SURVEY & TUTORIAL SERIES

A Survey of RISC Processors and Computers of the Mid-1980s

Charles E. Gimarc and Veljko M. Milutinović
Purdue University

Several years have elapsed since the early reduced computer architecture research was conducted at IBM, Stanford, and Berkeley. In this article, we briefly survey modern reduced instruction set computer architectures. A survey of this type helps illustrate the application areas into which RISC philosophy has moved, the wide range of technologies being used for processor implementation, features that seem to be common with all designs, and unique features introduced to solve specific problems. Finally, although this survey does not provide much depth on each architecture, we will provide interested readers with a more extensive list of references.

We briefly survey modern RISC architectures, illustrating common features, range of applications, implementation technologies, and some unique characteristics of today's RISC machines.

Definition of RISC

It seems to be difficult to provide a precise definition of a RISC architecture. The original reduced instruction set computer research, in which an attempt was made to reduce the semantic gap, produced a design philosophy that can be stated as follows[1,2]:

(1) Analyze target applications to determine which operations are used most frequently.

(2) Optimize the datapath design to execute these instructions as quickly as possible.

(3) Include other instructions only if they fit into the previously developed datapath, are relatively frequent, and their inclusion will not slow the execution of the more frequent instructions.

(4) Apply a similar strategy to the design of other processor resources. Include a resource only if it is justified by its frequency of use, and its inclusion will not slow other, more frequently used, resources.

(5) Push as much complexity as reasonable from runtime hardware into the compile-time software.

Within RISC design philosophy, one must be free to make tradeoffs across boundaries of architecture/implementation, hardware/software, and compile-time/runtime.

When a processor design is based upon RISC philosophy, the resulting architecture typically has features in common with other RISC designs. Inclusion or exclusion of particular features should not be used as a measure of a RISC design, however. As you will see in this survey, several computers that are definitely RISC designs have features generally not considered part of a RISC design. Some features commonly seen in reduced instruction set computers are[3,4,5]

- single cycle execution of most instructions,
- load/store instruction set,

0018-9162/87/0900-0059$01.00 ©1987 IEEE

- hardwired instruction decoding,
- relatively few instructions and address modes,
- fixed instruction format for simple decoding,
- complexity pushed into optimizing compiler,
- highly pipelined datapath for much concurrency,
- large register set (windowed or not windowed),
- many levels of memory hierarchy, and
- instruction set designed for a specific application class.

A brief study of rating RISC designs based upon a similar set of characteristics was done by Tabak[6] in 1986. This type of rating is useful, but must be carefully applied and interpreted. The danger in using a list such as this is that computers with features contrary to those listed may not be interpreted as RISC designs. For example, some of the surveyed computers have microcoded instruction decoders, large instruction sets, and small register sets, but are still definitely RISC designs. The important issue is that RISC philosophy is followed in the design of a processor for a specific application.

Controversy

Currently there is much controversy over several issues relating to comparison between reduced instruction set computers and complex instruction set computers. The controversy can be divided into two broad categories. First, what differentiates a RISC from a CISC? And second, how does one make reasonable and useful performance measurements for comparisons?

Many of the features now seen in reduced instruction set computers have been extensively used in complex instruction set computers. Some of the features, such as pipelined datapath, caching, and register windowing, are often viewed as attributes of a RISC design.[7] One advantage that RISC designs have is the existence of a coherent philosophy statement. Originally, RISC designs were targeted for a specific application and therefore optimized for execution of a well defined class of programs. Characteristically, CISCs are designed for a broad range of applications and consequently include support for many different programming environments. It is evident that there is now a trend away from the narrow RISC application and to a more general purpose design, again confusing the difference between RISC and CISC. RISC proponents often see instruction set size and complexity as a major feature of "RISCy" designs.

Many of the popular performance measurement techniques are of questionable value when comparing RISC performance with CISC performance. Typically, the effects of operating system overhead, compiler optimization, and multiple register sets are not properly considered. Benchmarks related to number of application transactions per second are more meaningful than simply measuring instructions executed per second.

What processors are out there?

Twenty-one processors are included in this survey. Table 1 presents a list of the surveyed processors with a brief description of each. It is apparent that RISC design has moved out of the university and into the marketplace. Fifteen processors are either currently available for purchase, or will be available this year. Three computers are part of continuing research at Stanford University and the University of California, Berkeley. Three computers are in the design stage and will become embedded signal processors for defense computers.

RISC designs are available as single-chip microprocessors, very large scale integration chip sets with more powerful functions, single-board computers, and superminicomputers. The ARM processor is marketed as VLSI Technology's VL86C010 microprocessor, indicating that some of these processors can appear under different names. A wide range of circuit technologies are used in their implementation. The Whetstone and AMD 29300 processors are bipolar VLSI devices with emitter coupled logic (ECL) internal cells and transistor-transistor logic (TTL) interfaces. Most designs are N-type metal oxide semiconductor (NMOS) or complementary metal oxide semiconductor (CMOS) implementations with geometries of one to three microns. The highest performance processors are being designed for gallium arsenide (GaAs) implementation. These processors have the longest time to their introduction as a product.

One of the typical characteristics of a RISC design is a certain narrowness of target application. Table 1 indicates a wide range of individual applications, plus some designs with a very broad application space. Many designs target scientific, engineering, computer-aided engineering (CAE), and computer-aided design (CAD) applications. Almost universally, these use some variant of a Unix operating system. Five of the designs—ARM, Dragon, FAIM-1, SOAR, and SPUR—target symbolic processing and artificial intelligence applications. Most of these are used in multiprocessor configurations. All three GaAs designs and the CAP target signal, data, and image processing applications. The GaAs processors have the advantage of raw speed. The CAP processor has the potential of massive parallelism. The Transputer and MIPS-X processors were designed with general purpose multiprocessing in mind. Four designs—AMD 29300, Spectrum, Ridge 32, and Whetstone—have a broad application target.

