecimal
RDI | Value to display
Ret | -
Note| Imm. must be in valid format

prnhex(1)

Display Signed Hexadecimal
RDI | Value to display
Ret | -
Note| Imm must be in valid format

prnoct(1)

Display Octal (signed)
RDI | Value to display
Ret | -
Note| Imm. must be in valid format

prnoctu(1)

Display Octal (unsigned)
RDI | Value to display.
Ret | -
Note| Imm. must be in valid format

prnbinu(1)

Display unsigned Binary
RDI | Value to display
Ret | -
Note| Imm. must be in valid format

prnbin(1)

Display Signed Binary
RDI | Value to display.
Ret | -
Note| Imm. must be in valid format

prnbinb(1)

Display Unsigned Binary with separators
RDI | Value to display
Ret | -
Note| Imm must be in valid format
| Full display

prnt128u(1)

Display Unsigned 128-bit integer
RDI | Addr of 128-bit value to display
Ret | -
Note| Two consecutive QWORDS in memory

prnoct128u(1)

Display Unsigned 128-bit octal integer
RDI | Addr of 128-bit value to display
Ret | -
Note| Two consecutive QWORDS in memory

prnhex128u(1)

Display Unsigned 128-bit Hex
RDI | Addr of 128-bit value to display
Ret | -
Note| Two consecutive QWORDS in memory

fpbin(1)

Display REAL8 Floating Point Binary
XMM0| REAL8 value to display
Ret | -
Note| -

fpbind(1)

Display REAL4 Floating Point Binary
XMM0| REAL4 value to display
Ret | -
Note| -

fpbinx(1)

Display REAL10 Floating Point Binary
RDI | Address of the REAL10
Ret | -
Note| DT type

prndbl(1)

Display double precision (REAL8)
XMM0| REAL8 value to display
Ret | -
Note| Var should be init as FP (0.0)
| Var must be of type DQ

prndble(2)

Display REAL8 with decimal places
RDI | Decimal places (1-15)
XMM0| REAL8 value to display
Ret | -
Note| Var should be init as FP (0.0)
| Variable must be of type DQ.

prnflt(1)

Display single precision (REAL4)
XMM0| REAL4 value to display
Ret | -
Note| Var should be initd to FP (0.0)

prnflte(2)

Display single precision (REAL4) with decimal places
RDI | Decimal places (1-7)
XMM0| REAL4 value to display
Ret | -
Note| Var should be initd to FP (0.0)
| XMM0 is of type DD.

prndblx(1)

Display extended precision (REAL10)
RDI | The address of a DT in memory
Ret | -
Note| Var should be initd to FP (0.0)
| Variable must be of type DT.
| Displays unrounded precision

dblsplit(1)/2

Split a REAL8 into parts
XMM0| REAL8 value
Ret | Integral part in RAX/XMM0
| Fraction part in XMM1
Note| Deals normal FP value only

decodeBASE64(2)

Decode 0-ended BASE64 string to ASCII
RSI | Address of destination buffer
RDI | Address of string to decode
Ret | -
Note| RSI at least 2/3 the size of RDI
| Returned string are null-ended
| RFC 4648

encodeBASE64(2)

Encode 0-ended ASCII to BASE64 (RFC 4648)
RSI | Address of destination buffer
RDI | Address of string to encode
Ret | -
Note| RSI twice the size of RDI
| Both strings are null-ended

int2str(4)

Convert integer to string
RCX | Signness. 0=unsigned. 1=signed
RDX | Target base (2 to 36 only)
RSI | Buffer's address
RDI | Value
Ret | String in the sent buffer
Note| Buffer size must reflect # of digits
| String will be 0-ended

dec2str(2)

Convert signed int to 0-ended decimal string
RSI | Buffer's address
RDI | Integer to convert
Ret | String in buffer
Note| Buffer must be large enough

dec2stru(2)

Convert unsigned int to 0-ended dec string
RSI | Buffer's address
RDI | Integer to convert
Ret | String in buffer
Note| Buffer must be large enough

oct2str(2)

Convert signed octal to 0-ended string
RSI | Buffer's address
RDI | Integer to convert
Ret | String in buffer
Note| Buffer must be large enough

oct2stru(2)

Convert unsigned octal to 0-ended octal string
RSI | Buffer's address
RDI | Integer to convert
Ret | String in buffer
Note| Buffer must be large enough

hex2str(2)

Convert signed int to 0-ended hex string
RSI | Buffer's address
RDI | Integer to convert
Ret | String in buffer
Note| Buffer must be large enough

hex2stru(2)

Convert unsigned int to 0-ended hex string
RSI | Buffer's address
RDI | Integer to convert
Ret | String in buffer
Note| Buffer must be large enough

bin2str(2)

Convert signed bin to 0-ended bin string
RSI | Buffer's address
RDI | Integer to convert
Ret | String in buffer
Note| Buffer must be large enough (>66)

bin2stru(2)

Convert unsigned bin to 0-ended bin string
RSI | Buffer's address
RDI | Integer to convert
Ret | String in buffer
Note| Buffer must be large enough (>64)

dbl2str(2)

Convert REAL8 to 0-ended string
RDI | Buffer's address to save to
XMM0 | REAL8 value
Ret | 0-ended string in sent buffer.
Note | XMM0 must be of type DQ/REAL8.
| Buffer > 24

dbl2str(3)

Convert REAL8 with decimal places to 0-ended string
RSI | Buffer's address to save to
RDI | Decimal places (1 to 15)
XMM0 | REAL8 value
Ret | 0-ended string in sent buffer.
Note | XMM0 must be of type DQ/REAL8
| Buffer > 24

flt2str(2)

Convert REAL4 to 0-ended string
RDI | Buffer's address to save to
XMM0 | REAL4 value
Ret | 0-ended string in sent buffer.
Note | XMM0 must be of type DD/REAL4
| Buffer > 15

flte2str(3)

Convert REAL4 with decimal places to 0-ended string
RSI | Buffer's address to save to
RDI | Decimal places. (1-7)
XMM0 | REAL4 value
Ret | 0-ended string in sent buffer.
Note | XMM0 must be of type DD/REAL4
| Buffer > 24

dblx2str(2)

Convert REAL10 to 0-ended string
RSI | Buffer address to save to
RDI | Address of a DT to convert
Ret | 0-ended string in sent buffer.
Note | RDI must point to type DT.
| Buffer >= 32

dumpmmxrf(1)

Dump MMX registers, reversed view.
Formatted.

RDI | Type of display:
0 - Unsigned integer BYTE
1 - Signed integer BYTE
2 - Unsigned integer WORD
3 - Signed integer WORD
4 - Unsigned integer DWORD
5 - Signed integer DWORD
6 - Unsigned integer QWORD
7 - Signed integer QWORD
8 - Unsigned hex BYTES

Note | -

dumpmmxf(1)

Dump MMX registers. Formatted.

RDI | Type of display:
0 - Unsigned integer BYTE
1 - Signed integer BYTE
2 - Unsigned integer WORD
3 - Signed integer WORD
4 - Unsigned integer DWORD
5 - Signed integer DWORD
6 - Unsigned integer QWORD
7 - Signed integer QWORD
8 - Unsigned hex BYTES

Note | -

dumpmmxr(1)

Dump MMX registers, reversed view

RDI | Type of display:
0 - Unsigned integer BYTE
1 - Signed integer BYTE
2 - Unsigned integer WORD
3 - Signed integer WORD
4 - Unsigned integer DWORD
5 - Signed integer DWORD
6 - Unsigned integer QWORD
7 - Signed integer QWORD
8 - Unsigned hex BYTES

Note | -

dumpmmx(1)

Dump MMX registers

RDI | Type of display:
0 - Unsigned integer BYTE
1 - Signed integer BYTE
2 - Unsigned integer WORD
3 - Signed integer WORD
4 - Unsigned integer DWORD
5 - Signed integer DWORD
6 - Unsigned integer QWORD
7 - Signed integer QWORD
8 - Unsigned hex BYTES

Note | -

dumpxmmrf(1)

Dump XMM registers, reversed view.
Formatted.

RDI | Type of display:
0 - Unsigned integer BYTE
1 - Signed integer BYTE
2 - Unsigned integer WORD
3 - Signed integer WORD
4 - Unsigned integer DWORD
5 - Signed integer DWORD
6 - Unsigned integer QWORD
7 - Signed integer QWORD
8 - Singles
9 - Doubles
10 - Unsigned hex BYTES

Note | -

dumpxmmf(1)

Dump XMM registers. Formatted.

RDI | Type of display:
0 - Unsigned integer BYTE
1 - Signed integer BYTE
2 - Unsigned integer WORD
3 - Signed integer WORD
4 - Unsigned integer DWORD
5 - Signed integer DWORD
6 - Unsigned integer QWORD
7 - Signed integer QWORD
8 - Singles
9 - Doubles
10- Unsigned hex BYTES

Note| -

dumpxmmr(1)

Dump XMM registers, reversed view

RDI | Type of display:
0 - Unsigned integer BYTE
1 - Signed integer BYTE
2 - Unsigned integer WORD
3 - Signed integer WORD
4 - Unsigned integer DWORD
5 - Signed integer DWORD
6 - Unsigned integer QWORD
7 - Signed integer QWORD
8 - Singles
9 - Doubles
10- Unsigned hex BYTES

Note| -

dumpxmm(1)

Dump XMM registers

RDI | Type of display:
0 - Unsigned integer BYTE
1 - Signed integer BYTE
2 - Unsigned integer WORD
3 - Signed integer WORD
4 - Unsigned integer DWORD
5 - Signed integer DWORD
6 - Unsigned integer QWORD
7 - Signed integer QWORD
8 - Singles
9 - Doubles
10- Unsigned hex BYTES

Note| -

dumpymmr(1)

Dump YMM registers, reversed view. Formatted.

RDI | Type of display:
0 - Unsigned integer BYTE
1 - Signed integer BYTE
2 - Unsigned integer WORD
3 - Signed integer WORD
4 - Unsigned integer DWORD
5 - Signed integer DWORD
6 - Unsigned integer QWORD
7 - Signed integer QWORD
8 - Singles
9 - Doubles
10- Unsigned hex BYTES

Note| Expects AVX CPU

Note| Byte options need wide display

dumpymmf(1)

Dump YMM registers. Formatted.

RDI | Type of display:
0 - Unsigned integer BYTE
1 - Signed integer BYTE
2 - Unsigned integer WORD
3 - Signed integer WORD
4 - Unsigned integer DWORD
5 - Signed integer DWORD
6 - Unsigned integer QWORD
7 - Signed integer QWORD
8 - Singles
9 - Doubles
10- Unsigned hex BYTES

Note| Expects AVX CPU

Note| Byte options need wide display

dumpymmr(1)

Dump YMM registers, reversed view

RDI | Type of display:
0 - Unsigned integer BYTE
1 - Signed integer BYTE
2 - Unsigned integer WORD
3 - Signed integer WORD
4 - Unsigned integer DWORD
5 - Signed integer DWORD
6 - Unsigned integer QWORD
7 - Signed integer QWORD
8 - Singles
9 - Doubles
10- Unsigned hex BYTES

Note| Expects AVX CPU

Note| Byte options need wide display

dumpymm(1)

Dump YMM registers

RDI | Type of display:
0 - Unsigned integer BYTE
1 - Signed integer BYTE
2 - Unsigned integer WORD
3 - Signed integer WORD
4 - Unsigned integer DWORD
5 - Signed integer DWORD
6 - Unsigned integer QWORD
7 - Signed integer QWORD
8 - Singles
9 - Doubles
10- Unsigned hex BYTES

Note| Expects AVX CPU

Note| Byte option needs wide display

prndblf(1)

Display Double-precision. Formatted

XMM0| Double value to display.

