The Harris RTX 2000 Microcontroller

Tom Hand
Senior Scientist
Harris Semiconductor
Melbourne, Florida 32902

Abstract

Harris Semiconductor has developed the RTX 2000, a highly integrated 10MHz 16-bit microcontroller, for embedded applications that have demanding real-time requirements.

The RTX 2000 is a high-performance stack machine that has many of the advantages of RISC processors, but without their disadvantages. The RTX 2000 has a predictable run-time behavior because of its advanced, yet simple, architectural features.

Introduction

The RTX 2000 is a highly integrated 16-bit microcontroller from Harris Semiconductor that has been specifically designed for solving problems associated with embedded real-time systems. It has a predictable run-time behavior, a key requirement of real-time systems.

The RTX 2000 is derived from earlier generation Novix stack machines. Novix Inc. of Cupertino, CA first developed the Novix NC 4016 stack machine that directly executed 40 Forth primitives as well as 123 combinations of Forth words as single instructions. The NC 4016 chip used only 4000 gates that were built from just 16,000 CMOS transistors. Harris bought complete rights to the Novix technology and in addition added on-chip stacks and other on-chip support such as counter/timers, interrupt controllers, and a single-cycle multiplier.

The architecture of the RTX 2000 encourages programming in a structured manner. To accomplish this, subroutine calls have been implemented in such a way that they execute in one machine cycle. More remarkably, subroutine returns are normally free; that is, they take zero clock cycles to execute. As a result, application code is both extremely compact and fast.

The RTX 2000 is a stack machine; its machine language corresponds to certain sequences of Forth instructions. Because four internal buses may be active at the same time, it is often possible to execute the equivalent of several Forth instructions in a single machine cycle. This guarantees that application code executes very quickly.

This article first briefly compares the RTX 2000 with RISC processors. Next, the RTX 2000 architecture is examined by discussing its basic buses and registers. Third, the instruction set of the RTX 2000 is studied in detail. Finally, additional features of the RTX 2000 are discussed; these include interrupts, byte swapping and the single-cycle multiplier.

RTX 2000 Versus RISC Machines

The RTX 2000 incorporates most of the advantages of RISC architectures, but without their disadvantages.

RISC (Reduced Instruction Set Computers) machines have most of the characteristics listed below:

- a large set of registers
- a simple set of instructions
- execution of most instructions in a single cycle
- equal lengths of all instructions
- very few addressing modes
- no instructions that manipulate or modify the contents of memory
- no microcode
- utilization of pipelines and caches

Not every RISC machine has all of these characteristics, but most do have a significant number of them.

The RTX 2000 has most of the attributes listed above; the two items that really distinguish the RTX 2000 from the RISC machines are the first and last items. Because RISC machines have register-based architectures, a degree of complexity is added to their architectures to get higher performance. Instructions are divided into components and pipelines are introduced to make these components execute as stages in parallel. Typical pipeline stages are instruction fetch, instruction decode, instruction execute and instruction write-out. Partly because of pipeline stalls and delayed branches, caches are introduced. The resulting RISC architectures have better performance than they had before the complexity was added, but at a definite cost.

In contrast, the RTX 2000 has a stack-based architecture so that operands are implicitly known. This subtle idea is what makes stack machines simple. RISC machines, on the other hand, require explicitly referenced operands and this leads to extra overhead, both in compiling by optimizing compilers and in the size of memory required for the final compiled code.

The Architecture of the RTX 2000

The RTX 2000 is a highly integrated 16-bit microcontroller with three on-chip counter/timers, an on-chip interrupt controller, two on-chip stack controllers and single cycle 16-by-16 on-chip hardware multiplier. Figure 1 is a block diagram that illustrates the basic architecture of the RTX 2000.

