

RPNX VA1-: RPNX Virtual Architecture v1 -bit

February 24, 2018

Contents

1	General	2
1.1	Registers	3
1.2	Instruction Opcodes	3
2	Instruction Opcodes	3
2.1	0x0 Control Operations	3
2.1.1	0x02 TGT: Target	3
2.1.2	0x03 GO: Goto	3
2.1.3	0x04 JMP: Absolute Jump	3
2.1.4	0x05 RJP: Relative Jump	4
2.1.5	0x06 PUSH: Push Register(s) to Stack	4
2.1.6	0x07 POP: Pop Register(s) from Stack	4
2.1.7	0x08 CALL: Call Function	4
2.1.8	0x09 RET: Return	4
2.1.9	0x0A CNN: Call Non-Null	4
2.1.10	0x0B MCALL: Member Function Call	4
2.1.11	0x0C MRET: Member Function Return	5
2.1.12	0x0D VCALL: Virtual Member Function Call	5
2.1.13	0x0E IF: If	5
2.1.14	0x0F SYSCALL: System Call	5
2.2	0x1 Bitwise Operations I	5
2.2.1	0x10 NAND	5
2.2.2	0x11 XOR	5
2.2.3	0x12 AND	5
2.2.4	0x13 OR	5
2.2.5	0x14 NOR	6
2.2.6	0x15 IMP	6

2.2.7	0x16 ROT	6
2.2.8	0x17 MSB: Most Significant Bit	6
2.2.9	0x18 MSM: Most Significant Mask	6
2.2.10	0x19 MSI: Most Significant Index	6
2.2.11	0x1A BCT: Bitcount	6
2.2.12	0x1A CAN: Compression AND	6
2.2.13	0x1B CXR: Compression XOR	7
2.2.14	0x1C ESW: Endian Swap	7
2.2.15	0x1D INV: Bitwise Inversion	7
2.2.16	0x1E REV: Bitwise Reverse	7
2.3	0x2 Math Operations I	7
2.3.1	0x20 ADD: Addition	7
2.3.2	0x21 SUB: Subtraction	7
2.3.3	0x22 MUL: Unsigned Multiply	7
2.3.4	0x23 IML: Signed Multiply	7
2.3.5	0x24 SXT: Sign Extension	8
2.3.6	0x25 DIV: Divide	8
2.3.7	0x26 MOD: Modulus	8
2.3.8	0x27 CMP: Unsigned Comparison	8
2.3.9	0x27 ICM: Signed Comparison	8
2.4	0x3 Data Operations I	9
2.4.1	0x30 MOV: Move	9
2.4.2	0x31 LOAD: Load Memory	9
2.4.3	0x32 STORE: Store Memory	9
2.4.4	0x33 LNN: Load Non-Null	9
2.4.5	0x34 SNN: Store Non-Null	9
2.4.6	0x35 CPY: Memory Copy	9
2.4.7	0x36 ARL: Array Load	10
2.4.8	0x36 ARS: Array Store	10
2.4.9	0x37 SAL: Structure Array Load	10
2.4.10	0x37 SAS: Structure Array Store	11
2.4.11	0x38 LCT: Load Constant	11
2.4.12	0x39 XCH: Exchange	11

1 General

Each Instruction Opcode is followed by 0 or more I/O Opcodes.

1.1 Registers

The virtual machine has 64-bit registers. These may be referred to as %r0 to %r63. A few notes though:

%r63 = %rip (instruction pointer)

%r62 = %rsp (stack pointer)

%r61 = %rtp (this pointer)

%r60 = %rvo (vtable offset)

1.2 Instruction Opcodes

Instruction Opcodes are divided into groups. Each group has a special prefix.

Control Operations Prefix 0x0. These provide flow control and mov.

Bitwise Operations I Prefix 0x1. Basic bitwise operations.

Math Operations I Prefix 0x2. Basic mathematical operations.

The order of arguments listed is the same order that the instructions take the argument opcodes in. Currently all instructions have inputs before outputs, but this is not guaranteed in any manner for future instructions (and you should not alter the order of arguments)

2 Instruction Opcodes

2.1 0x0 Control Operations

2.1.1 0x02 TGT: Target

tgt

Cross-page jump target. Non-local jumps to any instruction other than tgt will trigger a segmentation fault. This instruction is a no-op.

2.1.2 0x03 GO: Goto

go %i

Same-page goto. Takes a page sub-address as its only argument. An absolute jump within the same page. Other bits in the input are ignored. Unlike jmp and rjp, can jump to instructions other than tgt.

