Reduced Instruction Set Computers (RISC) and Complex Instruction Set Computers (CISC)

ABSTRACT

The RISC and the CISC concepts have been discussed in this brief document. Advantages and Disadvantages of both processors have been presented.

Samir A. El-Seoud

Reduced **I**nstruction **S**et **C**omputers (**RISC**)
and
Complex **I**nstruction **S**et **C**omputers (**CISC**)

RISC (reduced instruction set computer)

RISC (reduced instruction set computer) is a microprocessor that is designed to perform a smaller number of types of computer instructions so that it can operate at a higher speed (perform more millions of instructions per second, or MIPS). In other words, a reduced instruction set computer (**RISC**) is a computer that uses a central processing unit (CPU) that implements the processor design principle of simplified instructions. Since each instruction type that a computer must perform requires additional transistors and circuitry, a larger list or set of computer instructions tends to make the microprocessor more complicated and slower in operation.

Briefly, **RISC** is a type of microprocessor architecture that utilizes a small, highly-optimized set of instructions, rather than a more specialized set of instructions often found in other types of architectures. It has been proven that about 20% of the instructions in a computer did 80% of the work. A processor based upon this concept would use few instructions, which would require fewer transistors, and make them cheaper to manufacture. By reducing the number of transistors and instructions to only those most frequently used, the computer would get more done in a shorter amount of time. To date, RISC is the most efficient CPU architecture technology.

The RISC concept has led to a more thoughtful design of the microprocessor. Among design considerations are how well an instruction can be mapped to the clock speed of the microprocessor (ideally, an instruction can be performed in one clock cycle); how "simple" an architecture is required; and how much work can be done by the microchip itself without resorting to software help.

Besides performance improvement, some advantages of RISC and related design improvements are:

- A new microprocessor can be developed and tested more quickly if one of its aims is to be less complicated.
- Operating system and application programmers who use the microprocessor's instructions will find it easier to develop code with a smaller instruction set.
- The simplicity of RISC allows more freedom to choose how to use the space on a microprocessor.
- Higher-level language compilers produce more efficient code than formerly because they have always tended to use the smaller set of instructions to be found in a RISC computer.

The most basic RISC feature is a processor with a small core logic that allows engineers to increase the register set and increase internal parallelism by using the following:

- Thread level parallelism: Increases the number of parallel threads executed by the CPU
- Instruction level parallelism: Increases the speed of the CPU's executing instructions

The words "reduced instruction set" are often misinterpreted to refer to a reduced number of instructions. However, this is not the case, as several RISC processors, like the PowerPC, have numerous instructions. At the opposite end of the spectrum, the DEC PDP-8, a CISC CPU, has only eight basic instructions. Reduced instruction actually means that the amount of work done by each instruction is reduced in terms of number of cycles - at most only a single data memory cycle.

Complex instruction set computing (CISC)

CISC is a processor design, where single **instructions** can execute several low-level operations (such as a load from memory, an arithmetic operation, and a memory store) or are capable of multi-step operations or addressing modes within single **instructions**.

The term "CISC" (complex instruction set computer or computing) refers to computers designed with a full set of computer instructions that were intended to provide needed capabilities in the most efficient way. Later, it was discovered that, by reducing the full set to only the most frequently used instructions, the computer would get more work done in a shorter amount of time for most applications. Since this was called reduced instruction set computing (RISC), there was now a need to have something to call full-set instruction computers - thus, the term CISC.

In contrast to RISC, CISC chips have a large amount of different and complex instruction to make life easier for the programmer by reducing the amount of instructions required to program the CPU.

The PowerPC microprocessor, used in IBM's RISC System/6000 workstation and Macintosh computers, is a RISC microprocessor. Intel's Pentium microprocessors are CISC microprocessors. RISC takes each of the longer, more complex instructions from a CISC design and reduces it to multiple instructions that are shorter and faster to process.

The CISC architecture contains a large set of computer instructions that range from very simple to very complex and specialized. Though the design was intended to compute complex instructions in the most efficient way, it was later found that many small, short instructions could compute complex instructions more efficiently. This led to a design called Reduced Instruction Set Computing (RISC), which is now the other major kind of microprocessor architecture. Intel Pentium processors are mainly CISC-based, with some RISC facilities built into them, whereas the PowerPC processors are completely RISC-based.