Characteristics of GaAs material require changes in architecture when compared to designs in silicon. Of these characteristics, the major design implications are higher off-chip/on-chip delay ratio and lower transistor count per chip. Each of the three GaAs processors address these issues in similar ways, with deep pipelines, processing functions spread over several chips, small instruction sets, and complex memory hierarchies.

This survey is not exhaustive for several reasons. Relatively low cost, high performance computers are an extremely competitive business arena, and many manufacturers will not release details of their current or future products. Several new products are scheduled for introduction in 1987 and will appear between the time of writing this article and its publication. Several manufacturers market products as RISC or RISC-like that we feel do not follow the RISC design philosophy closely enough for inclusion. Other processors are too early in their design phase and will have to be discussed another time, after their designs have stabilized. We included some designs that may not be RISC processors because they exhibit some important RISC features. The Defense Advanced Research Project Agency (DARPA) is funding research for the development of GaAs and CMOS RISC processors. These CMOS processors

Table 1. Description of RISC processors.

Processor	Manufacturer	Configuration	Technology	Application
Accel	Celerity Computing	Single-board CPU and workstation	Uses NCR 32000 MOS chip set	Multiuser scientific and general purpose with Unix support
ARM	Acorn Computers	Microprocessor and single-board computer	2 to 3 μ MOS, 25K transistors	Workstation for HLL programming and real time AI
AMD29300	Advanced Micro Devices	Chip set to construct custom computer	Bipolar LSI with ECL internal and TTL interface	Construct target architecture without custom VLSI
CAP	IT&T Advanced Technology Center	Slave multiprocessor array	1.2 to 3 μ CMOS, 120K to 600K transistors	Signal processing, image processing, and scientific calculations
Clipper	Fairchild	Single-board computer	3 chips of 2 μ CMOS, total of 836K transistors	Scientific programming in a Unix environment
CRISP	AT&T	Microprocessor	1.75 μ CMOS, 172K transistors	General purpose
Dragon	Xerox PARC	Multiprocessor workstation	2 μ CMOS	Symbolic processing, AI
FAIM-1	Schlumberger, Palo Alto	Multiprocessor workstation	Custom VLSI CMOS	Symbolic processing, AI
MIPS	MIPS Computer Systems	Microprocessor, chip set, and minicomputer	2 μ CMOS, 100K transistors	General purpose programming with Unix support
Pyramid 90X	Pyramid Technology	Superminicomputer	Schottky TTL	Scientific programming and graphics with Unix support
Ridge 32	Ridge Computers	Workstation and superminicomputer	STTL and MOS VLSI in multiple chips	Scientific programming and graphics with Unix support
ROMP	IBM	Microprocessor chip set	2 chips of 1.8 μ NMOS with a total of 111K transistors	Scientific and graphics workstation with Unix support
Spectrum (HP Precision Architecture)	Hewlett-Packard	CPU family	1.5 μ NMOS, 115K transistors	Scientific, business, and instrumentation computing with Unix support
Transputer	INMOS	Microprocessor family	2 μ NMOS, 250K transistors	Multiprocessor
Whetstone I, Whetstone II	Integrated Digital Products	Single-board computer, one-chip CPU	Bipolar VLSI with ECL internal and TTL interface	Plug-in replacement general purpose computer
MIPS-X	Stanford University	Microprocessor chip set	2 μ CMOS	Updated MIPS, multiprocessing
SOAR	University of California, Berkeley	Microprocessor and processor board	NMOS, 35K transistors	Smalltalk-80 programming system workstation
SPUR	University of California, Berkeley	Multiprocessor workstation	3 chips in 2 μ CMOS	Parallel processing research in Lisp environment
CDC GaAs	Control Data	Microprocessor chip set	3 chips of 10K gates each in HIIL GaAs	Embedded computers for signal processing
McD GaAs	McDonnell Douglas	Microprocessor chip set	E-JFET GaAs, 41K transistors in 2 chips	Embedded computers for signal processing
RCA GaAs	RCA and Purdue University	Microprocessor chip set	ED-MESFET GaAs	Embedded computers for signal processing

Table 2. RISC processors instruction set features.

Processor	Number of Instructions	Decoder	Instruction Length
Accel	142 integer, 126 floating point	Microcoded	90% are 16 bits, 10% are 32 or 80 bits
ARM	?	Programmable logic array	All 32 bits
AMD29000	Variable	Hardwired or microcoded	32 bits
CAP	33 general, 7 scientific, 9 SIMD support	Hardwired; vector units are either 16 or 32 bits	16 or 32 bits
Clipper	101 general, 61 macroinstructions	Hardwired with 2048-word macroinstruction ROM	Multiples of 16 bits
CRISP	33	Decoder unit	16, 64, and 96 bits
Dragon	approx. 150	Programmable logic array	8, 16, 24, or 40 bits
FAIM-1	64	Finite state machine	?
MIPS	?	Hardwired	All 32 bits
Pyramid	90	Microcoded	32, 64, or 96 bits
Ridge 32	170	Microcoded	16, 32, and 48 bits
ROMP	112	Hardwired with 256 words of microcode	16 or 32 packed to 32 bits
Spectrum	140	PLA with millicode	All 32 bits
Transputer	16	Programmable logic array	All 8 bits
Whetstone	18 basic, 181 total	Basic instructions have hardware decode	16 bits
MIPS-X	< 32	Hardwired, distributed	All 32 bits
SOAR	20	Hardwired	8, 16, and 24 bits
SPUR	28 general, 10 Lisp support, 25 floating point	Hardwired CPU and FPU	All 32 bits
CDC GaAs	29 CPU, 31 FCOP, 6 MMU	Hardwired	Multiple; depends on execution unit
McD GaAs	< 64	Hardwired	All 32 bits
RCA GaAs	< 64	Hardwired	32 and 64 bits

are not treated in this survey because they are still early in their development cycle.