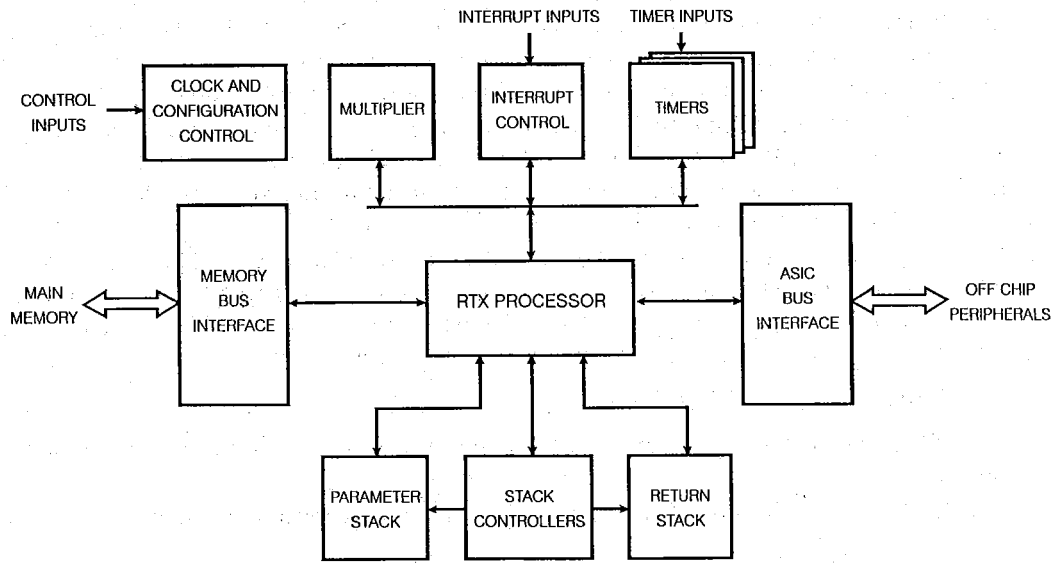

Figure 1: RTX 2000 Architecture

At the center of the figure is the RTX core processor, which is the stack machine. Other portions of the RTX 2000 are connected to the core processor through one of the RTX 2000's four buses listed below:

- parameter stack bus
- return stack bus
- ASIC bus
- memory bus

As a stack machine, the RTX 2000 has two on-chip stacks that are called the parameter and return stacks. The parameter stack is used for passing arguments and manipulating data, the return stack is used for return addresses and looping arguments. Having the stacks on-chip significantly improves the performance of the RTX 2000. The parameter and return stacks are connected to the core processor through separate parameter stack and return stack buses.

The ASIC bus is a high speed bus that connects the RTX 2000 to on-chip registers and devices as well as to external I/O devices. Each device attached to the ASIC bus is assigned a specific I/O address. The RTX 2000 can address 32 ASIC devices.

The processor's control and status registers are connected to the ASIC bus. ASIC addresses 0 through 23 are assigned to on-chip registers and devices. This includes the program counter, the square root register, the configuration register, the data page register, the code page register, the user page register, the user base register and the interrupt mask register.

In addition, the two on-chip stack controllers, the on-chip interrupt controller, the three on-chip 16-bit counter/timers and the 16-by-16 single cycle multiplier are all accessible through the ASIC bus via ASIC bus read and write instructions.

Again, one of the advanced features of the RTX 2000 is its high speed ASIC bus. This bus provides a natural extension to both internal and external devices.

Up to 512K words of memory may be addressed by the RTX 2000 through the memory bus. Memory is divided into sixteen pages, each containing 32K words of memory. Memory may consist of combinations of RAM, ROM or memory mapped I/O devices.

RTX 2000 Internal Registers

There are eight 16-bit registers which are internal to the processor. A brief description of these registers is given below:

TOP contains the top item of the parameter stack. It is called the Top Register.

NEXT contains the second item of the parameter stack. It is called the Next Register.

IR contains the instruction currently being executed. It is called the Instruction Register.

PC contains the address of the next instruction to be fetched from main memory. It is called the Program Counter.

CR contains bits that indicate the status of the RTX 2000. It is called the Configuration Register.

I contains the top item of the return stack. It is called the Index Register.

MD normally contains the divisor during multi-step math operations. It is called the Multi-step Divide Register.

