Computer Architecture

COEN 210

Yuan Wang

The CISC Approach

The primary goal of CISC architecture is to complete a task in as few lines of assembly as possible.
This is achieved by building processor hardware that is capable of
understanding and executing a series of operations.

For this particular task, a CISC processor would come prepared with a specific instruction

for example: "MULT"

When executed, this instruction loads the two values into separate registers, multiplies the operands in the execution unit, and then stores the product in the appropriate register.

Thus, the entire task of multiplying two numbers can be completed with one instruction:

MULT 3, 5

MULT is what is known as a "complex instruction."

It operates directly on the computer's memory banks and does not require the programmer to explicitly call any loading or storing functions. It closely resembles a command in a higher level language.

For instance, if we let "a" represent the value of 3 and "b" represent the value of 5,
then this command is identical to the C statement "a = a * b."

advantages: the compiler has to do very little work to translate a high-level language statement into assembly.
Because the length of the code is relatively short, very little RAM is required to store instructions.
The emphasis is put on building complex instructions directly into the hardware.

CISC architecture focuses on reducing the number of instructions per program

Examples: x86 family

Examples: x86 family

(architecture compatible with Intel 8086 and successors)

mostly used in desktop PC and laptop.

16 bit Intel (8086, 8088, 80186, 80186)
32 bit Intel (80386, 80486, Pentium…)
64 bit Intel (Pentium, Core i3, i5, i7, i9…)
32 bit AMD (AMD386, AMD486, AMD586…)
64 bit AMD (Phenom, Ryzen…)

backward compatible — new CPU can run older operating systems

The RISC Approach

RISC processors only use simple instructions that can be executed within one clock cycle.

Thus, the multiplication "MULT" command will be divided into three separate commands:
"LOAD," which moves data from the memory bank to a register
"PROD," which finds the product of two operands located within the registers
"STORE," which moves data from a register to the memory banks.

In order to perform the multiplication, a programmer would need to code four lines of assembly:

```
LOAD A, 3
LOAD B, 5
PROD A, B
 STORE A
```

At first, this may seem like a much less efficient way of completing the operation.
Because there are more lines of code, more RAM is needed to store the assembly level instructions.
The compiler must also perform more work to convert a high-level language statement into code of this form.

As 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 "MULT" command.

RISC strategy also brings some very important advantages. 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.

Separating the "LOAD" and "STORE" instructions actually reduces the amount of work that the computer must perform. After a CISC-style "MULT" command is executed, the processor automatically erases the registers. If one of the operands needs to be used for another computation, the processor must re-load the data from the memory bank into a register. In RISC, the operand will remain in the register until another value is loaded in its place.

Examples: ARM, MIPS, SPARC

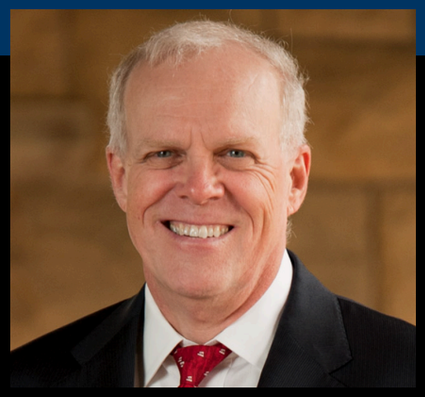
MIPS

developed as part of a VLSI research program at Stanford University in the early 80s.
Professor John_Hennessy,
started the development of MIPS with a brainstorming class for graduate students.

MIPS Computer Systems, Inc was founded in 1984
The company was purchased by Silicon Graphics, Inc. in 1992, and was spun off as MIPS Technologies, Inc. in 1998.
Today, MIPS powers many consumer electronics and other devices.

one of the first RISC processors

the MIPS processor implemented a smaller, simpler instruction set. Each of the instructions included in the chip design ran in a single clock cycle. The processor used a technique called pipelining to more efficiently process instructions.



Chairman of Alphabet Inc., Google's parent company

Turing Award 2017

John L. Hennessy

John Hennessy served as Stanford's tenth president from 2000 – 2016.

ARM

Arm Ltd.