Organization of this survey

Since this is a survey of modern RISC designs, we will not discuss the IBM 801, RISC I, RISC II, and SU-MIPS. Also, we included the early commercial products from Pyramid and Ridge only in the tables as a reference. All of these computers have been extensively treated in the literature. We will refer to the early research RISCs when such reference illustrates a common trait, change in design philosophy, or enhancement of a familiar design feature.

This article is organized into sections comparing groups of common features. We investigate the CPU design first, emphasizing instruction set, datapath, and memory system design. Next we investigate the RISC as a system, emphasizing multiple execution units, coprocessor support, multiprocessing, operating system support, language support, and family requirements. Finally, we briefly compare performance.

CPU issues

Instruction set. A tabulation of features related to the instruction set appears in Table 2. The size of instruction sets in this survey ranges from a minimum of 16 instructions for the Transputer to approximately 268 for the Accel with floating-point support. Even though the instruction set size and instruction length differ for all the surveyed RISC machines, they all use instruction formats that allow rapid decode through use of a consistent opcode field.

Accel designers chose not to limit the processor to too few commands. The instruction set includes 142 integer processor unit instructions and 126 floating-point unit instructions. Ninety percent of the instructions are 16 bits long, the rest are 32 or 80 bits long. This large instruction set is reduced in terms of format and register orientation.

The AMD 29332 ALU has a symmetrical and regular instruction set of byte-aligned and variable-length data manipulation instructions. A complete central processing unit instruction set is constructed out of the arithmetic logic unit instructions, with fields to manipulate registers, coprocessors, and other parts of the CPU. Instructions are limited to 32 bits since all datapaths support a 32-bit word.

The CAP instruction set can be divided into three subsets. Thirty-three instructions execute primarily in the scalar processor. Seven floating-point or scientific instructions are present, and nine instructions are devoted to control of the single instruction, multiple data (SIMD) processor array. Two formats are used, with the 32-bit format split between two vector processors. Vector processors are 16-bit machines.

Fairchild chose to split the Clipper instruction set into two parts, with 101 "simple" instructions forming the basic instruction set. These are all multiples of 16 bits in length. Complex or application-specific instructions may be added to this set through coding into the macroinstruction unit.

The CRISP 33-member instruction set contains instructions of 16, 64, and 96 bits in length. The first bit of each instruction determines length. A major departure from other RISC processors, this design uses memory-to-memory instructions that can execute in one cycle. Also, this instruction set provides direct support of multiplication and division.

Dragon allows variable-length byte-encoded instructions. Consequently, the decode logic is quite complex.

Most of the 64 instructions of the FAIM-1 processor can execute in a single cycle.

Like the earlier RISC designs, and in contrast to most modern RISC designs, MIPS has fixed-length instructions, a single address mode, and all instructions 32 bits long. It uses three instruction formats. Like the MIPS processor, Stanford's MIPS-X implements its full instruction set in only 32-bit words. A five-bit opcode field allows only 32 instructions. Multiplies and divides are performed iteratively. It uses only one address mode and four instruction formats.

Most ROMP instructions (79) are 16 bits wide, and the remaining 39 fit into a 32-bit word. Some instructions have both two-byte and four-byte versions depending upon address modes. All instructions are packed into 32 bits upon fetch by the ROMP CPU.

In the Transputer, single-byte instructions decouple instruction format from the word length of the machine. There are several 16- and 32-bit Transputer models. All instructions are byte aligned with a four-bit opcode field.

To allow compatibility with an existing user base, the Whetstone simulates the Data General Nova instruction set. The manufacturer asserts that the processor uses 18 basic instructions, a subset of the Nova instruction set. All 181 Nova instructions are then mapped into this set of 18, presumably by the compiler. Instructions and data are in 16-bit words.

Each SOAR instruction contains a bit that enables or disables tag checking, allowing conventional programming languages to compile to SOAR instructions. Three instruction lengths (one, two, and three bytes) are used in the instruction set.

All SPUR instructions fit into a 32-bit word, with opcode and register fields aligned. The 55 instructions can be divided into general operations, Lisp support, and floating-point support groups.

The CDC processor instruction set may be divided into three subsets, one for the central processing unit, one for the floating-point coprocessor, and one for the memory management unit. Length of an instruction depends upon where it is executed.

A six-bit opcode field is used for both the McD and RCA processors' instructions. All instructions in the McD processor are 32 bits long, while the RCA processor allows some 64-bit instructions.

In this survey are two basic instruction decoder designs and several interesting combinations. Recall that the original RISC examples all used hardwired logic decoders for the simplest and fastest possible instruction decode. Most of the example processors (12) employ some form of strictly hardware instruction decode. A hardwired decoder, as in CAP, MIPS, MIPS-X, SOAR, SPUR, CDC, and RCA, implies that an area of the CPU chip is designed with minimized logic to decode the instruction set. The MIPS-X processor uses a distributed design with partial decode at the register in which the fetched instruction is placed, and the remainder of the decode at the functional elements in the datapath. The McDonnell Douglas references did not specify instruction decoder design, but in view of the time required to decode, a hardwired decoder seems the appropriate choice. The FAIM-1 employs a finite state machine controlled by instruction opcodes as the method of decode. Three examples (ARM, Dragon, and Transputer) employ programmable logic arrays in the decode path. Use of a PLA is effectively the same as hardwired decode, but does not use minimized logic design. Also, it is much simpler to reprogram a PLA for instruction set modifications than to redesign a hardwired decoder.

