N1 Manual

Dirk Heisswolf

January 22, 2019

Revision History

Date	Change
January 22, 2019	Pre-release

Contents

List of Figures

List of Tables

1 Glossary

;

    End of a word definition in Forth.

ALU

    Arithmetic Logic Unit.

byte

    An 8-bit data entity.

call

    A change of the program flow, where a return address is kept on the return stack.

cell

    A data entity within a stack.

conditional branch

    A change of the program flow without return option, only if a certain (non-zero) argument value is given.

Forth

    Forth is a extensible stack-based programming language.

intermediate stack

    The section of the stack, which serves as a buffer between the lower stack and the upper stack. See Section 5 "Stacks".

IST

    A bit field in the stack instruction which contols data movement on the intermediate parameter stack or return stack. The mnemonic stands for "**I**intermediate **S**tack **T**ransition".

jump

    A change of the program flow without return option.

literal

    A fixed numerical value within the program code.

lower stack

    The section of the stack which stored in RAM. See Section 5 "Stacks".

opcode

    Encoding of a machine instruction. Short for "operation code".

parameter stack

    A LIFO storage mainly for keeping call parameters and return values.

RAM

>   Random access memory.

return stack

>   A LIFO storage mainly for maintaining return addresses of calls.

stack

>   A LIFO storage.

upper stack

>   The section of the stack, which contains the TOS. It supports reordering of its
>   storage cell. See Section 5 "Stacks".

UST

>   A bit field in the stack instruction which contols data movement between two
>   neighboring cells in the upper parameter stack or return stack. The mnemonic
>   stands for "**U**pper **S**tack **T**ransition".

word

>   The term word is used in two different contexts throughout this document.
>   It refers to either a 16-bit data entity or a callable code sequence in Forth
>   terminology.

2 Overview

The N1 is a snall stack machine, inspired by the J1 Forth CPU[1]. Just like its paragon, the N1 is a 16-bit processor wich implements basic Forth words directly in hardware. However the N1 parts from the J1's simplistic design approach in in two ways:

- The N1 support a larger code space of up to 32KB. Therefore it has its own instruction set (see Section 3 "Instruction Set".

- The N1 implements its parameter and return stacks as shallow register stacks, which overflow into RAM. The overall depth of each stack is determined by the available RAM. (see Section 5 "Stacks".

3   Instruction Set

The intent of the N1's instruction set is to map most of the essential Forth words to single cycle instructions. Figure 3-1 illustrates the basic structure of the instructuion encoding.
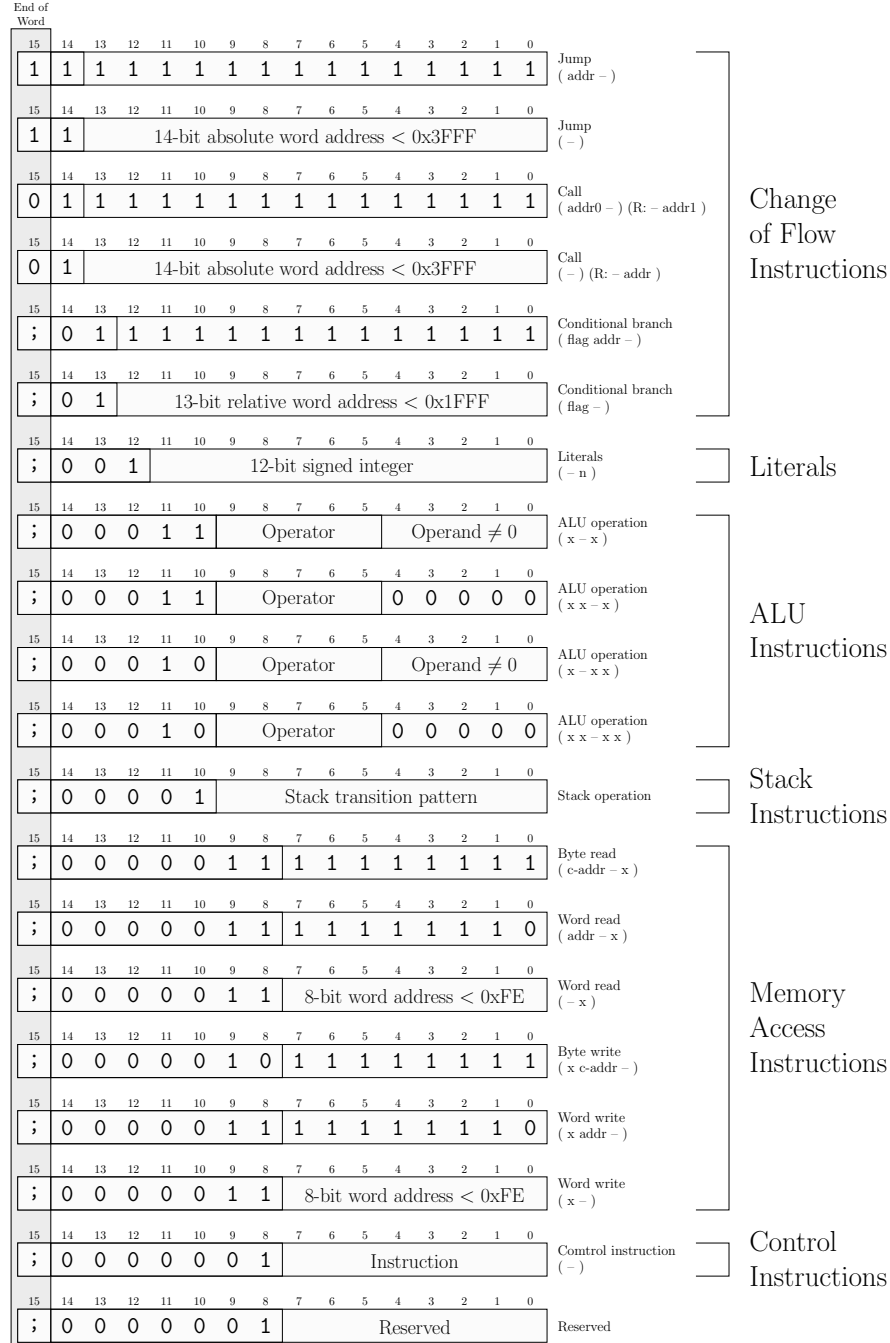