Advanced RISC Machines

most widely used ISA

used in smart phones, tablet computers PDAs and other mobile devices due to their low costs, minimal power consumption, and lower heat generation

What is RISC-V

David Patterson originated the Berkeley RISC in 1980
coined the term “RISC”

Krste Asanović at the University of California, Berkeley, had a research requirement
for an open-source computer system, decided to develop and publish one.
The plan was to aid both academic and industrial users.
David Patterson at Berkeley joined the collaboration

Clean up MIPS instruction set, offers a simple, elegant, modern design

is provided under open source licenses that do not require fees to use

there are open source tools (simulator, compilers, debuggers,
and open source RISC-V implementations written in
HDL (Hardware Description Language)

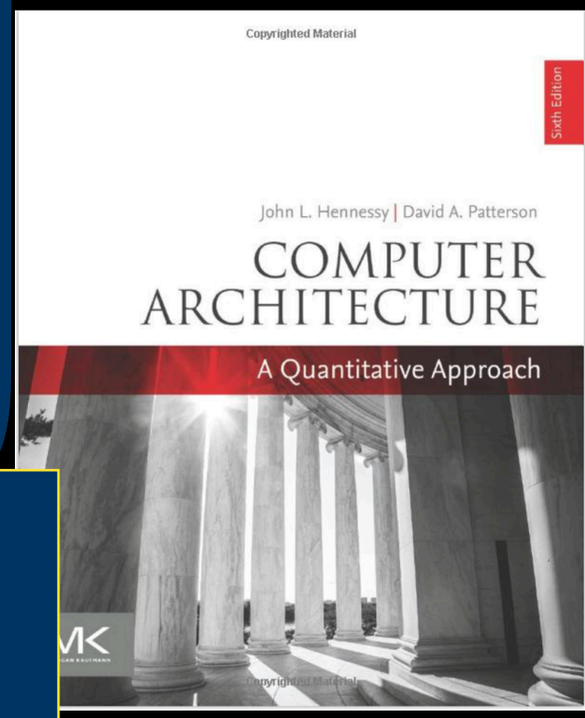
300 companies have joined the RISC-V foundation (except ARM and Intel)

Example

Nvidia plans to use RISC-V to replace their Falcon processor on
their GeForce graphics cards.

David Patterson

Krste Asanović



Hennessy and Patterson won the
2017 Turing Award for their work in developing RISC.
They coauthored the
“Computer Architecture: A Quantitative Approach”
and our textbook

RISC-V

SPARC

(Scalable Processor Architecture)

originally developed by Sun Microsystems (acquired by Oracle)

its design was strongly influenced by the experimental Berkeley RISC system

in 2017, Oracle terminated SPARC design

open and royalty free

Compatibility

Backward Compatibility

In hardware, this means that the new hardware can run old program for old hardware

For example

x86 architecture can run old application/operating system
for as early as 16 bit 8086 (not 8bit 8086) of 1974

this is because the instruction set is compatible

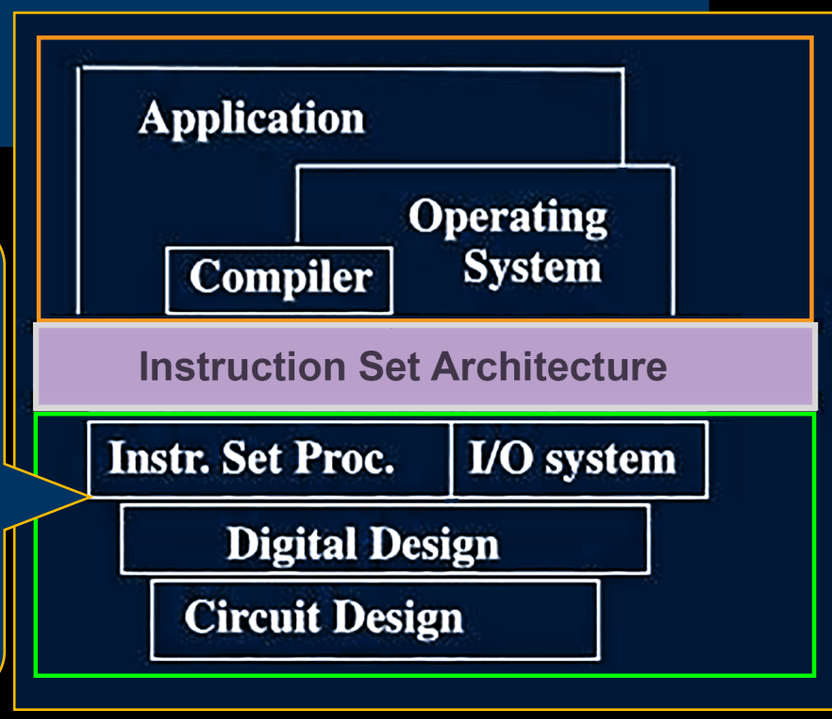
Newer game consoles are able to play older games
(XBOX 360, PS3)