Two processors, Accel and Ridge 32, use microcoded instruction decoders and also have the largest instruction sets. The Accel design is built around an NCR32000 CPU and support devices. The CPU is microcodable, allowing specification of a rich instruction set with RISC attributes. The Ridge processors are microcoded, with most instructions executed by a one-instruction microsequence.

The AMD 29300 may be used to implement a RISC CPU with either hardwired or microcoded instruction decode. It requires only the proper control signals for the ALU, registers, coprocessor, and other units in the proper time sequence.

Four processors—Clipper, ROMP, Spectrum, and Whetstone—use a combination of hardwired and microcoded instruction decode techniques. The Clipper has hardwired decode of 101 instructions. A 2048-word macroinstruction ROM is provided, currently with 61 instructions, to allow support of specific programming environments. More macroinstructions may be added by the user. ROMP uses hardwired decode for instruction prefetch and memory data requests. A 256-word microprogram store is used for execution control. Hewlett-Packard's Spectrum processors use hardwired decode for most instructions and a millicode store in virtual memory for implementation of complex instructions. The Whetstone processors use hardwired decode for the basic 18 instructions, with microcode interpretation of the 181-instruction Nova 1200 instruction set.

The CRISP processor employs a separate Prefetch and Decode Unit to break each instruction into fully decoded 192-bit words. These decoded instructions are then loaded into a Decoded Instruction Cache for access by the Execution Unit. The FAIM-1 processor employs a separate Instruction Stream Memory to select, format, and deliver decoded instructions to the Evaluation Processor Unit.

Table 3. RISC processors datapath features.

Processor	Pipeline Stages	Register Set	Delayed Branch	Execution Units
Accel	3	48 32-bit register windows with overlap of 16, 512 windows	Yes, with delay of 2	Separate integer and floating point units
ARM	3	32 32-bit, 25 for user	No, compiler rescheduling	1 plus fetch incrementer
AMD29300	2	Variable	Yes, plus reorganizer	2: execute and fetch/decode
CAP	?	16 per processor	Send both options, do according to data mask	Scalar unit plus vector data and address units
Clipper	3	16 user, 16 special, 16 supervisor, all 32-bit	?	2: integer and floating point, all on chip
CRISP	3	32 32-bit, as a stack cache	Branch folding	2: prefetch and decode, and execution units
Dragon	?	?	Static branch prediction	2: instruction fetch, execute
FAIM-1	2	Stack	Yes	2: evaluation and switching processors
MIPS	5 (see Figure 1)	32 general purpose, 1 PC, 2 arithmetic, no windows	Yes, with reordering	2: CPU and system coprocessor. Separate data and address units
Pyramid	3	528 32-bit in 16 windows of 64	Yes	Separate instruction fetch and execution units
Ridge 32	4	16 32-bit with overlap windows	Branch prediction	2: instruction fetch, execute
ROMP	3	16 32-bit, 10 system, 4-port register file	Yes, branch with execute instruction	1
Spectrum	5	32 32-bit general purpose	Yes, with reorganization	1 CPU plus coprocessors
Transputer	?	6 32-bit	?	2: execute, I/O
Whetstone	3	4	?	3: arithmetic, address, I/O
MIPS-X	5	32 32-bit general purpose	Yes, with squash instruction, delay of 2	Separate execution and PC
SOAR	3	72 32-bit dual port with 32 register windows, using overlap of 8	Yes, with delay of 1 cycle. Must do type checking.	1
SPUR	4	32 32-bit registers in each window, using overlap of 6 and 10 global	Yes, with cancel compare instruction	1, but instruction and data fetches concurrently
CDC GaAs	6	16 32-bit general purpose	Yes, with reorganization and delay of 2	1
McD GaAs	4	16 32-bit general purpose, 16 32-bit special	Yes, with reorganization and delay of 1	1
RCA GaAs	5 + 2 waits	16 with variable size windows or background loading	Yes, with ignore instruction and delay of 2	Separate execution and PC units

Datapath. Datapath design is quite complex in all the surveyed processors. All 21 examples have pipelines of depth ranging from two stages in the AMD 29300 to seven stages (five plus two wait stages) in the proposed RCA GaAs processor. Generally, the shorter the cycle time, the deeper the pipeline. This phenomenon is primarily due to two factors. First, all example processors are designed to attempt to begin a new instruction each clock cycle. The MIPS processor can begin any instruction each clock cycle. Most other processors have some subset of instructions that require use of the pipeline for more than one cycle. Second, the processors have memory access and wiring delays that make up a large fraction of the clock cycle period. Thus, more cycles must be allocated for memory access instructions. Table 3 presents some important

datapath features for the surveyed processors.

The Pipeline Stages column of Table 3 gives the number of CPU pipeline stages, but not details about how they overlap on successive instructions, nor the function of each stage. (Those details are available in the references.) The MIPS processor pipeline deserves special attention because of its complexity. Figure 1 illustrates the pipe for four instructions. It is difficult to give a number-of-stages figure for this design because there are half- and full-cycle stages, and concurrent activities within each instruction execution. The clock cycle is 60 nanoseconds divided into two 30-nanosecond phases. Complexity is partially due to the requirement to execute all instructions in one cycle.