Figure 3-1: Instruction encoding

3.1 Return from a Call (;)

Rather than providing a dedicated instruction to end the execution of word in Forth and to return the program flow to its caller, the N1 allows to perform this operation in parallel to the execution of any of its instructions. Each opcode contains a bit (bit 15) to indicate, that the current instruction in the last operation in the current word. If this bit is set, the program flow will resume at the calling word as soon as the operationis performed.

As shown in Figure 3-1, bit 15 is also used to distinguish jump and call. Considering that the last call in a word definition can be optimized to a jump to the first instruction of the called word, bit 15 can ber regarded as termination bit for these instructions as well.

For a Forth compiler, this means that the semi-colon (;) always translates to setting bit 15 of the last instruction.

3.2 Jump Instructions

Jump instructions transfer the program flow to any word location within the supported 64KB program space. Jump instructions consume an absolute destination address, which can be either placed on the top of the Parameter stack or encoded into the opcode of the instruction (only for destination addresses < 0x3FFF).

3.3 Call Instructions

Call instructions temporarily transfer the program flow to any word location within the supported 64KB program space, while pushing a return address onto the return stack. Call instructions consume an absolute destination address, which can be either placed on the top of the Parameter stack or encoded into the opcode of the instruction (only for destination addresses < 0x3FFF).

3.4 Conditional Branches

Conditional branches invoke a change of program flow depending on an argument on the parameter stack. The branch destination cab be either an absolute address placed on the the top of the Parameter stack or relative relative address, encoded into the opcode of the instruction (only for destination addresses < 0x1FFF).

3.5 Literals

Signed integer literals of 12-bit length can be pushed onto the parameter stack within a single instruction. For larger integers a supplemental TBD call is required.

3.6 ALU Instructions

ALU instructions perform an operation on two cell values, resulting in a new double cell value. The reult can be either placed entirely onto the parameter stack, or truncated, discarding the most significant cell. The first operand is always taken from the Parameter stack. The second operand can be either taken from the Parameter stack or encoded into the opcode of the instruction. In the latter case, the interpretation of the embedded 5-bit value depends on the operation. It is either regarded as an unsigned (*uimm*) or a sign extended value(*simm*). Table 3-1 lists the supported ALU operations.