Ret | -

Note| Internal use

prnflt(1)

Display Single-precision. Formatted

XMM0| Single value to display.

Ret | -

Note| Internal use

prnhexbf(1)

Display Unsigned Hex Byte (formatted)

RDI | Byte value to display.

Ret | -

Note| Internal use

printf(1)

Display Unsigned 64-bit Decimal. Formatted
RDI | Qword Value to display.
Ret | -
Note| Internal use

printf(1)

Display Signed 64-bit Decimal. Formatted
RDI | Qword Value to display.
Ret | -
Note| Internal use

printf(2)

Display 32-bit Decimal
RSI | 0-unsigned. 1-signed
RDI | Dword Value to display.
Ret | -
Note| Internal use

printf(2)

Display 32-bit Decimal. Formatted
RSI | 0-unsigned. 1-signed
RDI | Dword Value to display.
Ret | -
Note| Internal use

printf(2)

Display 16-bit Decimal
RSI | 0-unsigned. 1-signed
RDI | Word value to display.
Ret | -
Note| Internal use

printf(2)

Display 16-bit Decimal (Formatted)
RSI | 0-unsigned. 1-signed
RDI | Word value to display
Ret | -
Note| Internal use

printf(2)

Display Byte Decimal
RSI | 0-unsigned. 1-signed
RDI | Byte value to display.
Ret | -
Note| Internal use

printf(2)

Display Byte Decimal. Formatted)
RDX | 0-unsigned. 1-signed
RCX | Byte value to display.
Ret | -
Note| Internal use

clearxmm

Clear all XMM registers
Arg | -
Ret | -
Note| -

clearymm

Clear all YMM registers
Arg | -
Ret | -
Note| Expects AVX CPU

sse_flags

Display MXCSR register
Arg | -
Ret | -
Note| -

sse_round(1)

Set SSE rounding control
RDI | Rounding mode
0 = near
1 = down
2 = up
3 = zero
Ret | -
Note| -

rstmxcsr

Reset SIMD Flags register (mxcsr) to CPU default
Arg | -
Ret | -

fpdinfo(1)

Display Double FP information
XMM0| REAL8 value
Note| -
Ret | -

fpxinfo(1)

Display REAL10 FP information
RDI | Address of a REAL10
Ret | -
Note| -

fpfinfo(1)

Display Float FP information
XMM0| REAL4 value
Ret | -
Note| -

fpu_stack

Display FPU stack
Arg | -
Ret | -
Note| Display is real10

fpu_sflag

Display FPU Status Flag
Arg | -
Ret | -
Note| -

fpu_cflag

Display FPU Control Flag
Arg | -
Ret | -
Note| -

fpu_tag

Display FPU Tag Register
Arg | -
Ret | -
Note| 00 = valid
| 01 = zero
| 10 = invalid
| 11 = empty

fpu_reg(1)

Display one FPU stack register
RDI | Stack # to display (0-7)
Ret | -
Note| Special values show nothing

fpu_copy(1)/1

Copy a FPU register

RDI | Stack # to copy (0-7)
Ret | Copied value in RAX
Note| -

fpu_precision(1)

Set FPU precision control

RDI | Precision mode
0 = real8
1 = real10
2 = real4

Ret | -
Note| FSAVE/FINIT will reset it

fpu_round(1)

Set FPU rounding control

RDI | Rounding mode
0 = near
1 = down
2 = up
3 = zero

Ret | -
Note| FSAVE/FINIT will reset it

rndint/1

Generate 1 pseudorandom integer

Arg | -
Ret | Random integer in RAX
Note| -

randm(1)/1

Get 1 unsigned random integer

RDI | MAX value (0-MAX)
Ret | Random integer in RAX
Note| 0, MAX inclusive

randq(2)

Generate unique random integers

from 0 to MAX-1 and save in array
RSI | Addr of array initialized to -1
RDI | MAX value
Ret | Random elements in array
Note| Example: MAX = 50
elems dq 50 dup(-1)
elems: TIMES 50 dq -1

factorial(1)/1

Get factorial

RDI | +Value
Ret | Factorial in RAX
Note| Limit to 20! Only

add128(3)

Add two 64-bit values and keep result in buffer. Unsigned

RDX | Address of buffer (16 bytes)
RSI | Value
RDI | Value
Ret | -
Note| Use printf128u to display it

mul128(3)

Multiply two 64-bit values and keep result in buffer. Unsigned

RDX | Address of buffer (32 bytes)
RSI | Multiplier
RDI | Multiplicand
Ret | -
Note| Use printf128u to display it

powint(2)/1

Calculate 64-bit integral power

RSI | +Exponent/power (int)
RDI | Base (int)
Ret | Result in RAX
| # if overflow
Note| Var must be of type DQ
| Takes +exponent only

rad2deg(1)/1

Convert Radian to Degree

XMM0| REAL8 value in RADIAN
Ret | DEGREE value in XMM0
Note| Var should be init as FP (0.0)

deg2rad(1)/1

Convert Degree to Radian

XMM0| REAL8 value in DEGREE
Ret | RADIAN value in XMM0
Note| Var should be init as FP (0.0)

fpow(2)/1

Calculate x^n

XMM1| REAL8 power
0.3333333333333333 for cubic root
or 3FD5555555555555h
XMM0| +REAL8 Base
Ret | Raised value in XMM0 (Double)
Note| Args. must be in doubles
| base must be positive

g_ratio(2)/1

Find golden ratio for multi ops

RDI | 0 - mul
1 - div
2 - pow
XMM0| REAL8 value
Ret | golden ratio in XMM0 (double)
Note| 1.618033988749894;848
| XMM0 = XMM0 <op> G.ratio

lg10(1)/1

Find Common Log of Base 10

XMM0| +REAL8 value
Ret | Value in XMM0
Note| -

ln10(1)/1

Find Natural Log

XMM0| +REAL8 value
Ret | Value in XMM0
Note| -

isintREAL4(1)/1

Check if a REAL4 is a qualified integer

XMM0| REAL4 value
Ret | RAX: 0-no,1-yes
Note| Test normal FP values only

isintREAL8(1)/1

Check if a REAL8 is a qualified integer

XMM0| REAL8 value
Ret | RAX: 0-no,1-yes
Note| Test normal FP values only

iseven(1)/1

Check if even number

RDI | Integer to check
Ret | 0 if no. 1 if yes
Note|

isodd(1)/1

Check if odd number

RDI | Integer to check
Ret | RAX - 0 if no. 1 if yes

bconv(2)

Convert and display integer

RSI | Display base to use (2 to 36)
RDI | Integer to convert
Ret | -
Note | Imm value is limited to 32-bit

bitfield(3)

Display a bitfield from a 64-bit data

RDX | lower bit
RSI | higher bit
RDI | Value to be extracted from
Ret | -
Note | Display bit range

str2int(1)/1

Convert 0-ended string to integer

RDI | Address of the 0-ended string
Ret | Integer in RAX
Note | Expected Suffix- b(bin), d(dec)
| h(hex), o,q(octal). 'd' is optional
| String must be of type DB
| Digits string must be valid

str2dbl(1)/1

Convert string to double

RDI | Address of a 0-ended string
Ret | Double-precision XMM0
Note | String must be of type DB

str2flt(1)/1

Convert string to float

RDI | Address of a 0-ended string
Ret | Single-precision in XMM0
Note | Limit digits 7
| String must be of type DB

dbl2int(2)/1

Convert REAL8 to integer

RDI | Rounding mode:
0-near
1-down
2-up
3-zero
XMM0 | REAL8 value
Ret | Integer in RAX
Note | Input must be in FP format

dbl2int(1)/1

Convert REAL8 to integer, truncate to 0

XMM0 | REAL8 value
Ret | Integer in RAX
Note | Input must be in FP format

flt2int(2)/1

Convert REAL4 to integer

RDI | Rounding mode:
0-Near
1-Down
2-Up
3-Zero
XMM0 | REAL4 value
Ret | Integer in EAX
Note | Input must be in FP format

flt2int(1)/1

Convert REAL4 to integer, truncate to 0

XMM0 | REAL4 value
Ret | Integer in EAX
Note | Input must be in FP format

int2dbl(1)/1

Convert integer to real

RDI | Integer value to convert
Ret | Real in XMM0
Note | -

readint/1

Get multi suffix integer from keyboard

Arg | -
Ret | Integer in RAX
Note | Expected Suffix- b(binary), d(dec)
| h(hex), o,q(octal). d is optional
| Digits must be in valid format

readdbl/1

Get double-precision from stdin

Arg | -
Ret | Double precision in XMM0
Note | Doesn't take E format
| Must use FP format for input

readflt/1

Get single-precision from stdin

Arg | -
Ret | Single precision in XMM0
Note | Doesn't take E format
| Must use FP format for input

chr_isdigit(1)/1

Check if alphanumeric digit

RDI | Byte char to check
Ret | RAX - 0 if no. 1 if yes
Note | Digit '0' to '9' only.

chr_isalpha(1)/1

Check if alphabet

RDI | Byte char to check
Ret | RAX - 0 if no. 1 if yes
Note | 'A'-'Z' and 'a'-'z'

chr_islower(1)/1

Check if lowercase

RDI | Byte char to check
Ret | RAX - 0 if no. 1 if yes
Note |

chr_isupper(1)/1

Check if uppercase

RDI | Byte char to check
Ret | RAX - 0 if no. 1 if yes
Note | -

chr_change(3)

Change a char from a 0-ended string

RDX | Char byte to use
RSI | Char byte to change
RDI | Address of string
Ret | -
Note | -

chr_chcase(1)/1

Alternate character case

RDI | Byte char to change
Ret | Modified char
Note | -

chr_tolower(1)/1

Change to lowercase
RDI | Byte to change
Ret | Modified char
Note| -

chr_toupper(1)/1

Change to uppercase
RDI | Character to change
Ret | Modified char
Note| -

chr_find(2)/1

Find byte from a 0-ended string
RSI | Character to find
RDI | String address to look from
Ret | Index in RAX. -1 if not found
Note| Will stop at first one found

chr_count(2)/1

Count a char from a 0-ended string
RSI | Character to count
RDI | String's address
Ret | Count in RAX. 0 if none
Note| -

chr_shuffle(1)

Shuffle a 0-ended string
RDI | String's address
Ret | Same string shuffled
Note| -

byte_count(3)/1

Count byte recurrences
RDX | The byte to count
RSI | Size of source buffer
RDI | Source address
Ret | Count in RAX. 0 if none
Note| -

ascii(1)

Display ASCII equivalences
RDI | Char or Hex value to display
Ret | -
Note| -

str_trim(2)

Trim a 0-ended string
RSI | # of chars to cut
RDI | Address of the 0-ended string
Ret | The same string trimmed
Note| -

str_wordcnt(1)/1

Count words of a 0-ended string
RDI | Address of string
Ret | Word count
Note| Single char is a word

str_token(2)

Display tokens off a 0-ended string
RSI | Addr of 0-ended delimiter string
RDI | Addr of the string
Ret | -
Note| eg, delimiter db '? .-',0

str_copy(3)