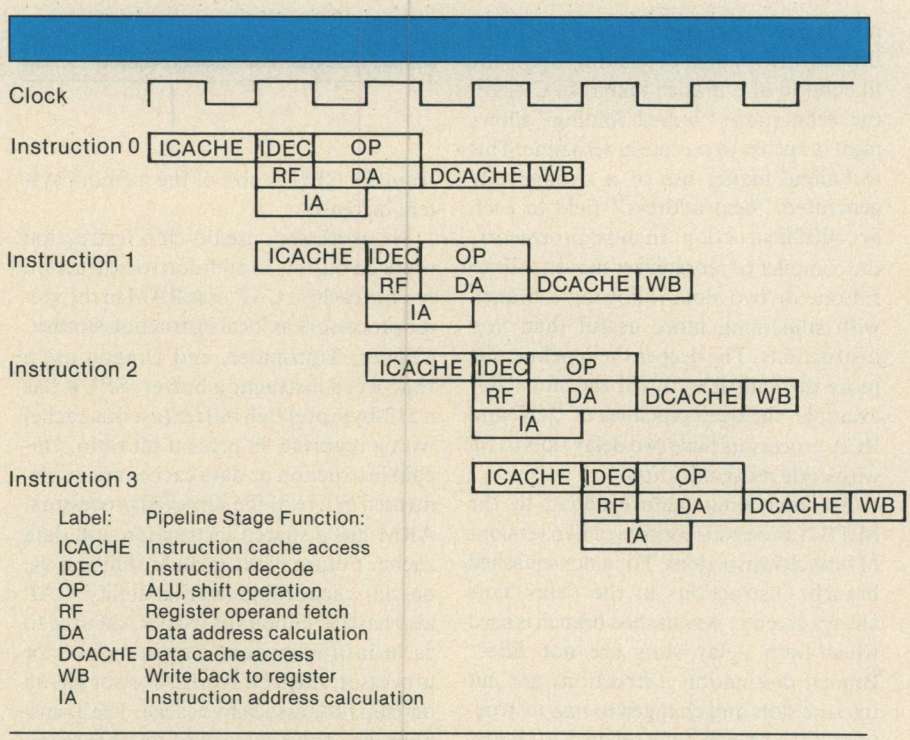


Figure 1. MIPS computer systems pipeline.

Several register set designs were used, with most processors using an organization with 16 to 32 32-bit registers. All the surveyed RISC designs attempt to gain performance advantages through (1) the speed of an on-chip variable store and (2) the compiler's ability to effectively use locality of program variables. Both windowed and unwindowed register files occur. Thirteen processors have register files with no windowing. Several nonwindowed designs, and even windowed designs such as SOAR, use *load multiple* or *store multiple* instructions to speed context switches. Table 3 gives window size details of the five windowed examples.

The ARM processor resembles the Pyramid design in the sense that it employs an extremely large number of registers. A few processors, such as the ROMP and AMD 29300, have four-port register files, allowing multiple reads and writes in a single clock cycle. The AMD allows expansion of the register file in length and width by adding more register chips. The Transputer has an extremely small register file. Even though only three registers are available for ALU operations, the Transputer maintains a workspace pointer in the register stack pointing to the first variable in memory. A delay of one cycle is thus incurred at each variable access. MIPS-X maintains 33 bits on each variable in the register file. The 33rd bit allows a read and write to be performed in a single cycle. The CRISP and FAIM-1 processors use a stack-based register file organization. In CRISP, a separate stack cache appears to the compiler as memory, is arranged as a circular buffer, and allows two reads and one write per cycle. Processors using stacks do not have the register allocation problems that other RISC processors' compilers have.

Most of the example processors divide CPU tasks into subtasks for separate execution units. A greater degree of concurrency is allowed at the expense of increased chip resource requirements and more complex control. Twelve processors (ARM, AMD, CAP, CRISP, Dragon, MIPS, Pyramid, Ridge, Transputer, Whetstone, MIPS-X, and RCA) have separate units for concurrent ALU execution and I/O operations. In most processors the I/O unit includes a separate ALU for address computation. The CAP divides execution into a scalar unit and separate vector data and vector address units. The Transputer I/O units are unique in providing four bidirectional links to other Transputers. The I/O unit coordinates the queueing of transmitted and received messages. The Whetstone divides tasks even further with separate address and I/O units. The SPUR processor is unique in performing independent and concurrent data and instruction fetches. Two logically separate units (prefetch and decode, and execution) in the CRISP are connected by a decoded instruction cache. This organization allows each unit to operate autonomously, with no central controller. Many processors use coprocessors to perform complex operations such as caching, memory management, and floating-point computations.

Several different shifter designs are included in the datapath of RISC processors. SPUR uses a simple unit that allows one-bit right or one-, two-, and three-bit left shifts. Due to timing constraints, RCA requires two instructions to perform a shift. The first instruction loads data into the shifter. A second instruction executes the shift and writes back results. Other processors use barrel or funnel shifters.

As in the original Stanford MIPS, some of the surveyed processors do not have hardware to manage pipeline interlocks. The processors with stated software interlocks are MIPS-X, MIPS, McD, RCA, and Spectrum. Of these, all but Spectrum appeared to be influenced by the SU-MIPS design.

Three processors (ARM, Clipper, and CDC) followed the Berkeley lead by providing hardware interlocks. Most processors provide register bypass paths to allow availability of a result of a computation for the next instruction.

Most processors use delayed branching to accommodate the condition when a branch is not taken. The ARM processor does not allow delayed branching. Three processors (CRISP, Ridge, and Dragon) use branch prediction in the compiler to increase the efficiency of delayed branch-

ing. The compiler sets or clears a bit in the branch instruction depending upon the likelihood of a branch taken. In CRISP, the technique of "branch folding" allows most branches to execute in zero time. This technique makes use of a dynamically generated "next address" field in each decoded instruction. In most processors, the compiler or reorganizer moves code to fill one or two slots following a branch with something more useful than *nop* instructions. The deeper the pipeline, the more difficult it is to fill the slots. For example, the deep pipelines of CDC and RCA processors have two delay slots to fill with code reorganization.