Table 3-1: ALU operations

Encoding	Operation	(x1 − d)	(x1 x2 − d)
00000	Signed greater-than comparison	x1 $>$ *oimm*	x1 $>$ x2
00001	Signed lower-than comparison	x1 $<$ *oimm*	x1 $<$ x2
00010	Equals comparison	x1 $\equiv$ *oimm*	x1 $\equiv$ x2
00011	Not-equals comparison	x1 $\neq$ *oimm*	x1 $\neq$ x2
00100	Sum	x1 $+$ *uimm*	x1 $+$ x2
00101	Difference	x1 $-$ *uimm*	x1 $-$ x2
00100	Sum with inverted summand	x1 $+$ *uimm*	x1 $+$ x2
00101	Difference with inverted summand	$\tilde{x1} - uimm$	$\tilde{x1} - x2$
01000	Unsigned product	x1 $*$ *uimm*	x1 $*$ x2
01001	Signed product	x1 $*$ *simm*	x1 $*$ x2
01010	Reserved		
01011	Reserved		
01100	Logic AND	x1 $\wedge$ *simm*	x1 $\wedge$ x2
01101	Logical OR	x1 $\vee$ *uimm*	x1 $\vee$ x2
01110	Logical XOR	x1 $\oplus$ *simm*	x1 $\oplus$ x2
01111	Reserved		
10000	Logic right shift	x1 $\gg$ *uimm*	x2 $\gg$ x1
10001	Logic left shift	x1 $\ll$ *uimm*	x2 $\ll$ x1
10010	Arithmetic right shift	x1 $\gg$ *uimm*	x2 $\gg$ x1
10011	Reserved		
10101	Reserved		
10101	Reserved		
10110	Reserved		
10111	Reserved		
11000	Set upper bits of an immediate value	*simm*, x1[11:0]	*simm*, x2[11:0]
11001	Reserved		
11010	Reserved		
11011	Reserved		
11000	Current interrupt vector	Vector address	
11000	Current error code	throw code	
10000	Parameter stack status	IPS:UPS	
10000	Return stack status	IPS:UPS	

3.7   Stack Instructions

The N1's stack instruction aims at efficiently implementing the essential stack operations in Forth only using the data pathes which needed for the stack's push and pull operations.

The opcode of the stack instruction contains a 10-bit wide field to specify a transition pattern of the upper cells of the parameter stack and the return stack. The structure transition patter is shown in Figure 3-2.

The stack instruction contains four UST fields which control the data transfer within the upper four cells of the parameter stack and the top of the return stack. Each UST field determines the direction of data transfer between two neighboring stack cells. Four options are selectable:
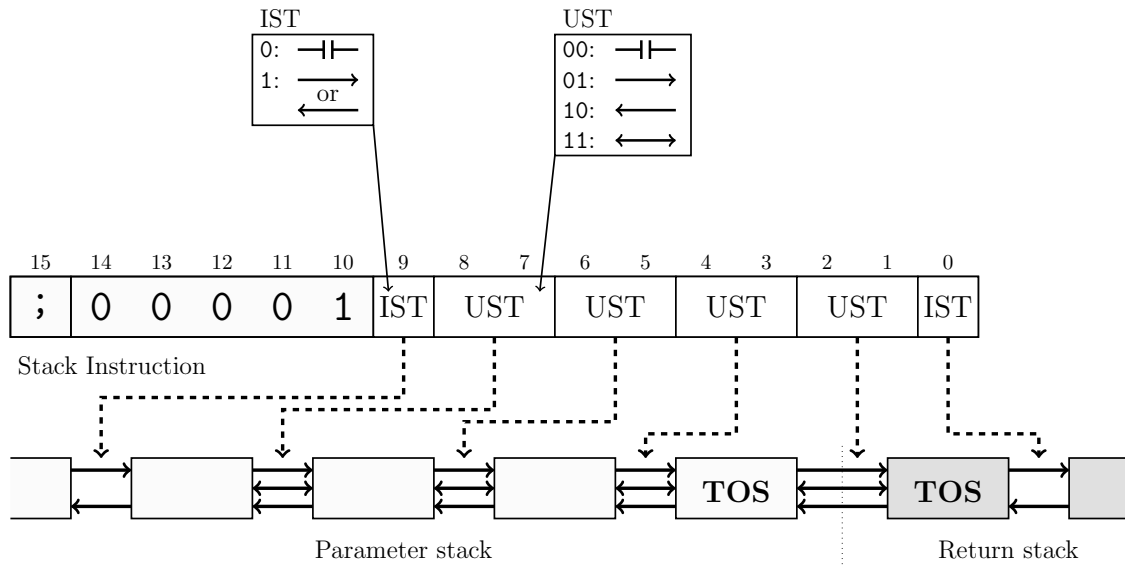
Figure 3-2: Transition encoding of stack instructions

- No data transfer

- Data transfer upwards (or towards the return stack)

- Data transfer downwards (or towards the parameter stack)

- Data exchange between two stack cells

It is possible to put the UST fields into a combination which would trigger a data transfer of two source cells to a single desination cell. In these cases, the resulting data in the desination cell is undefined.