2.1.3 0x04 JMP: Absolute Jump

jmp %i

Absolute address jump. Triggers segfault if the jump location is not executable or if the jump target is not a tgt instruction.

2.1.4 0x05 RJP: Relative Jump

rip %i

Relative jump. Triggers segfault if the jump location is not executable or if the jump target is not a tgt instruction.

2.1.5 0x06 PUSH: Push Register(s) to Stack

push %i

This instruction pushes registers to the stack. The actual input is a single -bit word by the ordinary input channel. Registers are pushed whose 2^n -bit is set.

However, if the mask is 0, then all registers are pushed except %rip and %rsp.

2.1.6 0x07 POP: Pop Register(s) from Stack

pop %i

This instruction pops registers from the stack. The actual input is a single -bit word by the ordinary input channel. Registers are popped whose 2^n -bit is set from the input.

2.1.7 0x08 CALL: Call Function

call %i %i

Call instruction. Jumps to function %0 and pushes register mask %1 to stack. %rip is ignored as an argument.

2.1.8 0x09 RET: Return

Return instruction

2.1.9 0x0A CNN: Call Non-Null

cnn %i

Call if not null. Skips the next instruction upon return if the argument was non-null. If the argument was null, this opcode is a no-op.

2.1.10 0x0B MCALL: Member Function Call

mcall %i %i

The same as CALL except that the register %rtp is implied to be in the push-mask regardless of whether or not it was specified. %rvo is implied if it not equal to 0.

2.1.11 0x0C MRET: Member Function Return

mret

Like ret except that it will set %rvo to 0 if it is not in the popmask.

This instruction pops a -bit word from the stack. (the popmask)

All registers %rN where bit 2^N is set in the popmask are popped from the stack in reverse order from mcall. If %rvo (bit 2^{60}) is not set, then %rvo is set to 0.

Finally, the instruction pops a bit absolute jump target and then jumps to it.

2.1.12 0x0D VCALL: Virtual Member Function Call

vcall %i %i

Virtual Call. Pushes register mask %0 and then calls *(%rtp+%rvo). %rtp and %rvo are implied arguments (except where %rtp or %rvo are 0, in which case they are *not* pushed)

2.1.13 0x0E IF: If

if %i, %i

Skips the next instruction unless %0 and %1 compare equal.

2.1.14 0x0F SYSCALL: System Call

2.2 0x1 Bitwise Operations I

2.2.1 0x10 NAND

nand %i, %i, %o

Bitwise NAND of %0, %1 written to %2.

2.2.2 0x11 XOR

xor %i, %i, %o

Bitwise XOR of %0, %1 written to %2.

2.2.3 0x12 AND

and %i, %i, %o

Bitwise AND of %0, %1 written to %2.

2.2.4 0x13 OR

or %i, %i, %o

Bitwise %0 OR %1 written to %2.

2.2.5 0x14 NOR

nor %i, %i, %o

Bitwise %0 NOR %1 written to %2.

2.2.6 0x15 IMP

imp %i, %i, %o

Bitwise %0 -> %1 written to %2.

2.2.7 0x16 ROT

rot %i, %i, %o

Rotate %0 by %1 high (positive) or low (negative) and store the result in %0.

2.2.8 0x17 MSB: Most Significant Bit

msb %i, %o

Most significant set bit of %0 written to %1. If no significant bit is set, writes 0 instead.

2.2.9 0x18 MSM: Most Significant Mask

msm %i, %o

Most Significant-Set Mask of %0 to %1. If any bit in the input is set, that bit and all lower-order bits are also set in the output, otherwise the output is 0.

2.2.10 0x19 MSI: Most Significant Index

msi %i, %o

Index of most significant set bit is returned as output. If no bit is set, -1 is returned as output instead.

2.2.11 0x1A BCT: Bitcount

bct %i, %o

Returns the number of 1-bits in the input %0 and writes it to %1.

2.2.12 0x1A CAN: Compression AND

can %i, %o

Takes compression-AND of %0 and writes it to %0.

2.2.13 0x1B CXR: Compression XOR

`cxr %i, %o`

Takes compression-XOR of %0 and writes it to %1.

2.2.14 0x1C ESW: Endian Swap

`esw %i, %o`

Swaps from little to big endian and vice-versa.

2.2.15 0x1D INV: Bitwise Inversion

`inv %i, %o`

Inverts all the bits in %0 and stores the result in %1.

2.2.16 0x1E REV: Bitwise Reverse

`rev %i, %o`

Reverses the bit-order in %0 and stores the result in %1.