An interesting feature added to the MIPS-X processor consists of two versions of branch instructions. For a nonsquashed branch, instructions in the delay slots always execute. A squashed branch is used when both delay slots are not filled. Branch destination instructions are put into the slots and changed to *nop* instructions if the branch is not taken. SPUR performs a similar operation with its cancel compare instructions. To keep the processor array synchronized in CAP, instructions for *branch-taken* and *branch-not-taken* are transmitted by the scalar processor to the array processors. Computation then continues first with the taken instructions, then with the not-taken instructions, according to a branch condition mask.

Memory system. Because of the need to keep instructions and data supplied to the processors, RISC memory systems are necessarily quite complex. Several levels of memory hierarchy are used, often with data and instructions separated. Virtual memory is universally supported in the surveyed processors.

A hierarchical memory is one in which parts of the total storage space exist at different physical distances from the CPU, and at different operational speeds. It is currently not possible to build all the desired memory at a speed in which it can be accessed by the pipeline without delay. Several memory structures are possible, but the most common includes an on-chip instruction buffer of sufficient size to hold the next few instructions. This buffer is kept full by prefetch logic. Next is an on-chip or off-chip instruction and/or data cache. Some processors cache both data and instructions, others have just one or no cache. Finally, main memory exists off-chip and often off the processor board. Of course, access time increases as distance from the CPU or size of the memory system increases.

Six processors use on-chip instruction prefetch buffers in addition to instruction or data caches. CAP uses RAM in the vector processors as local instruction storage. Clipper, Transputer, and Dragon use a four-word instruction buffer. SPUR has a 512-byte prefetch buffer (used as cache) with a reported 98 percent hit ratio. On-chip instruction or data caches are implemented in five of the surveyed processors. ARM has a shared instruction and data cache. Future plans include splitting the on-chip cache for greater flexibility. CAP also has 256 kilobits of local RAM used to cache instructions and data in each vector processor. Each Dragon processor has an on-chip fully associative cache. The Transputer has one kiloword of on-chip memory to store instructions and data. MIPS-X processor has a 512-word on-chip instruction cache.

All the processors without on-chip cache use some form of off-chip cache for data or instructions. The Accel processor board uses three distinct on-board cache systems. A 64-kilobyte cache holds instructions. To support virtual addressing, a 32-kilobyte four-way set associative address translation cache is used. And a 64-kilobyte stack register cache is used for expansion of the CPU registers.

Separate MMU chips in the Clipper implement separate external data and instruction caches. Each cache contains four kilobytes of two-way set associative memory. The instruction cache is reported to have a 96 percent hit ratio. A program counter inside the instruction cache MMU reduces CPU-to-MMU bus bandwidth requirements. The data cache supports copy back, write through, and noncacheable control schemes. With two MMU chips, instruction and data access can occur concurrently.

The MIPS processor uses separate instruction and data caches of up to 16 kilowords each. Cache control for both, and memory management, are on the CPU chip, while memory resides off-chip. A write buffer chip in MIPS allows up to four writes to be queued, freeing the CPU.

Ridge processors use a 256-byte instruction cache and no data cache. Spectrum uses separate instruction and memory caches, both located on the CPU board. A three-level cache scheme is implemented in the Whetstone processor. The CPU directly accesses 128 kilobytes of very fast main memory. Separate on-board and off-board cache memories allow physical memory expansion up to 32 megabytes. Cache memory is updated through a separate DMA interface with the system disk.

In addition to an instruction buffer, MIPS-X uses an expandable 16- to 64-kiloword external direct mapped instruction cache. This cache is designed to minimize memory bus traffic.

Separate 128-kilobyte instruction and data caches are located on the SPUR processor board. The Berkeley Ownership algorithm insures cache consistency in a multiprocessor environment. All three GaAs processors use separate instruction and data caches. Due to chip resource constraints, the caches must be located off-chip. The CDC processor, like Clipper, uses two MMU chips to separate instruction and data cache control. Each CDC cache is one kiloword, direct mapped.

System issues

Division of functions. All the surveyed processors reflect different approaches to the design of reduced instruction set computer systems. Many of the processors divide system functions across chip boundaries, allowing integration of a large amount of functionality. Almost all these processors provide floating-point support utilizing the IEEE 754 standard. Floating-point support usually is provided by a separate coprocessor chip.

The Accel processor is available through Celerity computers with one or two floating-point units. The floating-point unit is physically separate from the CPU and loads data and instructions through the CPU registers.

As a user configurable system, the AMD processor allows connection of separate ALU, multiplier, and floating-point units, encouraging inclusion of only necessary functions. Each unit is a separate VLSI chip.

CAP is unique in that it implements an array of 20 identical RISC processors with memory on a single integrated circuit. This array chip employs software-based fault tolerance, necessary with such an extremely large device, by disabling faulted processors or memories. A multiprocessor array can be constructed from one or more CAP chips, depending upon the application. A scalar execution unit

fetches instructions and provides control for an array of CAP chips that comprise the parallel execution unit. Each processor on the array chip has a microinstruction set similar to other bit slice processors, containing *and*, *add*, *or*, and *sub* type instructions. CAP may be treated as a collection of SIMD processors, each operating in synchrony on its own data.

The Clipper processor board uses three chips of two different functions. The CPU/FPU chip contains execution circuitry. Floating-point support is included on the same chip as the integer execution unit to avoid delays associated with moving floating-point operands between chips. Two combined cache and memory management unit chips are used on the Clipper board. One CAMMU chip manages instruction cache and memory access, and the other one manages operand cache and memory. This arrangement allows overlapped instruction and data access. Floating-point operations are performed on-chip for greater performance. This eliminates moving instructions and data to a coprocessor.

The Dragon processor is used in a workstation of one to ten identical processors with a shared memory. Cache in each processor mediates between processor and memory, reducing bus traffic—typically the bottleneck in shared memory systems.