The two remaining IST fields in the stack instruction control the data movement of the lower stacks. Two options are selectable:

- No data transfer

- Data shift throughout the entire intermediate stack. The direction is determined by the data movement of the lowest cell of the upper stack.
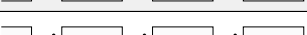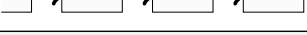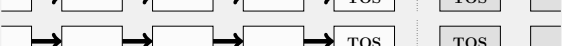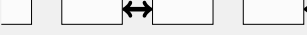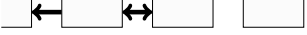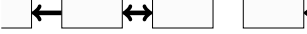
Table 3-2 shows how stack operations in Forth are mapped N1 instructions.

Table 3-2: Common stack operations

Word	Description	Transitions						Opcode
DROP	(x −)					TOS	TOS	0x06A8
DUP	(x − x x)					TOS	TOS	0x0750
SWAP	(x1 x2 − x2 x1)					TOS	TOS	0x0418
OVER	(x1 x2 − x1 x2 x1)					TOS	TOS	0x0758

...continued

11

Table 3-2: Common stack operations

Word	Description	Transitions	Opcode
NIP	(x1 x2 – x2)		0x06A0
TUCK	(x1 x2 – x2 x1 x2)		0x0750 0x0460
ROT	(x1 x2 x3 – x2 x3 x1)		0x0460 0x0418
-ROT	(x1 x2 x3 – x3 x1 x2)		0x0418 0x0460
RDROP	(R: x –)		0x0001
RDUP	(R: x – x x)		0x0007 0x0006
>R	(x –) (R: – x)		0x06AB
R@	(– x) (R: x – x)		0x0754
R>	(– x) (R: x –)		0x0755
2DROP	(x1 x2 –)		0x06A8 0x06A8
2DUP	(x1 x2 – x1 x2 x1 x2)		0x0758 0x0758
2SWAP	(x1 x2 x3 x4 – x4 x3 x1 x2)		0x0460 0x0598 0x0460
2OVER	(x1 x2 x3 x4 – x1 x2 x3 x4 x1 x2)		0x0780 0x0460 0x0798 0x0460
2NIP	(x1 x2 x3 x4 – x3 x4)		0x06A0 0x06A0

Table 3-2: Common stack operations

Word	Description	Transitions	Opcode
2TUCK	(x1 x2 x3 x4 – x3 x4 x1 x2 x3 x4)		0x046B 0x0487 0x0418 0x0460 0x0755 0x0755
2ROT	(x1 x2 x3 x4 x5 x6 – x3 x4 x5 x6 x1 x2)		0x06AB 0x0580 0x06AB 0x0598 0x0755 0x0598 0x0755 0x0598 0x0460
-2ROT	(x1 x2 x3 x4 x5 x6 – x5 x6 x1 x2 x3 x4)		0x0460 0x0598 0x06AB 0x0598 0x06AB 0x0598 0x0755 0x0018 0x0755
2RDROP	(R: x1 x2 –)		0x0001 0x0001
2RDUP	(R: x1 x2 – x1 x1 x1 x2)		0x0755 0x0757 0x06AB 0x06AB
2>R	(x1 x2 –) (R: – x1 x2)		0x0000 0x0000

13

Table 3-2: Common stack operations

Word	Description	Transitions	Opcode
2R@	(– x1 x2) (R: x1 x2 – x1 x2)		0x0000 0x0000
2R>	(– x1 x2) (R: x1 x2 –)		0x0000 0x0000

3.8   Memory Access Instructions

Memory access instruction perform read or write acesses to the system's 64-Kbyte address space. Data can be accessed in word or byte entities. Misaligned word accesses are not supported. Word accesses to a 510-Kbyte subset of the address space can be done through an immediate addressing. This will offer faster access to frequently used system variables.

3.9   Control Instructions

The N1 implements of set of instrictions to controls some of its internal components. None of these instructions consume input arguments from the parameter stack, nor do they produce a return value. The encoding of these instructions is shown in Table 3-3. Multiple control instructions can be combined to one.