SR normally contains intermediate values used during square root calculations. It is called the Square Root Register.

The RTX 2000 Instruction Set

The RTX 2000 is a 16-bit machine. All of its instructions are 16 bits wide, with the exception of long literals that take 16 bits for the instruction and 16 bits for the actual literal value.

RTX 2000 instructions execute in either one or two machine cycles. All primitive Forth words which do not perform memory accesses execute in one clock cycle. Memory access instructions require two cycles.

Most math, I/O and memory reference operations take their operands from the parameter stack and leave their results on the parameter stack.

Instruction Format

The general format for the RTX 2000 instructions is given below:

Field	Bits	Contents
Class	12 – 15	type of instruction
ALU	8 – 11	ALU function to be performed
SC	6 – 7	subclass, depends on the class field
;	5	return bit, causes a return when set
Data	0 – 4	indicates shift, short literal, ASIC bus address or memory address

Instruction Class

The four most significant bits of each instruction indicate the class type of that instruction. There are eight general types of instructions as illustrated below:

Class	Operation
0 – 7	Subroutine call
8 – 9	Branches and loops
10	Math/logic functions
11	Register and short literal operations
12	User memory access
13	Long literals
14	Memory access by word
15	Memory access by byte

Subroutine Calls

As mentioned earlier, the RTX 2000 is optimized for writing modular applications. Subroutine calls within the same memory page are performed in one clock cycle; a call to a subroutine in a different memory page takes three clock cycles. Returns take zero clock cycles when they are performed as part of another instruction and one cycle when they are performed as separate instructions.

If an instruction has bit 15 set to 0, it is a subroutine call. The format for the subroutine call is

Øaaa aaaa àaaa aaaa

where the address of the subroutine is **aaaa aaaa aaaa aaaØ**, which is calculated by shifting the low-order fifteen bits to the left by one bit and inserting a 0 in the least significant bit. Note that the address of the subroutine called is embedded in the 16-bit call instruction.

Branch and Loops

The **ØBRANCH** instruction belongs to class 8, while the **BRANCH** and **NEXT** instructions belong to class 9. These instructions perform conditional and unconditional branches, respectively.

All branches take one cycle, independent of whether or not the branch is actually taken. This is in contrast to many RISC and CISC processors which take a variable number of cycles depending on whether or not a branch is taken.

For high speed looping, the **NEXT** form of the conditional branch instruction may be used. With this instruction, a count is put in the Index register to indicate how many times the loop is to be performed. The **NEXT** instruction tests the contents of the Index register at the conclusion of each pass through the loop. If the contents of the Index register are 0, the return stack is popped and execution continues with the instruction that immediately follows **NEXT**; otherwise, the contents of the Index register are decremented by one and a branch back to the start of the loop is taken.

The format for this class of instructions is

1ØØc cbba aaaa aaaa

where the two bits 'cc' specify the condition for branching as indicated below:

cc	Branch condition
00	Branch if the contents of TOP is 0. Leave the stack unchanged.
01	Branch if the contents of TOP is 0. Pop the stack.
10	Perform an unconditional branch.
11	Branch if the contents of the INDEX register <> 0.

and the two bits 'bb' determine the block selection as indicated below:

bb	Block selection
00	Branch within the same memory block (no change to bits 10 – 15 of the instruction).
01	Branch to the next memory block (add 1 to the value in bits 10 – 15).
10	Branch to block 0 (set each bit in 10 – 15 to a value of 0).
11	Branch to the previous block (subtract one from the value in bits 10 – 15).

The value 'aaaaaaaaa' indicates the offset from the start of the new block. This value replaces bits 1 – 9 of the Program Counter. Branch addresses are always even numbers.

The RTX 2000 has a streamed instruction capability that can execute repeatedly an instruction without continually performing the fetch cycle of the instruction. This feature is very useful for fast data transfers, loops and certain math functions.

ALU Operations

The ALU operations are performed by the RTX's 16-bit ALU, one operand being the contents of the TOP register and the other being determined by the instruction. The result of an ALU operation is placed in the TOP register.