FAIM-1 is a fully distributed multiprocessor system with no shared memory. Each processing element, called a Hectagon, contains an Evaluation Processor, Switching Processor, Post Office (topology dependent hardware), Instruction Stream Memory, Scratch Memory, and Associative Memory. Hectagons are connected together into hexagonally arranged Surfaces. Surfaces can also be connected into larger structures. The memory system is a highly parallel associative store. FAIM-1 is used as a hardware accelerator unit attached to a host Lisp processor. The prototype FAIM-1 will have 19 processors, but the system can be scaled to an arbitrarily large number of processors.

The MIPS system is implemented as a three-chip set. A CPU chip contains all integer execution, memory management, and cache control. A tightly coupled floating-point processor supports IEEE single- and double-precision operations. A write buffer chip provides queueing of memory access. This allows the CPU to continue its task without waiting for a write to complete.

The IBM ROMP processor is implemented in two devices, a CPU chip and a separate memory management unit. CPU and MMU communicate over a 32-bit parallel packet switched channel. A floating-point accelerator board is available for inclusion in the IBM RT PC system.

The Spectrum processor is designed to support a wide range of applications. Three kinds of assist processors are available. Coprocessors may be added for graphics or floating-point support. Special function units can be added to perform fixed-point arithmetic, encryption, emulation, and other functions. Finally, multiprocessing units can be added to the system.

Transputer is a RISC processor for parallel processing. Each processor has four dedicated 10 megabit-per-second interfaces to other Transputers or peripherals. Other processing units, such as communications link adapter, graphic, or floating-point, can be added to the system. Multiprocessor performance can scale linearly with the number of processors.

Whetstone uses three independent processors to perform arithmetic/logic, memory/address, and DMA-I/O operations. An optional floating-point unit is also available.

MIPS-X differs from the original SU-MIPS in the inclusion of coprocessor support. MIPS-X is designed as a processor in a shared memory multiprocessor system. To that end, memory bus traffic is minimized by keeping the cache miss ratio very small through use of a large (16- to 64-kiloword) external cache. Simulations indicate that six to eight processors may share the same memory without noticeable performance degradation due to bus contention.

The SOAR processor is implemented as a board for use with a Sun workstation. The SPUR processor follows development of RISC I, RISC II, and SOAR at Berkeley. It provides support for Common Lisp and IEEE 754 floating-point standards in a parallel processing environment. SPUR is a general purpose processor with some Lisp support. Three types of chips comprise the majority of the SPUR processor. A CPU performs integer execution and instruction fetch. A cache controller chip manages the memory and the instruction cache. A separate FPU implements the IEEE standard. The FPU tracks CPU instructions and fetches required instructions from the instruction cache.

The CDC system uses four major integrated circuits. A CPU executes most instructions with a RISC processor. Up to four different coprocessors can be used to augment the instruction set, with execution capabilities for specific applications. A floating-point coprocessor (FCOP) is being developed. The operand and instruction caches are each controlled by their own memory management unit. System devices interconnect through 32-bit parallel operand and instruction data buses, a 24-bit parallel instruction address bus, and a 26-bit parallel operand bus. Each of the system chips is pipelined.

The McD GaAs processor design was derived from the SU-MIPS microprocessor. Modifications include reduction in transistor count and complexity, and the addition of two coprocessor interfaces for floating-point operations. The CPU performs all fetch operations. The FPU loads all instructions in parallel with the CPU, executing only the ones that apply to floating-point operations. A system controller chip contains interrupt management, clock generation, and low-speed I/O operations.

Operating system/language support. The dominant operating system supported on new RISC processors is Unix or one of its variants. The ARM, Clipper, MIPS, Pyramid, Ridge, and Spectrum processors all have applications that run under a resident Unix-type operating system. The Unix processors all support some subset of the C, Pascal, Fortran, Ada, and Cobol programming languages. Optimizing compilers exist for all these languages.

The ROMP processor was designed to support IBM's PL.8 compiler. The two Berkeley processors, SOAR and SPUR, support Smalltalk and Lisp. SPUR was designed to also allow execution of untagged languages. Dragon was designed primarily to execute Lisp, but also provides support for Cedar and Smalltalk. The FAIM-1 processor was designed around the OIL intermediate language, a high-level symbolic processing language. CRISP is based upon the Bell Labs C Machine, and was optimized for execution of programs written in the C language.

All three GaAs processors were designed for high level language execution. The CDC specifically supports, through the development of optimizing compilers, the Pascal and Ada languages. Whetstone represents a departure from the norm of

Table 4. RISC processors performance.

Processor	Cycle Time/Clock Rate	Instruction Rate (MIPS)
Accel	100 ns	3.2 MIPS
ARM	8 MHz	3-4 MIPS
AMD29000	125 ns	4-5 MIPS
CAP	I: 10 MHz, II: 25 MHz	12.5 MIPS peak, scalar unit
Clipper	33 MHz	5 MIPS
CRISP	16 MHz	> 10 MIPS
Dragon	10 MHz	5 MIPS per CPU
FAIM-1	?	?
MIPS	16.6 MHz	8 MIPS
Pyramid	125 ns	2-4 MIPS
Ridge 32	125 ns	1-4 MIPS
ROMP	170 ns	2 MIPS
Spectrum	30 MHz	10.8 MIPS
Transputer	50 ns	10 MIPS
Whetstone	50 ns	5-13.3 MIPS
MIPS-X	20 MHz	> 10 MIPS
SOAR	400 ns	See text
SPUR	150 ns	?
CDC GaAs	5 ns	91 MIPS
McD GaAs	10 ns	100 MIPS
RCA GaAs	200 MHz	200 MIPS peak

this survey by supporting an existing user base in the RDOS, SLICE, and IRIS systems.