Copy a string to another array
RDX | Size of bytes to copy
RSI | Source string's address
RDI | Target buffer's address
Ret | Copied string at the sent address
Note| Target must fit the size to copy
| Target buffer is 0-ended

str_cmpz(2)/1

Compare two 0-ended strings
RSI | String 1 address
RDI | String 2 address
Ret | RAX = 1 if Equal, 0 if not
Note| The two strings must be 0-ended

str_cmp(3)/1

Compare strings with size
RDX | Size of bytes to compare
RSI | String 1 address
RDI | String 2 address
Ret | RAX = 1 if Equal, 0 if not
Note| -

str_find(3)/1

Search for a sub-string
RDX | Size of key string
RSI | Addr of key string
RDI | Addr of 0-ended source string
Ret | Address. -1 -Not found
Note| Case-sensitive

str_findz(2)/1

Search for a sub-string from a C string
RSI | Addr of 0-ended search/key string
RDI | Addr of 0-ended source string
Ret | Location. -1 = Not found
Note| Case-sensitive

str_append(5)/1

Combine two 0-ended strings with size
R8 | Separator byte. 0-if none
RCX | # of bytes to copy from str2
RDX | Addr of str2
RSI | Addr of str1 (0-ended)
RDI | Addr of buffer
Ret | 0-ended combined string
| RAX - Combined string length
Note| 1st string + 2nd string
| Buffer must be large enough

str_appendz(4)/1

Combine two 0-ended strings
RCX | Separator byte. 0-if none
RDX | Addr of second string
RSI | Addr of first string
RDI | Addr of buffer
Ret | 0-ended combined string
| RAX - Combined string length
Note| 1st string + 2nd string
| Buffer must be large enough

str_reverse(1)

Reverse 0-ended string
RDI | Address of the 0-ended string
Ret | The same string reversed
Note| -

str_reverse2(1)

Reverse 0-ended aligned string
RDI | Address of the 0-ended string
Ret | The same string reversed
Note | String must be align16

str_tolower(1)

Change 0-ended string to lower case
RDI | Address of the 0-ended string
Ret | The same string converted
Note | -

str_toupper(1)

Change 0-ended string to upper case
RDI | Address of the 0-ended string
Ret | The same string converted
Note | -

str_length(1)/1

Find length of a 0-ended string
RDI | String's address
Ret | size in RAX.
Note | -

str_lengthd(2)/1

Get length of a 0-ended string with
delimiter
RSI | Delimiter byte
RDI | String's address
Ret | size in RAX
Note | -

str_length2(1)/1

Find length of a 0-ended string (aligned)
RDI | String's address
Ret | size in RAX.
Note | String must be aligned 16

scan_qword(3)/1

Find first matching quadword
RDX | Size of bytes to scan
RSI | The qword immediate
RDI | Starting location
Ret | Offset. -1 if not found
Note | -

scan_dword(3)/1

Find first matching dword
RDX | Size of bytes to scan
RSI | The dword immediate (DWORD)
RDI | Starting location
Ret | Offset. -1 if not found
Note | -

scan_word(3)/1

Find first matching two-byte
RDX | Size of bytes to scan
RSI | The word immediate (WORD)
RDI | Starting location
Ret | Offset. -1 if not found
Note | -

scan_byte(3)/1

Find first matching byte
RDX | Size of bytes to scan
RSI | The byte immediate (BYTE)
RDI | Starting location
Ret | Offset. -1 if not found
Note | -

sort_byte(3)

Sort char/byte array (signed)
RDX | 0-ascending, 1-descending
RSI | Number of array elements
RDI | Address of array (DB)
Ret | Return the sorted array
Note | Elements are of type DB

sort_int(3)

Sort array of signed int (bubble sort)
RDX | 0-ascending, 1-descending
RSI | Number of array elements
RDI | Address of array (DQ)
Ret | Return the sorted array
Note | Elements are of type DQ

sort_intu(3)

Sort array of unsigned int (bubble sort)
RDX | 0-ascending, 1-descending
RSI | Number of array elements
RDI | Address of array (DQ)
Ret | Return the sorted array
Note | Elements are of type DQ

sort_dbl(3)

Sort array of doubles (bubble sort)
RDX | 0-ascending, 1-descending
RSI | Number of array elements
RDI | Address of array (DQ)
Ret | Return the sorted array
Note | Elements are of type DQ

sort_dblx(3)

Sort array of REAL10 (bubble sort)
RDX | 0-ascending, 1-descending
RSI | Number of array elements
RDI | Address of array (DT)
Ret | Return the sorted array
Note | Elements are of type DT

sort_fit(2)

Sort array of singles (bubble sort)
RDX | 0-ascending, 1-descending
RSI | Number of array elements
RDI | Address of array (DD)
Ret | Return the sorted array
Note | Elements are of type DD

aprnbyte(3)

Display array of unsigned hex bytes.
Little-Endian
DL | Separator byte. 0 if none
RSI | Number of elements
RDI | Address of array
Ret | -
Note | Elements are of type DB

aprnbyter(3)

Display array of unsigned bytes. Reversed-
view
DL | Separator byte. 0 if none
RSI | Number of elements
RDI | Address of array
Ret | -
Note | Elements are of type DB

aprnbyte(3)

Display array of signed bytes. Little-Endian

DL | Separator byte. 0 if none
RSI | Number of elements
RDI | Address of array
Ret | -
Note| Elements are of type DB

aprnbyteu(3)

Display array of unsigned bytes. Little-Endian

DL | Separator byte. 0 if none
RSI | Number of elements
RDI | Address of array
Ret | -
Note| Elements are of type DB

aprnbyter(3)

Display array of signed bytes. Reversed-view

DL | Separator byte. 0 if none
RSI | Number of elements
RDI | Address of array
Ret | -
Note| Elements are of type DB

aprnbyteur(3)

Display array of unsigned bytes. Reversed-view

DL | Separator byte. 0 if none
RSI | Number of elements
RDI | Address of array
Ret | -
Note| Elements are of type DB

aprnword(3)

Display array of signed words. Little-Endian

DL | Separator byte. 0 if none
RSI | Number of elements
RDI | Address of array
Ret | -
Note| Elements are of type DW

aprnwordu(3)

Display array of unsigned words. Little-Endian

DL | Separator byte. 0 if none
RSI | Number of elements
RDI | Address of array
Ret | -
Note| Elements are of type DW

aprnwordr(3)

Display array of signed words. Reversed-view

DL | Separator byte. 0 if none
RSI | Number of elements
RDI | Address of array
Ret | -
Note| Elements are of type DW

aprnwordur(3)

Display array of unsigned words. Reversed-view

DL | Separator byte. 0 if none
RSI | Number of elements
RDI | Address of array
Ret | -
Note| Elements are of type DW

aprintd(3)

Display array of signed ints. Little-Endian

DL | Separator byte. 0 if none
RSI | Number of elements
RDI | Address of array
Ret | -
Note| Elements are of type DD

aprintdu(3)

Display array of unsigned ints. Little-Endian

DL | Separator byte. 0 if none
RSI | Number of elements
RDI | Address of array
Ret | -
Note| Elements are of type DD

aprintdr(3)

Display array of signed ints. Reversed-view

DL | Separator byte. 0 if none
RSI | Number of elements
RDI | Address of array
Ret | -
Note| Elements are of type DD

aprintdur(3)

Display array of unsigned ints. Reversed-view

DL | Separator byte. 0 if none
RSI | Number of elements
RDI | Address of array
Ret | -
Note| Elements are of type DD

aprint(3)

Display array of signed longs. Little-Endian

DL | Separator byte. 0 if none
RSI | Number of elements
RDI | Address of array
Ret | -
Note| Elements are of type DQ

aprintu(3)

Display array of unsigned longs. Little-Endian

DL | Separator byte. 0 if none
RSI | Number of elements
RDI | Address of array
Ret | -
Note| Elements are of type DQ

aprnintr(3)

Display array of signed longs. Reversed view

DL | Separator byte. 0 if none
RSI | Number of elements
RDI | Address of array
Ret | -
Note| Elements are of type DQ

aprnintur(3)

Display array of unsigned longs. Reversed-view

DL | Separator byte. 0 if none
RSI | Number of elements
RDI | Address of array
Ret | -
Note| Elements are of type DQ

aprnflt(3)

Display array of REAL4s. Little-Endian

DL | Separator byte. 0 if none
RSI | Number of elements
RDI | Address of array
Ret | -
Note| Elements are of type DD/REAL4

aprnfltr(3)

Display array of REAL4s. Reversed view

DL | Separator byte. 0 if none
RSI | Number of elements
RDI | Address of array
Ret | -
Note| Elements are of type DD/REAL4

aprndbl(3)

Display array of REAL8s. Little-Endian

DL | Separator byte. 0 if none
RSI | Number of elements
RDI | Address of array
Ret | -
Note| Elements are of type DQ/REAL8

aprndblr(3)

Display array of REAL8s. Reversed view

DL | Separator byte. 0 if none
RSI | Number of elements
RDI | Address of array
Ret | -
Note| Elements are of type DQ/REAL8

aprndblx(3)

Display array of REAL10s. Little-Endian

DL | Separator byte. 0 if none
RSI | Number of elements
RDI | Address of array
Ret | -
Note| Elements are of type DT/REAL10

aprndblxr(3)

Display array of REAL10s. Reversed view

DL | Separator byte. 0 if none
RSI | Number of elements
RDI | Address of array
Ret | -
Note| Elements are of type DT/REAL10

aprnbytef(3)

Display array of unsigned hex bytes. Little-Endian. Formatted.

DL | Separator byte. 0 if none
RSI | Number of elements
RDI | Address of array
Ret | -
Note| Elements are of type DB

aprnbyterf(3)

Display array of unsigned bytes. Reversed-view. Formatted.

DL | Separator byte. 0 if none
RSI | Number of elements
RDI | Address of array
Ret | -
Note| Elements are of type DB

aprnbytef(3)

Display array of signed bytes. Little-Endian. Formatted.

DL | Separator byte. 0 if none
RSI | Number of elements
RDI | Address of array
Ret | -
Note| Elements are of type DB

aprnbyteuf(3)

Display array of unsigned bytes. Little-Endian. Formatted.

DL | Separator byte. 0 if none
RSI | Number of elements
RDI | Address of array
Ret | -
Note| Elements are of type DB

aprnbyterf(3)

Display array of signed bytes. Reversed-view. Formatted.

DL | Separator byte. 0 if none
RSI | Number of elements
RDI | Address of array
Ret | -
Note| Elements are of type DB

aprnbyteurf(3)

Display array of unsigned bytes. Reversed-view. Formatted.

DL | Separator byte. 0 if none
RSI | Number of elements
RDI | Address of array
Ret | -
Note| Elements are of type DB

aprnwordf(3)

Display array of signed words. Little-Endian. Formatted.

DL | Separator byte. 0 if none
RSI | Number of elements
RDI | Address of array
Ret | -
Note| Elements are of type DW

aprnworduf(3)

Display array of unsigned words. Little-Endian. Formatted.

DL | Separator byte. 0 if none
 RSI | Number of elements
 RDI | Address of array
 Ret | -
 Note| Elements are of type DW

aprnwordrf(3)

Display array of signed words. Reversed-view. Formatted.

DL | Separator byte. 0 if none
 RSI | Number of elements
 RDI | Address of array
 Ret | -
 Note| Elements are of type DW

aprnwordurf(3)

Display array of unsigned words. Reversed-view. Formatted.