The twelve possible ALU operations are briefly described below. 'T' indicates the TOP register and 'Y' indicates the second source of the ALU operation.

cccc	aaa	Function	Resulting carry
0010	001	T AND Y	no change
0011		T NOR Y	no change
0100	010	T−Y	ALU carry
0101		T−Y with borrow	ALU carry
0110	011	T OR Y	no change
0111		T NAND Y	no change
1000	100	T + Y	ALU carry
1001		T + Y with carry	ALU carry
1010	101	T XOR Y	no change
1011		T XNOR Y	no change
1100	110	Y−T	ALU carry
1101		Y−T with borrow	ALU carry

A literal is a constant value that may be used as part of either an arithmetic or a logical operation. There are two types of literals that are recognized by the RTX 2000; namely, short literals and long literals. Short literals are stored as five bits whereas long literals take sixteen bits.

Short Literals

Short literals are 5-bit unsigned integer values between 0 and 31. They are embedded in short literal instructions and are denoted by 'ddddd'. The format for the short literal instructions is

```
1Ø11 vvvv v1;d dddd
```

where the 'v' bits determine the particular variation of the short literal instruction. For the complete list of variations, see Table 1 which contains all the RTX 2000 instructions.

When a short literal instruction is executed, the value represented by 'ddddd' is loaded into the TOP register.

User Memory Access

User memory space consists of blocks of 32 words that can be accessed without having to first calculate an address and then load it into the TOP register. The location of the user memory space can be specified by the user by loading a desired base address into the user base register. The offset within the 32-word block is embedded in the user access instruction. Therefore, this class of instructions perform fast reads and writes from and to user memory space.

The offset is encoded as a 5-bit field in the instruction and is denoted by 'uuuuu'. The format for the user memory access instructions is

```
11ØØ vvvv vØ;u uuuu
```

where the 'v' bits determine the particular variation of the user memory access instruction. For the complete list of variations, see Table 1.

Long Literals

A long literal value may be a signed or unsigned 16-bit integer. Long literals are fetched from memory, so that an additional clock cycle is necessary for the execution of an instruction that involves a long literal.

The long literal instructions generate 16-bit values. The format for the long literal instructions is

```
11Ø1 vvvv vØ;x xxxx  dddd dddd dddd dddd
```

where the 'v' bits determine the particular variation of the long literal instruction and the 'x' bits are don't care bits. For the complete list of variations, see Table 1. The value **dddd dddd dddd dddd** is the long literal value. Long literals are the only RTX 2000 instructions that occupy two words of memory.

Data Memory Access

The RTX 2000 can access data memory as either 16-bit words or as 8-bit bytes. The format for the data memory access instructions is

```
111s vvvv vv;v vvvv
```

where the 'v' bits determine the particular variation of the data memory access instruction and the 's' bit indicates the size of the operand:

s = 0, for 16-bit words,
s = 1, for 8-bit bytes.

The formats for both word and byte access instructions are the same. For the complete list of variations, see Table 1.

Subroutine Returns

Instructions, that are not call or branch instructions, that have the subroutine bit (bit 5) set execute a subroutine return. The format for the subroutine return instruction is

```
1vvv vvvv vv1v vvvv
```

where the 'v' bits determine the RTX 2000 instruction to be executed along with a return. See Table 1 for the many possible variations. Note that bit 5 cannot be used as a subroutine bit in subroutine call or branch instructions since it is one of the bits used to determine the branch address.

A definite architectural advantage is gained by dedicating a single bit for indicating a subroutine return. Other architectures require 16-bit or 32-bit instructions to accomplish the same thing. For example, on the RTX 2000, the optimized code corresponding to

```
SWAP DROP DUP ;
```

is A0A0 (hex), with the return bit set to 1; whereas, the optimized code for

```
SWAP DROP DUP
```

is A080 (hex), with the return bit set to 0. In these cases, the equivalent of three or four Forth instructions are executed in one machine cycle.