Backward Compatibility in software,

For example

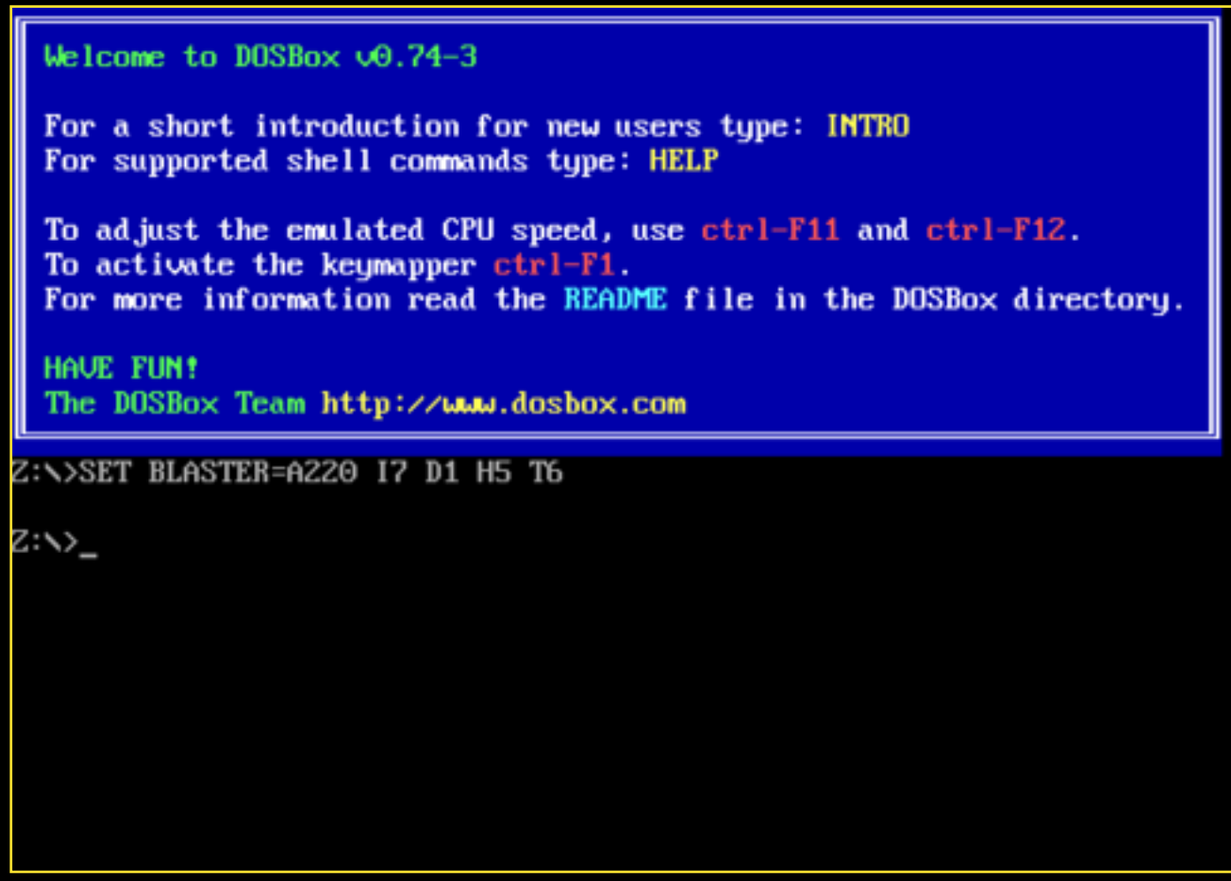
Newer compiler is able to compile older programs

with Instruction Set Architecture being compatible,
computer can be improved by improving the underlining organization
and physical implementation



Emulation

In computing, an emulator is hardware or software that enables one computer system (called the *host*) to behave like another computer system (called the *guest*).



DOSBox emulates the command-line interface of DOS.