DL | Separator byte. 0 if none
 RSI | Number of elements
 RDI | Address of array
 Ret | -
 Note| Elements are of type DW

aprnintdf(3)

Display array of signed ints. Little-Endian. Formatted.

DL | Separator byte. 0 if none
 RSI | Number of elements
 RDI | Address of array
 Ret | -
 Note| Elements are of type DD

aprnintduf(3)

Display array of unsigned ints. Little-Endian. Formatted.

DL | Separator byte. 0 if none
 RSI | Number of elements
 RDI | Address of array
 Ret | -
 Note| Elements are of type DD

aprnintdrf(3)

Display array of signed ints. Reversed-view. Formatted.

DL | Separator byte. 0 if none
 RSI | Number of elements
 RDI | Address of array
 Ret | -
 Note| Elements are of type DD

aprnintdurf(3)

Display array of unsigned ints. Reversed-view. Formatted.

DL | Separator byte. 0 if none
 RSI | Number of elements
 RDI | Address of array
 Ret | -
 Note| Elements are of type DD

aprnltf(3)

Display array of signed longs. Little-Endian. Formatted.

DL | Separator byte. 0 if none
 RSI | Number of elements
 RDI | Address of array
 Ret | -
 Note| Elements are of type DQ

aprnltuf(3)

Display array of unsigned longs. Little-Endian. Formatted.

DL | Separator byte. 0 if none
 RSI | Number of elements
 RDI | Address of array
 Ret | -
 Note| Elements are of type DQ

aprnltrf(3)

Display array of signed longs. Reversed-view. Formatted.

DL | Separator byte. 0 if none
 RSI | Number of elements
 RDI | Address of array
 Ret | -
 Note| Elements are of type DQ

aprnlturf(3)

Display array of unsigned longs. Reversed-view. Formatted.

DL | Separator byte. 0 if none
 RSI | Number of elements
 RDI | Address of array
 Ret | -
 Note| Elements are of type DQ

aprnfltf(3)

Display array of REAL4s. Little-Endian. Formatted.

DL | Separator byte. 0 if none
 RSI | Number of elements
 RDI | Address of array
 Ret | -
 Note| Elements are of type DD/REAL4

aprnfltrf(3)

Display array of REAL4s. Reversed view. Formatted.

DL | Separator byte. 0 if none
 RSI | Number of elements
 RDI | Address of array
 Ret | -
 Note| Elements are of type DD/REAL4

aprndbluf(3)

Display array of REAL8s. Little-Endian. Formatted.

DL | Separator byte. 0 if none
 RSI | Number of elements
 RDI | Address of array
 Ret | -
 Note| Elements are of type DQ/REAL8

aprndblrf(3)

Display array of REAL8s. Reversed view. Formatted.

DL | Separator byte. 0 if none
RSI | Number of elements
RDI | Address of array
Ret | -
Note| Elements are of type DQ/REAL8

prnchrp(2)*

Display char pattern
push| Push the char to use
push| Push number of times to print
Ret | -
Note| -

prnspaces(1)*

Print multiple whitespace
push| Push number of whitespaces
Ret | -
Note| Arguments on stack

prnstreg(1)

Display short string off register RDI
RDI | The string
Ret | -
Note| -

prnlines(1)*

Print multiple lines
push| Push number of lines
Ret | -
Note| -

prnline

Print a newline
Arg | -
Ret | -
Note| -

prnspace

Print a whitespace
Arg | -
Ret | -
Note| -

prnchrs(1)*

Display a char byte from stack
push| Push the char value
Ret | -
Note| Use "qword" for DB variable

prnchar(1)

Display a char byte from memory
RDI | The char's address
Ret | -
Note| Use "qword" for DB var

prnchr(1)

Display a char byte in RDI
RDI | The char value
Ret | -
Note| -

prnstrd(2)

Display a delimiter-ended string
RSI | Delimiter byte value
RDI | The string address
Ret | -
Note| -

prnstrz(1)

Display a 0-ended string
RDI | The string address
Ret | -
Note| -

prnstr(2)

Display string with size
RSI | Size of bytes to display
RDI | The string address
Ret | -
Note| -

readchr(1)

Get a char from keyb and save to memory
RDI | The address to save the char to
Ret | The char in the sent address
Note| Var should be of type DB/rb or resb

readch/1

Get a char from keyb into AL, with echo
Arg | -
Ret | The key in AL
Note| Internal use
| Use halt if needed

readstr(1)/1

Get string from keyboard (echoed)
RDI | Buffer's address to save string
Ret | 0-ended string in sent buffer
| RAX= # of actual bytes in
Note| Press ENTER to finish
| Buffer must be large enough
| Ret -1 signals error
| String will be 0-ended

mem_alloc(1)/1

Request memory of n bytes
RDI | Bytes requested
Ret | Pointer to the mem block. -1 = none
Note| -

mem_free(1)/1

Free memory allocated by mem_alloc
RDI | Pointer from mem_alloc
Ret | 1 = success. -1 = error
Note| -

mem_alloc2(1)/1

Request memory of n bytes
RDI | Bytes requested
Ret | Pointer to the mem block. -1 = none
Note| Uses HeapAlloc

mem_free2(2)/1

Free memory allocated by mem_alloc2
RSI | Size to free
RDI | Pointer from mem_alloc
Ret | 1 = success. -1 = error
Note| Uses HeapFree

mem_load(1)/2

Load a file to memory
RDI | Filename path string
Ret | RAX - block pointer
| RDX - Size
Note| Should re-claim memory at RAX
| File name must be 0-ended

file_new(1)/1

Create a new file
RDI | File name path string
Ret | -1 = Error. 0 = success
Note| Filename string must be 0-ended

file_open(2)/1

Open an existing file for reading/writing
RSI | Operation:
 0-Read
 1-Write

RDI | File name path string
Ret | File descriptor. -1 if error
Note| Filename string must be 0-ended

file_read(3)/1

Read from an existing opened file
RDX | Number of bytes to read
RSI | Input buffer
RDI | File handle from file_open
Ret | Number of actual read
Note| Filename string must be 0-ended
 | Input buffer must be >= RDI

file_write(3)/1

Write to an existing opened file
RDX | Number of bytes to write
RSI | Source buffer's address
RDI | File handle from file_open
Ret | Number of actual writes
Note| Filename string must be 0-ended
 | Handle must be opened-write

file_close(1)

Close handle issued by file_open
RDI | File handle
Ret | -
Note| RAX will be zeroed

file_size(1)/1

Get filesize of an opened file
RDI | File handle from file_open
Ret | Size in bytes. -1 says error
Note| -

file_copy(2)

Copy a file to a new file
RSI | Address of new file name
RDI | Address of source file name
Ret | -
Note| Both must be 0-ended string

file_delete(1)

Delete a file
RDI | Address of Filename string
Ret | 0-Fail (Win64), -ve (Linux64)
Note| Handles must be closed

timer_start

Start timer
Arg | -
Ret | R14=freq,R15-timer (silent ret)
Note| Should be followed by timer_stop
 | Destroys R14 and R15

timer_stop

Stop current timer
R14 | Freq (ignore if not used)
R15 | Timer (ignore if not used)
Ret | Time difference in seconds. XMM0
Note| Ignore arguments if you are not
 | using them in-between
 | Should come after timer_start
 | Destroys R14 and R15

delay(1)

Put execution at delay
RDI | Milliseconds
Ret | -
Note| 1000ms = 1s

get_ticks/1

Get tick count
- | -
Ret | Ticks (ms)
Note| 1000ms = 1s

halt

Pause screen
Arg | -
Ret | -
Note| Hit Enter to continue

exitp

Pause then exit to system
Arg | -
Ret | -
Note| Hit Enter to exit

exitx

Exit to system
Arg | -
Ret | -
Note| -

CPU 2.0 (cpu86)

32-bit Application Programming Interfaces (API) for cpu86

This API applies to both Win32 and Linux32 versions of cpu86. cpu86 uses C calling convention (CDECL)*:

- Parameters use stack memory
- Parameter size defaults to four bytes (dword)
- Arguments are passed / pushed on the stack; in correct sequence, types and numbers
- Double-precision and quadwords arguments are passed by values as two consecutive dwords. Larger arguments are passed by reference.
- Except for float-specific (real4) routines, all real arguments are passed as doubles on the stack
- Integer returns in EAX. Real returns in ST0 (FPU registers are clobbered)
- Quadword values are returned in EAX(low) and EDX(high).
- Caller unwind / clean-up stack

LISTING**

prnreg	prnregdu	prnregd
prnregou	prnrego	dumpreg
dumpregdu	dumpregd	dumprego
dumpregou	dumpregb	dumpseg
flags	opsize	opcode
stackview	memview	memviewn
memviewc	memviewb	mem_reset
mem_set	mem_swap	mem_insertr
mem_insert	mem_copy	mem_lcopy
mem_tofile	prnintu	prnint
prnhexu	prnhex	prnoct
prnoctu	prnbinu	prnbin
prnbinb	fpbin	fpbind
fpbinx		
prnintq	prnintqu	prnhexq
prnhexqu	prnoctq	prnoctqu
prndbl	prndble	prnflt
prnflte	prndblx	fltsplit
decodeBASE4	encodeBASE64	int2str
dec2str	dec2stru	oct2str
oct2stru	hex2str	hex2stru
bin2str	bin2stru	dbl2str
dbl2str	flt2str	flte2str
dbl2str	dumpmmxrf	dumpmmxf
dumpmmxr	dumpmmx	dumpxmmf
dumpxmmf	dumpxmmr	dumpxmm
dumpymmrf	dumpymm	dumpymmr
dumpymm	prndblf	prnfltf
prnhexbf	prnintuf	prnintf
prnintqf	prnintquf	prnintbf
prnintb	prnintwf	prnintw
clearxmm	clearymm	sse_flags
sse_round	rstmxcsr	fpdinfo

fpfinfo	fpxinfo	fpu_stack
fpu_sflag	fpu_cflag	fpu_tag
fpu_reg	fpu_precision	fpu_round
rndint	randm	randq
factorial	powint	rad2deg
deg2rad	fpow	g_ratio
lg10	ln10	isintREAL4
isintREAL8	iseven	isodd
bconv	bitfield	str2int
str2dbl	str2flt	dbl2int
dbl2int	flt2int	flt2int
int2flt	readint	readdbl
readflt	chr_isdigit	chr_isalpha
chr_islower	chr_isupper	chr_change
chr_chcase	chr_tolower	chr_toupper
chr_find	chr_count	chr_shuffle
byte_count	ascii	str_trim
str_wordcnt	str_token	str_copy
str_cmpz	str_cmp	str_find
str_findz	str_append	str_appendz
str_reverse	str_reverse2	str_tolower
str_toupper	str_length	str_lengthd
str_length2	scan_dword	scan_word
scan_byte	sort_byte	sort_int
sort_intu	sort_dbl	sort_dblx
sort_flt	aprnbyte	aprnbyter
aprnbyte	aprnbyteu	aprnbyter
aprnbyteur	aprnword	aprnwordu
aprnwordr	aprnwordr	aprnwordu
aprnintdu	aprnintdr	aprnintdur
aprnint	aprnintu	aprnintr
aprnintur	aprnflt	aprnfltr
aprndbl	aprndblr	aprndblx
aprndblxr	aprnbytef	aprnbyterf
aprnbytef	aprnbyteuf	aprnbyterf
aprnwordf	aprnwordrf	aprnworduf
aprnwordrf	aprnwordrf	aprnintdf
aprnintduf	aprnintdrf	aprnintdurf
aprnintf	aprnintuf	aprnintrf
aprninturf	aprnfltf	aprnfltrf
aprndblf	aprndblrf	