2.3 0x2 Math Operations I

2.3.1 0x20 ADD: Addition

`add %i, %i, %o`

Adds %0 to %1 and stores the result in %2.

2.3.2 0x21 SUB: Subtraction

`sub %i, %i, %o`

Subtracts %1 from %0 and stores the result in %2

2.3.3 0x22 MUL: Unsigned Multiply

`mul %i, %i, %o`

Multiplies %0 by %1 and stores the result in %2. (unsigned multiplication)

2.3.4 0x23 IML: Signed Multiply

`add %i, %i, %o`

Adds %0 to %1 and stores the result in %2.

2.3.5 0x24 SXT: Sign Extension

sxt %i, %o

Sign extension of %0 stored in %1.

2.3.6 0x25 DIV: Divide

div %i, %i, %o

Divides %0 by %1 if %1 is not 0 and stores the (floored) result in %2.

If %1 is 0, then stores %0 in %2 instead.

If %1 was not 0, then the next instruction is skipped.

2.3.7 0x26 MOD: Modulus

div %i, %i, %o

Divides %0 by %1 if %1 is not 0 and stores the (floored) modulus in %2.

If %1 is 0, then stores %0 in %2 instead.

If %1 was not 0, then the next instruction is skipped.

2.3.8 0x27 CMP: Unsigned Comparison

cmp %i, %i, %o

Three cases:

$\%0 < \%1$ Stores -1 in %2.

$\%0 = \%1$ Stores 0 in %2.

$\%0 > \%1$ Stores 1 in %2.

This comparison is unsigned.

2.3.9 0x27 ICM: Signed Comparison

icm %i, %i, %o

Three cases:

$\%0 < \%1$ Stores -1 in %2.

$\%0 = \%1$ Stores 0 in %2.

$\%0 > \%1$ Stores 1 in %2.

This comparison is signed.

2.4 0x3 Data Operations I

2.4.1 0x30 MOV: Move

```
mov %i %o
```

Copies the source operand to the destination operand.

2.4.2 0x31 LOAD: Load Memory

```
load %i %o
```

Loads from the value pointed to by the pointer %0 and writes it to %1.

Note: A full pointer is always read from the input operand, but the argument size will affect the amount of memory read instead of treating the other bits as 0s.

2.4.3 0x32 STORE: Store Memory

```
store %i %i
```

Writes the value %0 to the value pointed by the pointer %1. Note: A full pointer is always read from the input operand, but the argument size will affect the amount of memory stored instead.

2.4.4 0x33 LNN: Load Non-Null

```
lnn %i %o
```

Loads memory from %0 and writes it to %1 if it is non-null. If %0 is null, writes 0 to %1 instead. Note: A full pointer is always read from the input operand, but the argument size will affect the amount of memory read instead of treating the other bits as 0s.

Skips the next instruction if the pointer %0 was non-null.

2.4.5 0x34 SNN: Store Non-Null

```
snn %i %i
```

Stores the value %0 into the value pointed to by the pointer %1 if it is not null, otherwise the value is unchanged.

2.4.6 0x35 CPY: Memory Copy

```
cpy %i %i %o
```

Copies %0 bytes starting at %1 and stores them in the memory region %2.

2.4.7 0x36 ARL: Array Load

```
arl %i, %i, %i, %o
```

Dereferenced array read.

%0 Array pointer

%1 Array size

%2 Array index

%3 Output

Loads the integer from the array if %2 is less than %1, otherwise stores 0 into %3. If successful, skips the next instruction.

2.4.8 0x36 ARS: Array Store

```
ars %i, %i, %i, %i
```

Dereferenced array read.

%0 Array pointer

%1 Array size

%2 Array index

%3 Value

Stores the value into the array if it does not exceed the array bounds. Skips the next instruction if successful.

2.4.9 0x37 SAL: Structure Array Load

```
sal %i, %i, %i, %i, %i, %o
```

Like arl, but with extra parameters.

%0 Array pointer

%1 Array element size

%2 Array size

%3 Array index

%4 Read offset

%5 Output

Skips the next instruction on success.

2.4.10 0x37 SAS: Structure Array Store

sas %i, %i, %i, %i, %i, %i

Like ary, but with extra parameters.

%0 Array pointer

%1 Array element size

%2 Array size

%3 Array index

%4 Read offset

%5 Value

Skips the next instruction on success.

2.4.11 0x38 LCT: Load Constant

lce %c, %o

Loads the constant value %0 into the register %1.

2.4.12 0x39 XCH: Exchange

xch %io, %io

Exchanges the contents of the input/ouputs %0 and %1.