The RTX 2000 Instructions

Table 1 contains the specification of the complete RTX 2000 instruction set. Each type of instruction has been discussed in an earlier section. For reference, all instructions are placed here in one table.

Interrupts

The RTX 2000 has an on-chip interrupt controller with fourteen interrupt request inputs for servicing on-chip as well as external interrupts.

Thirteen of the interrupt requests are maskable while the remaining interrupt request is non-maskable. The interrupt controller samples the interrupt request inputs during each instruction, prioritizes the active requests and signals the processor of pending interrupt requests.

Subroutine Call		
Øaaa aaaa aaaa aaaa	call word address	aaa aaaa aaaa aaaa
Subroutine Call		
1xxx xxxx xx1x xxxx	return from subroutine	
Single Step Math Functions		
1Ø1Ø ØØØi ØØ;Ø ssss	{NOT} shift	
1Ø1Ø 111i ØØ;Ø ssss	DROP DUP {NOT} shift	
1Ø1Ø cccc ØØ;Ø ssss	OVER SWAP alu-op shift	
1Ø1Ø ØØØi Ø1;Ø ssss	SWAP DROP {NOT} shift	
1Ø1Ø 111i Ø1;Ø ssss	DROP {NOT} shift	
1Ø1Ø cccc Ø1;Ø ssss	alu-op shift	
1Ø1Ø ØØØi 1Ø;Ø ssss	SWAP DROP DUP {NOT} shift	
1Ø1Ø 111i 1Ø;Ø ssss	SWAP {NOT} shift	
1Ø1Ø cccc 1Ø;Ø ssss	SWAP OVER alu-op shift	
1Ø1Ø ØØØi 11;Ø ssss	DUP {NOT} shift	
1Ø1Ø 111i 11;Ø ssss	OVER {NOT} shift	
1Ø1Ø cccc 11;Ø ssss	OVER OVER alu-op shift	
Step Math Functions		
1Ø1Ø vvvv vvv1 vvvv		
Branch Functions		
1ØØØ Øbba aaaa aaaa	?DUP ØBRANCH	
1ØØØ 1bba aaaa aaaa	ØBRANCH	
1ØØ1 Øbba aaaa aaaa	BRANCH	
1ØØ1 1bba aaaa aaa	NEXT	
Register and I/O Access		
1Ø11 ØØØi ØØ;g gggg	g G@ DROP {NOT}	
1Ø11 111i ØØ;g gggg	g G@ {NOT}	
1Ø11 cccc 1Ø;g gggg	g G@ OVER alu-op	
1Ø11 ØØØi 1Ø;g gggg	DUP g G! {NOT}	
1Ø11 111i 1Ø;g gggg	g G! {NOT}	
1Ø11 cccc 1Ø;g gggg	g G@ SWAP alu-op	
Short Literal		
1Ø11 ØØØi x1;d dddd	d DROP {NOT}	
1Ø11 111i Ø1;d dddd	d {NOT}	
1Ø11 cccc Ø1;d dddd	d OVER alu-op	
1Ø11 111i 11;d dddd	d SWAP DROP {NOT}	
1Ø11 cccc 11;d dddd	d SWAP alu-op	
User Space	**1st cycle**	**2nd cycle**
11ØØ ØØØi ØØ;u uuuu	u @ SWAP	{NOT}
11ØØ 111i ØØ;u uuuu	u @ SWAP	SWAP {NOT}
11ØØ cccc ØØ;u uuuu	u @ SWAP	SWAP OVER alu-op
11ØØ ØØØi 1Ø;u uuuu	DUP u !	{NOT}
11ØØ 111i 1Ø;u uuuu	DUP u !	DROP {NOT}
11ØØ cccc 1Ø;u uuuu	u @ SWAP	alu-op
Long Literal	**1st cycle**	**2nd cycle**
1101 ØØØi xØ;x xxxx	D SWAP	{NOT}
1101 111i ØØ;x xxxx	D SWAP	SWAP {NOT}
1101 cccc ØØ;x xxxx	D SWAP	SWAP OVER alu-op
1101 111i 1Ø;x xxxx	D SWAP	DROP {NOT}
1101 cccc 1Ø;x xxxx	D SWAP	alu-op
Memory Access	**1st cycle**	**2nd cycle**
111s ØØØi ØØ;x xxxx	@ SWAP	{NOT}
111s 111i ØØ;x xxxx	@ SWAP	SWAP {NOT}
111s cccc ØØ;x xxxx	@ SWAP	SWAP OVER alu-op
111s ØØØp Ø1;x xxxx	{SWAP DROP} DUP @ SWAP	NOP
111s 111p Ø1;d dddd	{SWAP DROP} @ d	NOP
111s aaap Ø1;d dddd	{SWAP DROP} DUP @ SWAP d SWAP alu-op	NOP
111s ØØØi 1Ø;x xxxx	OVER SWAP !	{NOT}
111s 111i 1Ø;x xxxx	OVER SWAP !	DROP {NOT}
111s cccc 1Ø;x xxxx	@ SWAP	alu-op
111s ØØØp 11;x xxxx	{OVER SWAP} SWAP OVER !	NOP
111s 111p 11;d dddd	{OVER SWAP} ! d	NOP
111s aaap 11;d dddd	{OVER SWAP} SWAP OVER ! d SWAP alu-op	NOP