Kernel Specifics

prnchrp	prnspaces	prnstreg***
prnlines	prnline	prnspc
prnchr	prnchr	prnstrd
prnstrz	prnstr	readchr
readch	readstr	mem_alloc
mem_free	mem_load	file_new
file_open	file_read	file_write
file_close	file_size	file_copy
file_delete	timer_start	timer_stop
delay	get_ticks	halt
exitp	exitx	

* cpu86 does not use C library. Only uses CDECL as calling convention
 ** A number of routines are not publicly visible due to their supportive nature.
 *** Argument in EAX

prnreg(1)

Display 32-bit register in unsigned hex
push1| Register to display
Ret | -
Note | -

prnregdu(1)

Display 32-bit register in unsigned int
push1| Register to display
Ret | -
Note | -

prnregd(1)

Display 32-bit register in signed int
push1| Register to display
Ret | -
Note | -

prnregou(1)

Display 32-bit register in unsigned octal
push1| Register to display
Ret | -
Note | -

prnrego(1)

Display 32-bit register in signed octal
push1| Register to display
Ret | -
Note | -

dumpreg

Display registers dump (unsigned hex)
Arg |
Ret | -
Note | -

dumpregdu

Display registers dump (decimal)
Arg |
Ret | -
Note | -

dumpregd

Display registers dump (signed int)
Arg |
Ret | -
Note | -

dumpregou

Display registers dump (unsigned oct)
Arg |
Ret | -
Note | -

dumprego

Display registers dump (signed oct)
Arg |
Ret | -
Note | -

dumpregb

Display registers dump (unsigned binary)
Arg |
Ret | -
Note | -

dumpseg

Display segment registers
Arg | -
Ret | -
Note | -

flags(1)

Display EFLAG register
push1| pushfd
Ret | -
Note | -

opsize(2)/1

Get size between 2 labels
push2| Label
push1| Label
Ret | Size
Note | Order of label is irrelevant

opcode(2)

Encode instrns between 2 labels
push2| Label 2
push1| Label 1 (first/lead label)
Ret | -
Note | Label 1 must be the first

stackview(1)

Display stack
push1| Number of items to view (+up,-down)
Ret | -
Note | content |hex_address
| 1 item = 4 bytes implied
| If +ve, last line is TOS
| If -ve, First line is TOS

memview(2)

View memory dump + offsets
push2| label, address
push1| size (+up,-down)
Ret | -
Note | Leftmost is MSB
| Content |Addr| HexOffs|DecOffs

memviewn(2)

View memory dump in Little-Endian layout
push2| label, address
push1| size (+up,-down)
Ret | -
Note | -

memviewc(2)

View memory dump (strings) + offsets
push2| label, address
push1| size (+up,-down)
Ret | -
Note | Leftmost is MSB
| Content |Addr| HexOffs|DecOffs

memviewb(3)

View memory dump in flat layout
push3| Type: 0=Hex, 1=Decimal
push2| +size
push1| label, address
Ret | -
Note | -

mem_reset(2)

Clear memory by arg2 bytes
push2| number of bytes to clear
push1| offset / location
Ret | -
Note | Clear by bytes

mem_set(3)

Set memory with arg3 byte
push3| Set byte to use
push2| size of bytes to set
push1| Address to set
Ret | -
Note | Set by bytes

mem_swap(2)

Swap two dword memory content
Swap two dword array elements
push2| Address of item
push1| Address of item
Ret | -
Note | Could use LEA / OFFSET

mem_insertr(2)

Insert register content into memory
push2| Value to insert
push1| Target location
Ret | -
Note | Put value in register first

mem_insert(4)

Insert bytes into memory
push4| Num of source bytes to write
push3| Source Address
push2| Destn offsets to write
push1| Destn address
Ret | -
Note | Overwrites

mem_copy(3)

Copy memory content of R8 bytes
push3| Number of bytes to copy
push2| Source pointer
push1| Destination pointer
Ret | -
Note | -

mem_lcopy(3)

Copy memory content of arg3 bytes
push3| Number of bytes to copy (Large n)
push2| Source pointer (from)
push1| Destination pointer (to)
Ret | -
Note | -

mem_tofile(3)

Copy memory content to a new file
push3| Address of new filename
push2| Size to copy
push1| Address of memory source
Ret | A new filled file
Note | Must be a new file or else...

prntu(1)

Display unsigned Decimal
push1| Value to display
Ret | -
Note | -

prnint(1)

Display Signed Decimal Integer
push1| Value to display
Ret | -
Note | -

prnhexu(1)

Display Unsigned Hexadecimal
push1| Value to display
Ret | -
Note | Imm. must be in valid format

prnhex(1)

Display Signed Hexadecimal
push1| Value to display
Ret | -
Note | Imm must be in valid format

prnoct(1)

Display Octal (signed)
push1| Value to display
Ret | -
Note | Imm. must be in valid format

prnoctu(1)

Display Octal (unsigned)
push1| Value to display.
Ret | -
Note | Imm. must be in valid format

prnbinu(1)

Display unsigned Binary
push1| Value to display
Ret | -
Note | Imm. must be in valid format

prnbin(1)

Display Signed Binary
push1| Value to display.
Ret | -
Note | Imm. must be in valid format

prnbinb(1)

Display Unsigned Binary with separators
push1| Value to display
Ret | -
Note | Imm must be in valid format
| Full display

fpbin(2)

Display REAL8 Floating Point Binary
push2| A DQ/double/REAL8, upper DWORD
push1| A DQ/double/REAL8, lower DWORD
Ret | -
Note | push1 and push2 are the same double

fpbind(1)

Display REAL4 Floating Point Binary
push1| REAL4 value to display
Ret | -
Note | -

fpbinx(1)

Display REAL10 Floating Point Binary
push1| Address of the REAL10
Ret | -
Note | DT type.

prnintq(2)

Display Signed 64-bit integer (qword)
push2| The DQ's upper dword
push1| The DQ's lower dword
Ret | -
Note | -

prnintqu(2)

Display Unsigned 64-bit integer (qword)
push2| The DQ's upper dword
push1| The DQ's lower dword
Ret | -
Note | -

prnhexq(2)

Display Signed 64-bit hexadecimal (qword)
push2| The DQ's upper dword
push1| The DQ's lower dword
Ret | -
Note | -

prnhexqu(2)

Display Unsigned 64-bit hex (qword)
push2| The DQ's upper dword
push1| The DQ's lower dword
Ret | -
Note | -

prnoctq(2)

Display Signed 64-bit octal (qword)
push2| The DQ's upper dword
push1| The DQ's lower dword
Ret | -
Note | -

prnoctqu(2)

Display Unsigned 64-bit octal (qword)
push2| The DQ's upper dword
push1| The DQ's lower dword
Ret | -
Note | -

prndbl(2)

Display double precision (REAL8)
push2| REAL8 upper dword
push1| REAL8 lower dword
Ret | -
Note | Var should be init as FP (0.0)

prndble(3)

Display REAL8 with decimal places
push3| Decimal places (1-15)
push2| REAL8 upper dword
push1| REAL8 lower dword
Ret | -
Note | Var should be init as FP (0.0)

prnflt(1)

Display single precision (REAL4)
push1| REAL4 value to display
Ret | -
Note | Var should be initd to FP (0.0)

prnflte(2)

Display single-precision (REAL4) with decimal places
push2| Decimal places (1-7)
push1| REAL4 value to display
Ret | -
Note | Var should be initd to FP (0.0)

prndblx(1)

Display unrounded extended-precision (REAL10)
push1| The address of a DT in memory
Ret | -
Note | Var should be initd to FP (0.0)
| Variable must be of type DT.

fltsplit(1)/2

Split a REAL4 into parts
push1| REAL4 value
Ret | Integer in EAX. Fraction in ST0.
Note | Takes normal FP value only
| Destroys FPU

decodeBASE64(2)

Decode 0-ended BASE64 to ASCII (RFC4648)
push2| Address of destination buffer
push1| Address of string to decode
Ret | -
Note | Arg2 at least 2/3 the size of Arg1
| Returned string are null-ended

encodeBASE64(2)

Encode 0-ended ASCII to BASE64 (RFC 4648)
push2| Address of destination buffer
push1| Address of string to encode
Ret | -
Note | Arg2 twice the size of Arg1
| Both strings are null-ended

int2str(4)

Convert integer to 0-ended string
push4| Signness. 0=unsigned. 1=signed
push3| Target base (2 to 36 only)
push2| Buffer's address
push1| Value
Ret | String in the sent buffer
Note | Arg2 size must reflect # of digits

dec2str(2)

Convert signed int to 0-ended decimal string
push2| Buffer's address
push1| Integer to convert
Ret | String in buffer
Note | Buffer must be large enough

dec2stru(2)

Convert unsigned int to 0-ended dec string
push2| Buffer's address
push1| Integer to convert
Ret | String in buffer
Note | Buffer must be large enough

oct2str(2)

Convert signed octal to 0-ended string
push2| Buffer's address
push1| Integer to convert
Ret | String in buffer
Note | Buffer must be large enough

oct2stru(2)

Convert unsigned octal to 0-ended octal string
push2| Buffer's address
push1| Integer to convert
Ret | String in buffer
Note | Buffer must be large enough

hex2str(2)

Convert signed int to 0-ended hex string
push2| Buffer's address
push1| Integer to convert
Ret | String in buffer
Note | Buffer must be large enough

hex2stru(2)

Convert unsigned int to 0-ended hex string
push2| Buffer's address
push1| Integer to convert
Ret | String in buffer
Note | Buffer must be large enough

bin2str(2)

Convert signed bin to 0-ended bin string
push2| Buffer's address
push1| Integer to convert
Ret | String in buffer
Note | Buffer must be large enough (>34)

bin2stru(2)

Convert unsigned bin to 0-ended bin string
push2| Buffer's address
push1| Integer to convert
Ret | String in buffer
Note | Buffer must be large enough (>32)

dbl2str(3)

Convert REAL8 to 0-ended string
push3| Address of bufffer
push2| Double's upper DWORD
push1| Double's lower DWORD
Ret | 0-ended string in sent buffer.
Note | Buffer > 24

double2str(4)

Convert REAL8 with decimal places to 0-ended string
push4| Buffer's address
push3| Decimal places
push2| Double's upper DWORD
push1| Double's lower DWORD
Ret | 0-ended string in sent buffer.
Note | Buffer > 24

flt2str(2)

Convert REAL4 to 0-ended string
push2| Buffer's address to save to
push1| REAL4 value
Ret | 0-ended string in sent buffer.
Note | Buffer > 15

flte2str(3)

Convert REAL4 with decimal places to 0-ended string
push3| Buffer's address to save to
push2| Decimal places. (1-7)
push1| REAL4 value
Ret | 0-ended string in sent buffer.
Note | Buffer > 15

dblx2str(2)

Convert REAL10 to 0-ended string
push2| Buffer address to save to
push1| Address of a DT to convert
Ret | 0-ended string in sent buffer.
Note | push1 must point to type DT.
| Buffer >= 32
| Unrounded

dumpmmxf(1)

Dump MMX registers, reversed view.
Formatted.

push1| Type of display:
0 - Unsigned integer BYTE
1 - Signed integer BYTE
2 - Unsigned integer WORD
3 - Signed integer WORD
4 - Unsigned integer DWORD
5 - Signed integer DWORD
6 - Unsigned integer QWORD
7 - Signed integer QWORD
8 - Unsigned hex BYTES

