RISC vs CISC

Kate Ericson

August 30, 2006

Dawn of Time

- Before the advent of compilers, all programming was done in machine code or assembly.
  - To make programming easier, more and more complex instructions were created
  - These instructions were direct representations of high level functions in high level programming languages
  - This was deemed a good idea, since at the time, hardware design was easier than compiler design

Dawn of Time

- Computers had very little memory
  - This promoted a high density of information in programs
  - Instructions of variable size, instructions which perform multiple operations, and instructions that both moved data and performed data computations
  - The ability to pack instructions densely was considered more important than instruction decodability

Dawn of Time

- Register count was small
  - Bits in registers are more expensive than external memory, and would have been difficult to include in large numbers due to technology limitations
  - More registers require more instruction bits (so all registers can be addressed) - this would take up expensive RAM
    - In 1997, 1 MB of DRAM was about $5,000

Summary of sorts

- Because of these reasons, instructions were designed to do as much work as possible.
  - One instruction could load up two numbers, add them, and store the result back to memory
  - Another version of that instruction would do the same, but store the result in a register
  - Yet another version would read one number from memory, the second from register, and write the result back to memory
- This design philosophy became known as CISC (Complex Instruction Set Computer).

Orthogonality

- The goal of the hour was to provide every possible variation of every instruction. This principle is known as "orthogonality"
- This leads to complexity on the CPU, but this was considered a fair trade-off since in theory it is possible to tune each command individually

Research

- In the late 1970s researchers at IBM (other places too, but we don't really care about them) demonstrated that the majority of the orthogonal instructions were being ignored.
  - Compilers were gaining steam, and they only had a limited ability to make full use of CISCs orthogonal capabilities
  - Extremely specific instructions were found to be slower than more general instructions doing the same thing

CPU speedup

- CPUs started to run faster than the memory they used
- In the late 1970s, it was already apparent that CPU and memory speed would grow further apart for at least the next decade
- To support the higher CPU speeds, more registers were needed
  - Additional registers would require more space on the chip
  - This space could be created by reducing the complexity of the CPU

Real World Backup

- Real world examples showed that most processors were over-designed
  - On average, 98% of the constants in a program will fit into 13 bits, while almost every CPU stored them in individual words
  - This suggests that the CPU should store the constants in unused bits of the instruction itself, decreasing the number of memory accesses
  - If this scheme is used, the operation needs to be small, so that there is enough room left in the 32 bit instruction to hold larger constants.

Real World Backup

- Real world programs spend most of their time executing simple operations
  - Focus was put on making these common operations as simple and fast as possible
  - The goal was to make instructions so simple, each could be fully completed in a single clock cycle.

RISC

- The focus on "reduced instructions" led to the result being called "reduced instruction set computer"(RISC)
- The term "reduced instruction set computer" is somewhat misleading-many are under the impression that there are fewer instructions in the processor's instruction set
- RISC designs often have huge command sets
- Over time, the old design technique became known as Complex Instruction Set Computer(CISC)

RISC

- Instead of a single complex instruction, code was implemented as a series of smaller instructions
  - This left more room in the instruction for data
- Unfortunately, this also meant that the total number of instructions that needed to be read from memory for any single program is larger, and takes longer.

More Speed

- In an attempt to speed up processors, the ideas of having pipelined and superscalar processors were conceived
- Both pipelined and superscalar designs required adding complexity to the CPU
  - Because of the streamlining of the RISC architecture, RISC chips easily took advantage of these new techniques
  - The complexity of CISC architecture kept CISC chips from immediately taking advantage of the new technology.

Side by Side

CISC

- Complexity found in hardware
- Memory-to-memory : load and store functionality found in a single instruction
- Less lines of code needed to provide same functionality

RISC

- Complexity on software side
- Register-to-Register : load and store are seperate instructions
- More instructions necessary to provide same functionality

Side by Side

CISC

- instructions not always the same size
- instructions are difficult to decode because instructions are not uniform
- to make use of pipelining, instructions need to be broken down to smaller components at processor level

RISC

- all instructions of a uniform size
- instructions are easier to decode because of how they are set up-ex: opcode will always be in the same place
- capable of using pipelining by design

Real World

- Where can I find these processor types?
  - Intel's x86 processors have CISC architecture
  - IBM's PowerPC processors have RISC architecture
- But Mac's don't use PowerPC processors anymore...
  - This is true however we can still find Power processors in Gamecubes
  - Wii
  - Xbox 360
  - Playstation 3
  - Other RISC processors can be found in other console and portable gaming systems
  - not to mention cellphones
  - and embedded chips in found in cars

So which one's better?

- Good Question!
  - Right now this is still pretty much in the air. While the PC world is dominated by CISC processors, elsewhere mostly RISC processors are used.
  - You can usually start a flame war by strongly taking one side or the other
- But really now, which one is better?
  - Only time will be able to tell for this one

So which one's better?

- Some will claim that RISC is cheaper and faster, so it is the processor that will withstand the test of time
- Others say that RISC architecture puts too much of a burden on software, that the only way to go is to push complexity to the hardware with CISC processors, as they are becoming faster and cheaper
- Yet more believe that RISC and CISC processors will someday merge
  - Today's RISC chips support as many instructions as older CISC chips
  - CISC chips are using starting to use techniques that were associated with RISC chips

Sources Used

- A CPU: RISC vs CISC
- Complex instruction set computer
- RISC Architecture
- Reduced instruction set computer
- The Pentium 4 and the G4e: an Architectural Comparison