Family requirements. A few processors were designed under the constraint of compatibility with an existing family of processors. The HP Spectrum gained some complexity by the requirement of object code compatibility with existing Hewlett-Packard computers. This level of compatibility carries with it a great marketing advantage, since users can experience at least a two times performance increase with existing applications by changing hardware.

The Whetstone design was completely constrained by the requirement of assembly code compatibility with Data General Nova 1200 computers. This provides a reduced architecture product that is software compatible with an existing user base. Both Ridge Computers and Pyramid Technology are working on new additions to their existing RISC-based computer products.

Advanced Micro Devices has announced a new AMD 29000 RISC processor. This 25-megahertz device, with 192 registers configured as a stack cache, a four-stage pipeline, provisions for macroinstructions (similar to the Clipper), and single-cycle instruction execution, will be the next step beyond their current AMD 29300 series devices.

Performance notes

The processors in this survey can be divided into two performance groups. The largest group contains those processors with silicon implementations. From Table 4 we can see that most processors have cycle times in the range of 30 nanoseconds to 400 nanoseconds. The data in the Instruction Rate column are, where possible, average processing rates from measured or simulated benchmarks, and not peak processing rates. Transputer's 10 MIPS rate may be misleading, since it relies on the assumption that all instructions and operands reside in the on-chip RAM, with no external memory delays. Overall processing rates of multiprocessors such as SPUR, Transputer, and Dragon are scalable on the number of active processors.

A second performance group contains the gallium arsenide processors. These three processors are designed for a 200-megahertz clock. The clock rates and instruction rates are an order of magnitude greater than those possible from silicon processors. The difference in instruction rate figures for the CDC and McD processors is most likely due to the measurement method.

SOAR's performance is compared to other Smalltalk systems, the Xerox Dorado, and the VAX 11/780. Benchmarks show SOAR operating at 50 percent of the Dorado rate and about six times faster than the VAX. Choice of benchmark has a tremendous effect on performance measurement, since SOAR is a language specific processor.

Reduced instruction set computers have quickly moved into many different application areas, indicating that the RISC philosophy can be applied to special purpose processors as well as to large general purpose computers. Most of these computers were designed for engineering and scientific computing. Few of the surveyed processors possess every characteristic commonly attributed to RISC designs. Most share some CISC characteristics, providing additional processing capacity for a given application. In fact, it is interesting to note that every processor surveyed contained architectural features typically attributed to CISCs, and every CISC feature was represented in at least one RISC design, indicating that in future designs good and useful architectural concepts will survive.

It should be obvious that an optimizing compiler is an integral part of any RISC design. Most RISCs use some form of delayed branching and also require code reorganization for optimum performance. The development of optimizing compilers and reorganizers often lags behind development of new computing hardware. One must be certain these tools exist for the language in which their applications programs are to be implemented.

We can expect to see more introductions of RISC machines in 1987 and beyond. Several of the recently introduced processors have begun to find their way into commercially available systems (such as Clipper in Opus Systems personal mainframe computers, Integraph workstations, and an IBM PC accelerator card). We can expect this trend to continue as software support of these processors matures. □

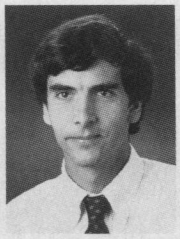
Acknowledgment

This research has been partially supported by NCR.

References

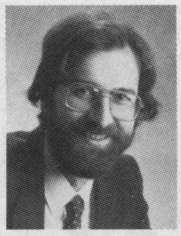
1. D.A. Patterson and J. Hennessy, "Response to 'Computers, Complexity, and Controversy,'" *Computer*, Nov. 1985, pp. 142-143.
2. M.G.H. Katevenis, *Reduced Instruction Set Computer Architecture for VLSI*, PhD thesis, University of California, Berkeley, Oct. 1983.
3. R.P. Colwell et al., "Computers, Complexity, and Controversy," *Computer*, Sept. 1985, pp. 8-19.
4. J.R. Mashey, "RISC, MIPS, and the Motion of Complexity," *Proc. Uniforum*, Feb. 1986.
5. M.E. Hopkins, "A Definition of RISC," *Proc. Int'l Workshop on High-Level Architecture*, 1984, pp. 3.8-3.11.
6. D. Tabak, "Which System is RISC?," *Computer*, Oct. 1986, pp. 85-86.
7. N. Mokhoff, "New RISC Machines Appear as Hybrids With Both RISC and CISC Features," *Computer Design*, April 1, 1986, pp. 22-25.

Additional references for this article are available by using the Reader Service Card. Circle number 183.



Charles E. Gimarc is currently a doctoral student in electrical engineering at Purdue University. He has held design engineering positions at Texas Instruments and Ampex Corp. He has also been a computer systems engineer in the Digital Signal Processing Laboratory at the Georgia Institute of Technology. His research interests include computer architectures for digital signal processing.

Gimarc received a BSEE degree from Texas A&M University and an MSEE from the Georgia Institute of Technology. He is a student member of the IEEE and the Audio Engineering Society.



Veljko M. Milutinović is on the faculty of the School of Electrical Engineering at Purdue University. He has published over 60 technical papers, two books, and four edited books. His current interests include VLSI computer architecture for GaAs, high-level language computer architecture, and artificial intelligence computer architecture. He has consulted for a number of high-tech companies and is currently involved in the industrial implementation of a 32-bit VLSI microprocessor for GaAs technology, with responsibilities in the microarchitecture domain.

Milutinović received his PhD from the University of Belgrade in 1982. He is a senior member of the IEEE and is on the Euromicro board of directors.

Readers may write to Milutinović at the School of Electrical Engineering, Purdue University, West Lafayette, IN 47907.