Note | -

dumpmmxf(1)

Dump MMX registers. Formatted.

push1| Type of display:
0 - Unsigned integer BYTE
1 - Signed integer BYTE
2 - Unsigned integer WORD
3 - Signed integer WORD
4 - Unsigned integer DWORD
5 - Signed integer DWORD
6 - Unsigned integer QWORD
7 - Signed integer QWORD
8 - Unsigned hex BYTES

Note | -

dumpmmxr(1)

Dump MMX registers, reversed view

push1| Type of display:
0 - Unsigned integer BYTE
1 - Signed integer BYTE
2 - Unsigned integer WORD
3 - Signed integer WORD
4 - Unsigned integer DWORD
5 - Signed integer DWORD
6 - Unsigned integer QWORD
7 - Signed integer QWORD
8 - Unsigned hex BYTES

Note | -

dumpmmx(1)

Dump MMX registers

push1| Type of display:
0 - Unsigned integer BYTE
1 - Signed integer BYTE
2 - Unsigned integer WORD
3 - Signed integer WORD
4 - Unsigned integer DWORD
5 - Signed integer DWORD
6 - Unsigned integer QWORD
7 - Signed integer QWORD
8 - Unsigned hex BYTES

Note | -

dumpxmmrf(1)

Dump XMM registers, reversed view. Formatted.

push1| Type of display:
0 - Unsigned integer BYTE
1 - Signed integer BYTE
2 - Unsigned integer WORD
3 - Signed integer WORD
4 - Unsigned integer DWORD
5 - Signed integer DWORD
6 - Unsigned integer QWORD
7 - Signed integer QWORD
8 - Singles
9 - Doubles
10- Unsigned hex BYTES

Note | -

dumpxmmf(1)

Dump XMM registers. Formatted.

push1| Type of display:
0 - Unsigned integer BYTE
1 - Signed integer BYTE
2 - Unsigned integer WORD
3 - Signed integer WORD
4 - Unsigned integer DWORD
5 - Signed integer DWORD
6 - Unsigned integer QWORD
7 - Signed integer QWORD
8 - Singles
9 - Doubles
10- Unsigned hex BYTES

Note | -

dumpxmmr(1)

Dump XMM registers, reversed view

push1| Type of display:
0 - Unsigned integer BYTE
1 - Signed integer BYTE
2 - Unsigned integer WORD
3 - Signed integer WORD
4 - Unsigned integer DWORD
5 - Signed integer DWORD
6 - Unsigned integer QWORD
7 - Signed integer QWORD
8 - Singles
9 - Doubles
10- Unsigned hex BYTES

Note | -

dumpxmm(1)

Dump XMM registers

push1| Type of display:
0 - Unsigned integer BYTE
1 - Signed integer BYTE
2 - Unsigned integer WORD
3 - Signed integer WORD
4 - Unsigned integer DWORD
5 - Signed integer DWORD
6 - Unsigned integer QWORD
7 - Signed integer QWORD
8 - Singles
9 - Doubles
10- Unsigned hex BYTES

Note | -

dumpymmr(1)

Dump YMM registers. Reversed. Formatted.

push1| Type of display:
0 - Unsigned integer BYTE
1 - Signed integer BYTE
2 - Unsigned integer WORD
3 - Signed integer WORD
4 - Unsigned integer DWORD
5 - Signed integer DWORD
6 - Unsigned integer QWORD
7 - Signed integer QWORD
8 - Singles
9 - Doubles
10- Unsigned hex BYTES

Note | Expects AVX CPU.

Note | Byte options need wide display

dumpymmf(1)

Dump YMM registers. Formatted.

push1| Type of display:
0 - Unsigned integer BYTE
1 - Signed integer BYTE
2 - Unsigned integer WORD
3 - Signed integer WORD
4 - Unsigned integer DWORD
5 - Signed integer DWORD
6 - Unsigned integer QWORD
7 - Signed integer QWORD
8 - Singles
9 - Doubles
10- Unsigned hex BYTES

Note | Expects AVX CPU

| Byte options need wide display

dumpymmr(1)

Dump YMM registers. Reversed

push1| Type of display:
0 - Unsigned integer BYTE
1 - Signed integer BYTE
2 - Unsigned integer WORD
3 - Signed integer WORD
4 - Unsigned integer DWORD
5 - Signed integer DWORD
6 - Unsigned integer QWORD
7 - Signed integer QWORD
8 - Singles
9 - Doubles
10- Unsigned hex BYTES

Note | Expects AVX CPU

| Byte options need wide display

dumpymm(1)

Dump YMM registers

push1| Type of display:
0 - Unsigned integer BYTE
1 - Signed integer BYTE
2 - Unsigned integer WORD
3 - Signed integer WORD
4 - Unsigned integer DWORD
5 - Signed integer DWORD
6 - Unsigned integer QWORD
7 - Signed integer QWORD
8 - Singles
9 - Doubles
10- Unsigned hex BYTES

Note | Expects AVX CPU

| Byte options need wide display

prndblf(2)

Display Double-precision. Formatted
push1| Double's upper dword.
push1| Double's lower dword.
Ret | -
Note | -

prnflt(1)

Display Single-precision. Formatted
push1| Single value to display.
Ret | -
Note | -

prnhexbf(1)

Display Unsigned Hex Byte (formatted)
push1| Byte value to display.
Ret | -
Note | Internal use

prnintuf(1)

Display 32-bit unsigned int (formatted)
push1| Dword Value to display
Ret | -
Note | Internal use

prnintf(1)

Display 32-bit signed integer (formatted)
push1| Dword Value to display
Ret | -
Note | Internal use

prnintqf(2)

Display 64-bit signed int (formatted)
push1| The quad's upper dword
push1| The quad's lower dword
Ret | -
Note | Internal use

prnintqf(2)

Display 64-bit unsigned int (formatted)
push1| The quad's upper dword
push1| The quad's lower dword
Ret | -
Note | Internal use

prnintbf(2)

Display Byte Decimal (formatted)
push2| 0-unsigned. 1-signed
push1| Byte value to display.
Ret | -
Note | Use "dword" override for push1

prnintb(2)

Display Byte Decimal
push2| 0-unsigned. 1-signed
push1| Byte value to display.
Ret | -
Note | Use "dword" override for push1

prnintwf(2)

Display 16-bit Decimal (formatted)
push2| 0-unsigned. 1-signed
push1| Word value to display
Ret | -
Note | Use "dword" override for push1

prnintw(2)

Display 16-bit Decimal
push2| 0-unsigned. 1-signed
push1| Word value to display.
Ret | -
Note | Use "dword" override for push1

clearxmm

Clear all XMM registers
Arg | -
Ret | -
Note | -

clearymm

Clear all YMM registers
Arg | -
Ret | -
Note | Expects AVX CPU

sse_flags

Display MXCSR register
Arg | -
Ret | -
Note | -

sse_round(1)

Set SSE rounding control
push1| Rounding mode
0 = near
1 = down
2 = up
3 = zero
Ret | -
Note | -

rstmxcscr

Reset SIMD Flags (mxcsr) to CPU default
Arg | -
Ret | -
Note | -

fpdinfo(2)

Display Double FP information
push2| REAL8's upper dword
push1| REAL8's lower dword
Note | -
Ret | -

fpfinfo(1)

Display Float FP information
push1| REAL4 value
Ret | -
Note | -

fpxinfo(1)

Display REAL10 FP information
push1| Address of a REAL10
Ret | -
Note | -

fpu_stack

Display FPU stack
Arg | -
Ret | -
Note | Display is real10

fpu_sflag

Display FPU Status Flag
Arg | -
Ret | -
Note | -

fpu_cflag

Display FPU Control Flag
Arg | -
Ret | -

fpu_tag

Display FPU Tag Register

Arg | -
Ret | -
Note | 00 = valid
| 01 = zero
| 10 = invalid
| 11 = empty

fpu_reg(1)

Display one FPU stack register

push1| Stack # to display (0-7)
Ret | -
Note | Special values show nothing

fpu_precision(1)

Set FPU precision control

push1| Precision mode:
| 0 = real8
| 1 = real10
| 2 = real4
Ret | -
Note | FSAVE/FINIT will reset it

fpu_round(1)

Set FPU rounding control

push1| Rounding mode:
| 0 = near
| 1 = down
| 2 = up
| 3 = zero
Ret | -
Note | FSAVE/FINIT will reset it

rndint/1

Generate 1 pseudorandom integer

Arg | -
Ret | Random integer in EAX
Note | -

randm(1)/1

Get 1 unsigned random integer

push1| MAX value (0-MAX)
Ret | Random integer in EAX
Note | 0, MAX inclusive

randq(2)

Generate unique random integers from 0 to MAX-1 and save and array

push2| Addr of array initialized to -1
push1| MAX value
Ret | Random elements in array
Note | Example: MAX = 50
| elems dd 50 dup(-1)
| elems: TIMES 50 dd -1

factorial(1)/1

Get factorial

push1| +Value
Ret | Factorial in EAX
Note | Limit to !12 Only

powint(2)/1

Calculate 32-bit integral power

push2| +Exponent/power (int)
push1| Base (int)
Ret | Result in EAX
| # if overflow
Note | Var must be of type DD
| Takes +exponent only

rad2deg(2)/1

Convert Radian to Degree

push2| REAL8 value upper dword, in RADIAN
push1| REAL8 value lower dword, in RADIAN
Ret | DEGREE value in ST0
Note | Var should be init as FP (0.0)
| Destroys FPU
| push1 & push2 is same double arg.

deg2rad(2)/1

Convert Degree to Radian

push2| REAL8 value upper dword, in DEGREE
push1| REAL8 value lower dword, in DEGREE
Ret | RADIAN value in ST0
Note | Var should be init as FP (0.0)
| push1 & push2 is same double arg.
| Destroys FPU

fpow(4)/1

Calculate x^n

push4| pow upper DWORD (+4)
push3| pow lower DWORD
push2| +base upper DWORD (+4)
push1| +base lower DWORD
Ret | Raised value in ST0 (Double)
Note | base must be positive
| push1 and push2 is one double
| push3 and push4 is one double
| For cubic root, use
| pow=0.3333333333333333
| Destroys FPU

g_ratio(3)/1

Find golden ratio for multi ops

push3| 0 - mul
| 1 - div
| 2 - pow
push2| REAL8 value, upper DWORD
push1| REAL8 value, lower DWORD
Ret | golden ratio in ST0 (double)
Note | 1.618033988749894848
| ST0 = arg <op> g.ratio
| Destroys FPU

lg10(2)/1

Find Common Log of Base 10

push2| REAL8 value, upper DWORD
push1| REAL8 value, lower DWORD
Ret | Value in ST0
Note | Destroys FPU

ln10(2)/1

Find Natural Log

push2| REAL8 value, upper DWORD
push1| REAL8 value, lower DWORD
Ret | Value in ST0
Note | Destroys FPU

isintREAL4(1)/1

Check if a REAL4 is a qualified integer

push1| REAL4 value
Ret | EAX: 0-no,1-yes
Note | Test normal FP values only

isintREAL8(2)/1

Check if a REAL8 is a qualified integer
push2| REAL8 value, upper DWORD
push1| REAL8 value, lower DWORD
Ret | EAX: 0-no,1=yes
Note | Test normal FP values only

iseven(1)/1

Check if integer is even number
push1| Integer to check
Ret | 0 if no. 1 if yes
Note | -

isodd(1)/1

Check if integer is odd number
push1| Integer to check
Ret | 0 if no. 1 if yes
Note | -

bconv(2)