Overview

One of the big debates in the 1980s and 1990s was whether the CISC or RISC approach to instruction set design was correct. The idea behind RISC was to provide a simple set of instructions that could be used to perform complex operations, while the CISC people wanted complex instructions that could be used on their own.

The embodiment of the CISC design was the VAX. Writing VAX assembly was not much different from writing high-level code. In later VAX systems, many of these instructions were microcoded, which meant that they were decomposed into simpler real instructions that were then run on the real hardware.

After the VAX, Digital created the Alpha, a chip at the opposite extreme. The Alpha had a very small instruction set, but it ran incredibly fast. For many years, it was the fastest microprocessor that money could buy. Even now, several of the top 500 supercomputers are Alpha-based, in spite of the fact that the chip hasn't been in active development for five years.

In the early years, RISC did very well. Compiler writers loved the chips; they could easily remember the instruction sets, and it was easier to map complex language constructs onto sequences of RISC instructions than to try to map them to CISC instructions.

The first problems in the RISC philosophy became apparent with improvements in the way division was handled. Early RISC chips didn't have a divide instruction; some didn't even have a multiply instruction. Instead, these were created from sequences of more primitive operations, such as shifts. This wasn't a problem for software developers; they would just copy the sequence of instructions to accomplish a divide from the architecture handbook, put it in a macro somewhere, and then use it as if they had a divide instruction. Then someone worked out a more efficient way of implementing a divide instruction.
However, the RISC strategy also brings some very important advantages. Because each instruction requires only one clock cycle to execute, the entire program will execute in approximately the same amount of time as the multi-cycle "MUL command. These RISC "reduced instructions" require less transistors of hardware space than the complex instructions, leaving more room for general purpose registers. Because all of the instructions execute in a uniform amount of time (i.e. one clock), pipelining is possible.

The next generation of CPUs with divide instructions could execute the operation in fewer cycles, while those without took the same number of cycles to execute the series of instructions used as a substitute. This has been taken even further with Intel's latest Core micro-architecture. Some sequences of simple x86 operations are now combined into a single instruction that's executed internally.

Comparison between CISC Architecture and RISC Architecture

Comparison between CISC and RISC has always produced a very lively debate. This seems to be one issue where supporters of each side seemed to be distributed evenly. A close look at both though shows us that both have their own pros and cons.

Advantages and Disadvantages

RISC

- Small instruction set
- Few addressing modes
- Only load and store instructions can access memory
- Simple, fixed-length instruction code format
- Most instructions execute in 1 clock cycle (common exceptions are load, store, mul, div)
- Hardwired control unit
- Many registers
- Produces faster and cheaper processors
- Apple Mac G3 instructions are executed over 4x faster over its Intel equivalent.
- Simpler hardware
- Shorter design cycle
- Requires more lines of code that becomes increasingly complex

- Despite the speed advantages of the RISC processor, it cannot compete with a CISC CPU that boasts twice the number of clock cycles
- Code debugging, code expansion and system design are the areas that become contentious subjects

Examples:

- Alpha Originally known as **Alpha AXP**, is a 64-bit reduced instruction set computing (RISC) instruction set developed by Digital Equipment Corporation (DEC).
- ARM Originally **Acorn RISC Machine**, later **Advanced RISC Machine**, is a family of reduced instruction set computing (RISC) architectures for computer processors.
- PowerPC is a RISC instruction set architecture created by the 1991 Apple–IBM–Motorola alliance, known as *AIM.*
- Sun Sparc The **Scalable Processor Architecture** (**SPARC**) is a reduced instruction set computing (RISC) instruction set architecture (ISA) originally developed by Sun Microsystems.