Table 3-3: Control instructions

Encoding	Instruction
xxxxxx11	Enable interrupts
xxxxxx10	Disable interrupts
xxxxx1xx	Reset parameter stack
xxxx1xxx	Reset return stack

4   ANS Forth Words

Table 4-1 provides a list of standard ANS Forth words (see [**?**]) which directly map to hardware instructions of the N1 processor.

Table 4-1: ALU operations

Word	Stack	Description	Opcode
!	(x a-addr –)	Store cell	0000
*	(n1|u1 n2|u2 – n3|u3)	Multiply two cells	0000
+	(n1|u1 n2|u2 – n3|u3)	Add two cells	0000
-	(n1|u1 n2|u2 – n3|u3)	Subtract a cell from another.	0000

5 Stacks

The N1 operates with two stacks: the parameter stack to perform data transactions and the return stack to manage the program flow. As illustrated in Figure 5-1, each of these stacks consists of three hardware components: the upper stack, the intermediate stack, and the lower stack.
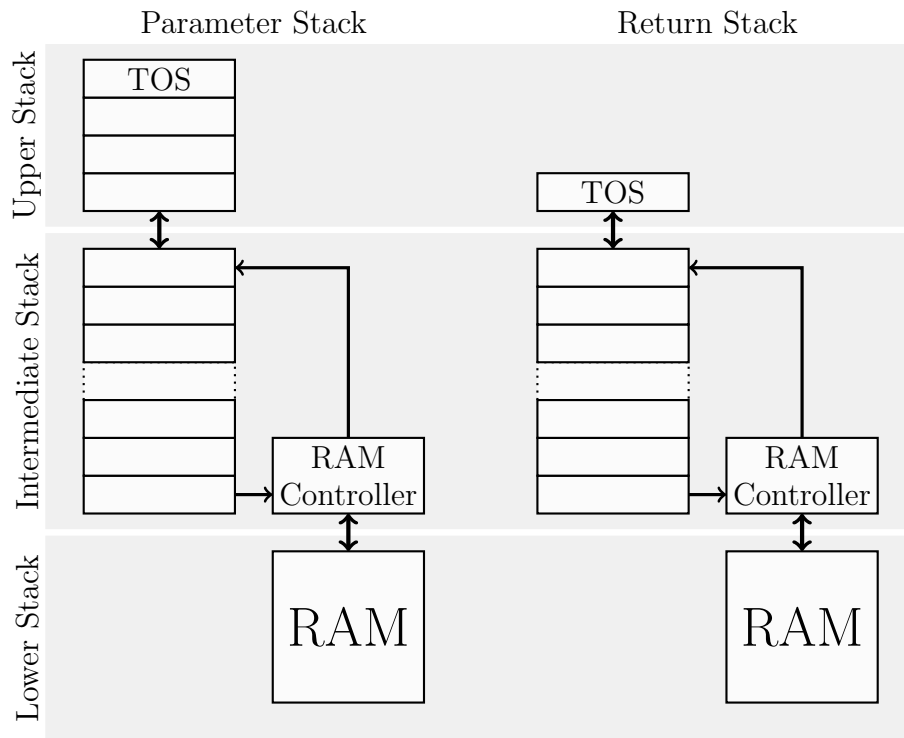
Figure 5-1: Stack Architecture

5.1 Parameter Stack

The upper stack of the parameter stack contains is four cells deep and contains the most recent data entries. It's purpose is to perform stack and ALU operations (see Section 3.7 "Stack Instructions" and Section 3.6 "ALU Instructions"). When the capacity of the upper stack is exceeded, older data entries are transferred to the intermediate stack.

The intermediate stack serves as a buffer between the upper stack and the lower stack which resides in RAM. The purpose of the intermediate stack is to minimize RAM traffic to and from the lower stack. Push operation to the intermediate stack are only propagated to the lower stack, when the buffer capacity is exceeded. Pull operations are onle propagated, when the intermediate stack is empty. Stack fluctuations within the buffer capacity are not visible to the lower stack.

The lower stack is a region of the RAM, which is managed by the memory controller of the intermediate stack.

5.2 Return Stack Stack

The upper stack of the parameter stack has the capacity of one cell. The intermediate stack and lower stack are similar to the ones of the parameter stack.

References

[1] Menlo Park James Bowman, Willow Garage. J1: a small forth cpu core for fpgas. http://www.excamera.com/files/j1.pdf, 2010.