Convert and display integer
push2| Display base to use (2 to 36)
push1| Integer to convert
Ret | -
Note | -

bitfield(3)

Display a bitfield from a 64-bit data
push3| lower bit
push2| higher bit
push1| Value to be extracted from
Ret | -
Note | Display bit range

str2int(1)/1

Convert 0-ended string to integer
push1| Address of the 0-ended string
Ret | Integer in EAX
Note | Expected Suffix- b(bin), d(dec)
| h(hex), o,q(octal). 'd' is optional
| String must be of type DB
| Digits string must be valid

str2dbl(1)/1

Convert string to double
push1| Address of a 0-ended string
Ret | Double-precision ST0
Note | String must be of type DB
| Destroys FPU

str2flt(1)/1

Convert string to float
push1| Address of a 0-ended string
Ret | Single-precision in ST0
Note | Limit digits 7
| String must be of type DB
| Destroys FPU

dbl2int(3)/2

Convert REAL8 to integer
push3| Rounding mode
0 = near
1 = down
2 = up
3 = zero
push2| REAL8 value upper DWORD
push1| REAL8 value lower DWORD
Ret | 64-bit integer in;
EAX(low)
EDX(High)
Note | Precision will be truncated
Input must be in FP format

dbl2int(2)/2

Convert REAL8 to integer, truncate to 0
push2| REAL8 value upper DWORD
push1| REAL8 value lower DWORD
Ret | 64-bit integer in;
EAX(low)
EDX(High)
Note | Precision will be truncated
Input must be in FP format

flt2int(2)/1

Convert REAL4 to integer
push2| Rounding mode:
0-Near
1-Down
2-Up
3-Zero
push1| REAL4 value
Ret | Integer in EAX
Note | Precision will be truncated
Input must be in FP format

fltt2int(1)/1

Convert REAL4 to integer, truncate to 0
push1| REAL4 value
Ret | Integer in EAX
Note | Input must be in FP format

int2flt(1)/1

Convert integer to real4
push1| Integer value to convert
Ret | Real4 in ST0
Note | Destroys FPU

readint/1

Get multi suffix integer from keyboard
Arg | -
Ret | Integer in EAX
Note | Expected Suffix- b(binary), d(dec)
| h(hex), o,q(octal). d is optional
| Digits must be in valid format

readdbl/1

Get double-precision from stdin
Arg | -
Ret | Double precision in ST0
Note | Doesn't take E format
| Must use FP format for input
| Destroys FPU

readflt/1

Get single-precision from stdin
Arg | -
Ret | Single precision in ST0
Note | Doesn't take E format
| Must use FP format for input
| Destroys FPU

chr_isdigit(1)/1

Check if alphanumeric digit
push1| Byte char to check
Ret | EAX - 0 if no. 1 if yes
Note | Digit '0' to '9' only.

chr_isalpha(1)/1

Check if alphabet
push1| Byte char to check
Ret | EAX - 0 if no. 1 if yes
Note | 'A'-'Z' and 'a'-'z'

chr_islower(1)/1

Check if lowercase
push1| Byte char to check
Ret | EAX - 0 if no. 1 if yes
Note | -

chr_isupper(1)/1

Check if uppercase
push1| Byte char to check
Ret | EAX - 0 if no. 1 if yes
Note | -

chr_change(3)

Change char occurrences from a 0-ended string to char in arg3
push3| Char byte to use
push2| Char byte to change
push1| Address of string
Ret | -
Note | -

chr_chcase(1)/1

Alternate character case
push1| Byte char to change
Ret | Modified char
Note | -

chr_tolower(1)/1

Change to lowercase
push1| Byte to change
Ret | Modified char in AL
Note | -

chr_toupper(1)/1

Change to uppercase
push1| Character to change
Ret | Modified char in AL
Note | -

chr_find(2)/1

Find char byte from a 0-ended string
push2| String address to look from
push1| Character byte to find
Ret | Index in EAX. -1 if not found
Note | Will stop at first one found

chr_count(2)/1

Count a char from a 0-ended string
push2| Character to count
push1| String's address
Ret | Count in EAX. 0 if none
Note | -

chr_shuffle(1)

Shuffle a 0-ended string
push1| String's address
Ret | Same string shuffled
Note | -

byte_count(3)/1

Count byte recurrences
push3| The byte to count
push2| Size of source buffer
push1| Source address
Ret | Count in EAX. 0 if none
Note | -

ascii(1)

Display ASCII equivalences
push1| Char or Hex value to display
Ret | -
Note | -

str_trim(2)

Trim a 0-ended string
push2| # of chars to cut
push1| Address of the 0-ended string
Ret | The same string trimmed
Note | String will be 0-ended

str_wordcnt(1)/1

Count words of a 0-ended string
push1| Address of string
Ret | Word count
Note | Single char is a word

str_token(2)

Display tokens off a 0-ended string
push2| Addr of 0-ended delimiter string
push1| Addr of the string
Ret | -
Note | eg, delimiter db '? .-',0

str_copy(3)

Copy a string to another array
push3| Size of bytes to copy
push2| Source string's address
push1| Target buffer's address
Ret | Copied string at the sent address
Note | Target must fit the size to copy
| Target buffer is 0-ended

str_cmpz(2)/1

Compare two 0-ended strings
push2| String 1 address
push1| String 2 address
Ret | EAX = 1 if Equal, 0 if not
Note | The two strings must be 0-ended

str_cmp(3)/1

Compare strings with size
push3| Size of bytes to compare
push2| String 1 address
push1| String 2 address
Ret | EAX = 1 if Equal, 0 if not
Note | -

str_find(3)/1

Search for a sub-string
push3| Size of key string
push2| Addr of key string
push1| Addr of 0-ended source string
Ret | Address. -1 -Not found
Note | Case-sensitive

str_findz(2)/1

Search for a sub-string from a C string
push2| Addr of 0-ended search/key string
push1| Addr of 0-ended source string
Ret | Location. -1 = Not found
Note | Case-sensitive

str_append(5)/1

Combine two 0-ended strings with size & copy to a new 0-ended buffer
push5| # of bytes to copy from str2
push4| Separator byte. 0-if none
push1| Addr of new buffer
push3| Addr of str2
push2| Addr of str1 (0-ended)
Ret | 0-ended combined string
| EAX - Combined string length
Note | 1st string + 2nd string
| Buffer must be large enough

str_appendz(4)/1

Combine two 0-ended strings
push4| Separator byte. 0-if none
push3| Addr of buffer
push2| Addr of second string
push1| Addr of first string
Ret | 0-ended combined string
| EAX - Combined string length
Note | 1st string + 2nd string
| Buffer must be large enough

str_reverse(1)

Reverse 0-ended string
push1| Address of the 0-ended string
Ret | The same string reversed
Note | -

str_reverse2(1)

Reverse 0-ended aligned-16 string
push1| Address of the 0-ended string
Ret | The same string reversed
Note | String must be align16

str_tolower(1)

Change 0-ended string to lower case
push1| Address of the 0-ended string
Ret | The same string converted
Note | -

str_toupper(1)

Change 0-ended string to upper case
push1| Address of the 0-ended string
Ret | The same string converted
Note | -

str_length(1)/1

Find length of a 0-ended string
push1| String's address
Ret | size in EAX.
Note | -

str_lengthd(2)/1

Get length of a 0-ended string with delimiter
push2| Delimiter byte
push1| String's address
Ret | size in EAX
Note | -

str_length2(1)/1

Find length of a 0-ended string (aligned)
push1| String's address
Ret | size in EAX.
Note | String must be aligned 16

scan_dword(3)/1

Find first matching dword
push3| Size of bytes to scan
push2| The dword key immediate
push1| Starting location
Ret | Offset. -1 if not found
Note | -

scan_word(3)/1

Find first matching two-byte
push3| Size of bytes to scan
push2| The word key immediate
push1| Starting location
Ret | Offset. -1 if not found
Note | -

scan_byte(3)/1

Find first matching byte
push3| Size of bytes to scan
push2| The byte key immediate
push1| Starting location
Ret | Offset. -1 if not found
Note | -

sort_byte(3)

Sort char/byte array (signed)
push3| 0-ascending, 1-descending
push2| Number of array elements
push1| Address of array (DB)
Ret | Return the sorted array
Note | Elements are of type DB

sort_int(3)

Sort array of signed int (bubble sort)
push3| 0-ascending, 1-descending
push2| Number of array elements
push1| Address of array
Ret | Return the sorted array
Note | Elements are of type DWORD

sort_intu(3)

Sort array of unsigned int (bubble sort)
push3| 0-ascending, 1-descending
push2| Number of array elements
push1| Address of array
Ret | Return the sorted array
Note | Elements are of type DWORD

sort_dbl(3)

Sort array of doubles (bubble sort)
push3| 0-ascending, 1-descending
push2| Number of array elements
push1| Address of array
Ret | Return the sorted array
Note | Elements are of type DQ

sort_dblx(3)

Sort array of REAL10 (bubble sort)
push3| 0-ascending, 1-descending
push2| Number of array elements
push1| Address of array
Ret | Return the sorted array
Note | Elements are of type DT

sortflt(2)

Sort array of singles (bubble sort)
push3| 0-ascending, 1-descending
push2| Number of array elements
push1| Address of array (DD)
Ret | Return the sorted array
Note | Elements are of type DD

aprnbyte(3)

Display array of unsigned hex bytes. Little-Endian
push3| Separator byte. 0 if none
push2| Number of elements
push1| Address of array
Ret | -
Note | Elements are of type DB

aprnbyter(3)

Display array of unsigned hex bytes. Reversed-view
push3| Separator byte. 0 if none
push2| Number of elements
push1| Address of array
Ret | -
Note | Elements are of type DB

aprnbyte(3)

Display array of signed int bytes. Little-Endian
push3| Separator byte. 0 if none
push2| Number of elements
push1| Address of array
Ret | -
Note | Elements are of type DB

aprnbyteu(3)

Display array of unsigned int bytes. Little-Endian
push3| Separator byte. 0 if none
push2| Number of elements
push1| Address of array
Ret | -
Note | Elements are of type DB

aprnbyter(3)

Display array of signed int bytes. Reversed-view
push3| Separator byte. 0 if none
push2| Number of elements
push1| Address of array
Ret | -
Note | Elements are of type DB

aprnbyteur(3)

Display array of unsigned bytes. Reversed-view
push3| Separator byte. 0 if none
push2| Number of elements
push1| Address of array
Ret | -
Note | Elements are of type DB

aprnword(3)

Display array of signed words. Little-Endian
push3| Separator byte. 0 if none
push2| Number of elements
push1| Address of array
Ret | -
Note | Elements are of type DW

aprnwordu(3)

Display array of unsigned words. Little-Endian
push3| Separator byte. 0 if none
push2| Number of elements
push1| Address of array
Ret | -
Note | Elements are of type DW

aprnwordr(3)

Display array of signed words. Reversed-view
push3| Separator byte. 0 if none
push2| Number of elements
push1| Address of array
Ret | -
Note | Elements are of type DW

aprnwordur(3)

Display array of unsigned words. Reversed-view
push3| Separator byte. 0 if none
push2| Number of elements
push1| Address of array
Ret | -
Note | Elements are of type DW

aprnintd(3)