where s = 0, for memory access by word (**@** and **!**)
s = 1, for memory access by byte (**C@** and **C!**)
{SWAP DROP} and **{OVER SWAP}** are performed if p = 0

Table 1. RTX 2000 Instructions

Interrupts can be enabled or disabled by means of the interrupt disable bit in the processor's configuration register. Interrupts are disabled when this bit is set to 1 and enabled when it is set to 0.

The RTX 2000 has a single level software interrupt capability.

Byte Swapping

Interfacing with non-RTX processors is supported through shared memory. The RTX 2000 has a byte swapping feature that allows 16-bit values to be read and written so that the most significant byte can be associated with either an even or an odd address.

The Multiplier

The on-chip single cycle hardware multiplier of the RTX 2000 multiplies two 16-bit numbers yielding a 32-bit result in only one clock cycle. The two 16-bit operands can be treated as either signed or unsigned integers. In addition, the resulting product can optionally be rounded to 16 bits. As mentioned earlier, the multiplier is connected to the ASIC bus.

Conclusion

In summary, the RTX 2000 is a high performance highly integrated stack machine that has been designed for embedded real-time applications. Very compact and fast code results from its advanced, yet simple, architectural features.

References

[ASPL87] *Proceedings: Second International Conference on Architectural Support for Programming Languages and Operating Systems (ASPLOS II),* The Computer Society Press, IEEE, 1987.

[GLAS89] Glass, H., Mellen, M., Hand, T., "The Design of a Real-time Multi-tasking Operating System Kernel for the Harris RTX 2000," *Proceedings of the 1989 SIGForth Conference,* Austin, Texas, 1989.

[HAND89] Hand, T., "Architecture Impact on Compiler Construction," *Proceedings of the 1989 SIGForth Conference,* Austin, Texas, 1989.

[HAND89a] Hand, T., "Metrics for Real-time Systems," *Proceedings of the 1989 SIGForth Conference," Austin, Texas, 1989.*

[HARR88] "RTX Real Time Express: Reference Manual," Harris Semiconductor, Melbourne, Florida, 1988.

[HAYE88] Hayes, J., *Computer Architecture and Organization,* McGraw Hill, 1988.

[PATT85] Patterson, D., "Reduced Instruction Set Computers," *CACM* 28(1), Jan. 1985.

Dr. Tom Hand received his Ph.D. in Mathematics from the University of Oklahoma in 1972, and has been in the computer field for more than twenty years in industry and at the University. As Graduate Program Chairman in Computer Science at the Florida Institute of Technology he examined Forth as a vehicle for implementing compilers, expert systems, natural language front-ends and operating systems. Currently, he is a Senior Scientist at Harris Semiconductor and is involved with the development of the RTX family of microcontrollers.