Simulation

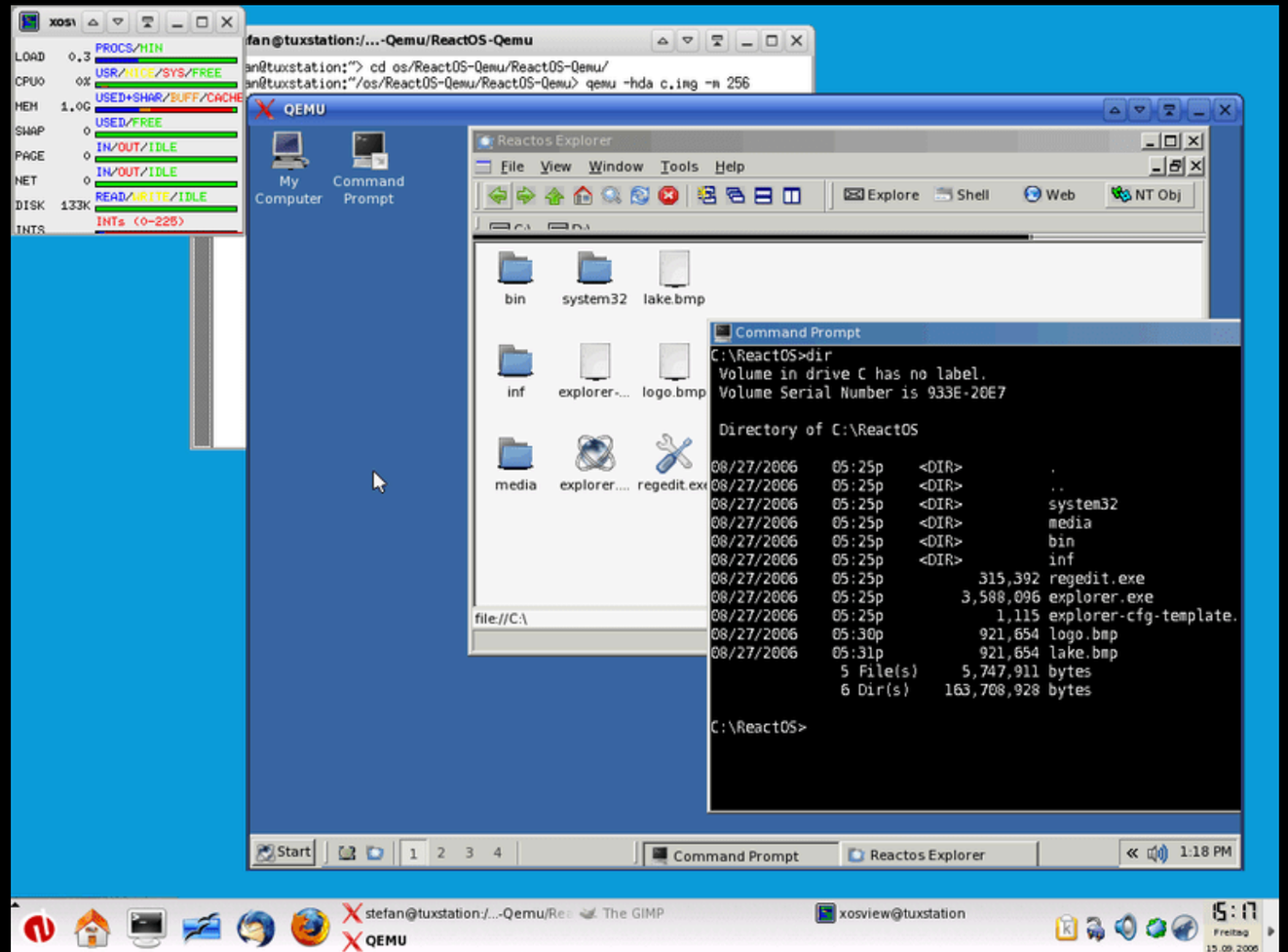
A computer simulation (or "sim") is an attempt to model a real-life or hypothetical situation on a computer so that it can be studied to see how the system works

Simulation is extensively used for educational purposes.

ISA-based virtualization

implemented in software
emulates the ISA in software where it interprets and
translates guest ISA to native (or host) ISA

A typical ISA based virtualization is QEMU (https://www.qemu.org/)
(a full implementation of virtualized ISA supporting systems such as x86, ARM, PowerPC, SPARC, etc.)



Running ReactOS on Linux

Schedule (temp)

Week 1: Introduction, numbers in computer, basic logic design

Week 2: Instruction set of RISC V, instruction formats, addressing modes,

Week 3: Arithmetic

Week 4: Data path and control of single cycle implementation

Week 5: Pipelining data path & control

Week 6: Pipelining, data and control dependence, forwarding

Week 7: Memory hierarchy, caches

Week 8: Memory hierarchy, caches (continued).

Week 9: Memory hierarchy, virtual memory

Week 10: Parallel processors