Display array of signed dwords. Little-Endian
push3| Separator byte. 0 if none
push2| Number of elements
push1| Address of array
Ret | -
Note | Elements are of type DD

aprnintdu(3)

Display array of unsigned dwords. Little-Endian
push3| Separator byte. 0 if none
push2| Number of elements
push1| Address of array
Ret | -
Note | Elements are of type DD

aprnintdr(3)

Display array of signed dwords. Reversed-view
push3| Separator byte. 0 if none
push2| Number of elements
push1| Address of array
Ret | -
Note | Elements are of type DD

aprnintdur(3)

Display array of unsigned dwords. Reversed-view
push3| Separator byte. 0 if none
push2| Number of elements
push1| Address of array
Ret | -
Note | Elements are of type DD

aprint(3)

Display array of signed longs. Little-Endian
push3| Separator byte. 0 if none
push2| Number of elements
push1| Address of array
Ret | -
Note | Elements are of type DQ

aprintu(3)

Display array of unsigned longs. Little-Endian
push3| Separator byte. 0 if none
push2| Number of elements
push1| Address of array
Ret | -
Note | Elements are of type DQ

aprintr(3)

Display array of signed longs. Reversed view
push3| Separator byte. 0 if none
push2| Number of elements
push1| Address of array
Ret | -
Note | Elements are of type DQ

aprintur(3)

Display array of unsigned longs. Reversed-view
push3| Separator byte. 0 if none
push2| Number of elements
push1| Address of array
Ret | -
Note | Elements are of type DQ

aprnflt(3)

Display array of REAL4s. Little-Endian
push3| Separator byte. 0 if none
push2| Number of elements
push1| Address of array
Ret | -
Note | Elements are of type DD/REAL4

aprnfltr(3)

Display array of REAL4s. Reversed view
push3| Separator byte. 0 if none
push2| Number of elements
push1| Address of array
Ret | -
Note | Elements are of type DD/REAL4

aprndbl(3)

Display array of REAL8s. Little-Endian
push3| Separator byte. 0 if none
push2| Number of elements
push1| Address of array
Ret | -
Note | Elements are of type DQ/REAL8

aprndblr(3)

Display array of REAL8s. Reversed view
push3| Separator byte. 0 if none
push2| Number of elements
push1| Address of array
Ret | -
Note | Elements are of type DQ/REAL8

aprndblx(3)

Display array of REAL10s. Little-Endian
push3| Separator byte. 0 if none
push2| Number of elements
push1| Address of array
Ret | -
Note | Elements are of type DT/REAL10

aprndblxr(3)

Display array of REAL10s. Reversed view
push3| Separator byte. 0 if none
push2| Number of elements
push1| Address of array
Ret | -
Note | Elements are of type DT/REAL10

aprnhbytef(3)

Display array of unsigned hex bytes. Little-Endian. Formatted.
push3| Separator byte. 0 if none
push2| Number of elements
push1| Address of array
Ret | -
Note | Elements are of type DB

aprnhbyterf(3)

Display array of unsigned hex bytes. Reversed-view. Formatted.
push3| Separator byte. 0 if none
push2| Number of elements
push1| Address of array
Ret | -
Note | Elements are of type DB

aprnbytef(3)

Display array of signed bytes. Little-Endian. Formatted.
push3| Separator byte. 0 if none
push2| Number of elements
push1| Address of array
Ret | -
Note | Elements are of type DB

aprnbyteuf(3)

Display array of unsigned bytes. Little-Endian. Formatted.
push3| Separator byte. 0 if none
push2| Number of elements
push1| Address of array
Ret | -
Note | Elements are of type DB

aprnbyterf(3)

Display array of signed bytes. Reversed-view. Formatted.
push3| Separator byte. 0 if none
push2| Number of elements
push1| Address of array
Ret | -
Note | Elements are of type DB

aprnbyteurf(3)

Display array of unsigned bytes. Reversed-view. Formatted.
push3| Separator byte. 0 if none
push2| Number of elements
push1| Address of array
Ret | -
Note | Elements are of type DB

apnwordf(3)

Display array of signed words. Little-Endian. Formatted.
push3| Separator byte. 0 if none
push2| Number of elements
push1| Address of array
Ret | -
Note | Elements are of type DW

apnworduf(3)

Display array of unsigned words. Little-Endian. Formatted.
push3| Separator byte. 0 if none
push2| Number of elements
push1| Address of array
Ret | -
Note | Elements are of type DW

apnwordrf(3)

Display array of signed words. Reversed-view. Formatted.
push3| Separator byte. 0 if none
push2| Number of elements
push1| Address of array
Ret | -
Note | Elements are of type DW

apnwordurf(3)

Display array of unsigned words. Reversed-view. Formatted.
push3| Separator byte. 0 if none
push2| Number of elements
push1| Address of array
Ret | -
Note | Elements are of type DW

aprintdf(3)

Display array of signed dwords. Little-Endian. Formatted.
push3| Separator byte. 0 if none
push2| Number of elements
push1| Address of array
Ret | -
Note | Elements are of type DD

aprintduf(3)

Display array of unsigned dwords. Little-Endian. Formatted.
push3| Separator byte. 0 if none
push2| Number of elements
push1| Address of array
Ret | -
Note | Elements are of type DD

aprintdrf(3)

Display array of signed dwords. Reversed-view. Formatted.
push3| Separator byte. 0 if none
push2| Number of elements
push1| Address of array
Ret | -
Note | Elements are of type DD

aprintdurf(3)

Display array of unsigned dwords. Reversed-view. Formatted.
push3| Separator byte. 0 if none
push2| Number of elements
push1| Address of array
Ret | -
Note | Elements are of type DD

aprinttf(3)

Display array of signed longs. Little-Endian. Formatted.
push3| Separator byte. 0 if none
push2| Number of elements
push1| Address of array
Ret | -
Note | Elements are of type DQ

aprinturf(3)

Display array of unsigned longs. Little-Endian. Formatted.
push3| Separator byte. 0 if none
push2| Number of elements
push1| Address of array
Ret | -
Note | Elements are of type DQ

aprinttrf(3)

Display array of signed longs. Reversed view. Formatted.
push3| Separator byte. 0 if none
push2| Number of elements
push1| Address of array
Ret | -
Note | Elements are of type DQ

aprinturf(3)

Display array of unsigned longs. Reversed-view. Formatted.
push3| Separator byte. 0 if none
push2| Number of elements
push1| Address of array
Ret | -
Note | Elements are of type DQ

aprnfltf(3)

Display array of REAL4s. Little-Endian. Formatted.
push3| Separator byte. 0 if none
push2| Number of elements
push1| Address of array
Ret | -
Note | Elements are of type DD/REAL4

aprnfltrf(3)

Display array of REAL4s. Reversed view. Formatted.
push3| Separator byte. 0 if none
push2| Number of elements
push1| Address of array
Ret | -
Note | Elements are of type DD/REAL4

aprndblf(3)

Display array of REAL8s. Little-Endian. Formatted.

push3| Separator byte. 0 if none
push2| Number of elements
push1| Address of array
Ret | -
Note | Elements are of type DQ/REAL8

aprndblrf(3)

Display array of REAL8s. Reversed view. Formatted.

push3| Separator byte. 0 if none
push2| Number of elements
push1| Address of array
Ret | -
Note | Elements are of type DQ/REAL8

prnchrp(2)

Display char pattern

push| Push the char to use
push| Push number of times to print
Ret | -
Note | -

prnspaces(1)

Print multiple whitespace

push| Push number of whitespaces
Ret | -
Note | Arguments on stack

prnstreg(1)**

Display short string off register EAX

EAX | The string immediate
Ret | -
Note | -

prnlines(1)

Print multiple lines

push1| Push number of lines
Ret | -
Note | -

prnline

Print a newline

Arg | -
Ret | -
Note | -

prnspace

Print a whitespace

Arg | -
Ret | -
Note | -

prnchar(1)

Display a char byte from memory

push1| The char's address
Ret | -
Note | -

prnchr(1)

Display a char byte

push1| The char value
Ret | -
Note | Use "dword" override if so needed

prnstrd(2)

Display a delimiter-ended string

push2| Delimiter byte value
push1| The string address
Ret | -
Note | -

prnstrz(1)

Display a 0-ended string

push1| The string address
Ret | -
Note | -

prnstr(2)

Display string with size

push2| Size of bytes to display
push1| The string address
Ret | -
Note | -

readchr(1)

Get a char from keyb and save to memory

push1| The address to save the char to
Ret | -
Note | Var should be of type DB/rb/resb

readch/1

Get a char from keyb into AL, with echo

Arg | -
Ret | The key in AL
Note | Internal use
| Use halt if needed

readstr(1)/1

Get string from keyboard (echoed)

push1| Buffer's address to save string
Ret | 0-ended string in sent buffer
| EAX= # of actual bytes in
Note | Press ENTER to finish
| Buffer must be large enough
| Ret -1 signals error
| String will be 0-ended

mem_alloc(1)/1

Request memory of n bytes

push1| Bytes requested
Ret | Pointer to the mem block. -1 = none
Note | -

mem_free(1)/1

Free memory allocated by mem_alloc

push1| Pointer from mem_alloc
Ret | 1 = success. -1 = error
Note | -

mem_load(1)/2

Load a file to memory

push1| Filename path string
Ret | EAX - block pointer
| EDX - Size
Note | Should re-claim memory at EAX
| File name must be 0-ended

file_new(1)/1

Create a new file

push1| File name path string
Ret | -1 = Error. 0 = success
Note | Filename string must be 0-ended

file_open(2)/1

Open an existing file for reading/writing

push2| Operation:

0-Read

1-Write

push1| File name path string

Ret | File descriptor. -1 if error

Note | Filename string must be 0-ended

file_read(3)/1

Read from an existing opened file

push3| Number of bytes to read

push2| Input buffer

push1| File handle from file_open

Ret | Number of actual read

Note | Filename string must be 0-ended

| Input buffer must be >= arg3

file_write(3)/1

Write to an existing opened file

push3| Number of bytes to write

push2| Source buffer's address

push1| File handle from file_open

Ret | Number of actual writes

Note | Filename string must be 0-ended

| Handle must be opened-write

file_close(1)

Close handle issued by file_open

push1| File handle

Ret | -

Note | EAX will be zeroed

file_size(1)/1

Get filesize of an opened file

push1| File handle from file_open

Ret | Size in bytes. -1 says error

Note | -

file_copy(2)

Copy a file to a new file

push2| Address of new file name

push1| Address of source file name

Ret | -

Note | Both must be 0-ended string

file_delete(1)

Delete a file on disk

push1| Address of filename string

Ret | 0-Fail (Win64), -ve (Linux64)

Note | Handle must be already closed

timer_start/1

Start timer

Arg | -

Ret | Pointer to memory

Note | Must be followed by timer_stop
to clear allocated memory

timer_stop(1)

*Stop current timer and display time lapse
in seconds*

push1| pointer from timer_start

Ret | -

Note | Should come after timer_start

delay(1)

Put execution at delay

push1| Milliseconds

Ret | -

Note | 1000ms = 1s

get_ticks/1

Get tick count

- | -

Ret | Ticks (ms) in EAX

Note | 1000ms = 1s

halt

Pause screen

Arg | -

Ret | -

Note | Hit Enter to continue

exitp

Pause then exit to system

Arg | -

Ret | -

Note | Hit Enter to exit

exitx

Exit to system

Arg | -

Ret | -

Note | -