CISC

- Large instruction set
- Usually memory-to-memory or register-memory
- Sophisticated instructions
- Instruction cycle takes many clock cycles
- Many addressing modes
- Expensive processors than RISC types
- Comparatively slow
- Efficient coding means easier life for developers and the result has been some tremendous software that RISC users could not
- AMD, Cyrix, and Intel have driven the market very well and outwit the RISC based technology, where Macs once were fighting with problems to 500MHz+ PowerPC chips, their contemporaries were looking 1GHz processors.
- Microprogramming is as easy as assembly language to implement
- CISC machines upwardly compatible
- the compiler does not have to be as complicated
- instruction set & chip hardware become more complex with each generation of computers
- different instructions will take different amounts of clock time to execute
- Many specialized instructions aren’t used frequently
- CISC instructions typically set the condition codes as a side effect of the instruction

Examples:

- x86 is a family of backward compatible instruction set architectures based on the Intel 8086 CPU and its Intel 8088 variant. The 8086 was introduced in 1978 as a fully 16-bit extension of Intel's 8-bit based 8080 microprocessor, with memory segmentation as a solution for addressing more memory than can be covered by a plain 16-bit address. The term "x86" came into being because the names of several successors to Intel's 8086 processor end in "86", including the 80186, 80286, 80386 and 80486 processors.
- 68k series Register-memory, 2-operand, 56 instructions on 68000, more added to 68010, etc.
- AX Memory-to-memory, 3-operand, 240 instructions, 16 addressing modes, 16 general registers. The polyf instruction evaluates a polynomial given a pointer to an array of coefficients, the order of the polynomial, and a value for x.

CISC	RISC
Large number of complex instructions	Small number of instructions
Instructions are of variable number of bytes	Instructions are of fixed number of bytes
Instructions take varying amounts of time for execution	Instructions take fixed amount of time for execution
It is prominent on Hardware, i.e. it emphasis on hardware	It is prominent on the Software, i.e. emphasis on software
Instruction set & chip hardware become more complex with each generation of computers.	Simpler hardware
It has higher cycle per second	It has low cycle per second
It has transistors used for storing instructions which are complex	More transistors are used for storing instructions in memory
LOAD and STORE memory-to-memory is induced in instructions	LOAD and STORE register-to-register are independent
It has multi-clock	It has a single-clock
Extensive use of microprogramming	Complexity in compiler
Expensive processors than RISC types	Produces faster and cheaper processor
Pipelining is difficult	Pipelining is easy
Less registers	Use more registers
More addressing modes	Fewer addressing modes
Small code sizes	Large code sizes

Solved example:
Rewrite the following program fragment that is written using the GPR instruction set for execution correctly on a CISC processor that provides the same instruction set as the **GPR** (General Purpose Register) processor but allows the register addressing mode to be used on the input operands or destination of any instruction. Assume that the GPR ISA provides only the register addressing mode, and that there are enough registers in the processor to hold any temporary values that you need to generate.

```
DIV    R1, [R2], [R3]
SUB R0, R1, [R5]
ADD R3, R0,          R1
MUL [R1], R0, R3
```

Solution:
Your goal should be to reduce the number of instructions as much as possible. This program fragment makes four memory references as part of arithmetic operations. To execute the fragment, we need to replace each of those with an explicit load or store. Here is a code fragment that does that:

```
LDR   R1, [R2]
LDR   R3, [R4]
LDR   R5, [R6]
LDR   R7, [R8]
DIV   R9, R1, R3
ADD  R10, R9, R5
SUB   R11, R7, R10
```

```
    STR    [R12], R11
```

Systems Suitability

If we try to sum up the differences between the two architectures, the competition is between the speed and the convenience, and as mentioned earlier, both have their own fans. Faster processing of RISC architecture makes it an ideal candidate for the systems that require extreme speed, probably gaming and Image editing solutions. CISC architecture probably is the best candidate for normal commercial systems, where the simplicity of producing new software is of great importance.

Though there could be another valid argument in favor of speed of RISC architecture is that with advent of new generation Operating Systems where they emit an API for doing all that you need for software development, and then this abstraction can absolve a normal user from the complexities generally associated with RISC systems programming.

Future

So what does the future holds for us? Where are we heading to? Can RISC system have similar success of CISC based system?
Or RISC producers will continue succumbing to market monopoly of CISC processor producers like Intel, AMD and Cyrix?
Probably these questions could only be answered by time.

“As the world enters the 21st century the CISC Vs. RISC arguments have been swept aside by the recognition that neither terms are accurate in their description. The definition of ‘Reduced’ and ‘Complex’ instructions has begun to blur, RISC chips have increased in their complexity (compare the PPC 601 to the G4 as an example) and CISC chips have become more efficient.” (Gareth Knight, 2003)

EPIC (Explicitly Parallel Instruction Computing) developed by Intel for Server market has also emerged during this tussle between CISC and RISC.