Reduced Instruction Set Computer

RISC - Reduced Instruction Set Computer

- By reducing the number of instructions that a processor supports and thereby reducing the complexity of the chip,
- it is possible to make individual instructions execute faster and achieve a net gain in performance
- even though more instructions might be required to accomplish a task.

RISC trades-off

instruction set complexity for instruction execution timing.

RISC Features

- Large register set: having more registers allows memory access to be minimized.

- Load/Store architecture: operating data in memory directly is one of the most expensive in terms of clock cycle.
- Fixed length instruction encoding: This simplifies instruction fetching and decoding logic and allows easy implementation of pipelining.

All instructions are register-to-register format
except Load/Store which access memory

All instructions execute in a single cycle
save branch instructions which require two.

Almost all single instruction size & same format.

Complex Instruction Set Computer

CISC - Complex Instruction Set Computer

Philosophy: Hardware is always faster than the software.

Objective: Instruction set should be as powerful as possible

With a power instruction set, fewer instructions needed to complete (and less memory) the same task as RISC.

- CISC was developed at a time (early 60's), when memory technology was not so advanced.
- Memory was small (in terms of kilobytes) and expensive.

But for embedded systems, especially Internet Appliances, memory efficiency comes into play again, especially in chip area and power.

Comparison

CISC	RISC
Any instruction may reference memory	Only load/store references memory
Many instructions & addressing modes	Few instructions & addressing modes
Variable instruction formats	Fixed instruction formats
Single register set	Multiple register sets
Multi-clock cycle instructions	Single-clock cycle instructions
Micro-program interprets instructions	Hardware (FSM) executes instructions
Complexity is in the micro-program	Complexity is in the complier
Less to no pipelining	Highly pipelined
Program code size small	Program code size large

Which is better

RISC

Or

CISC

?

Analogy (Chakravarty, 1994)

Construct a 5 foot wall

Method A: a large amount of small concrete blocks.

Method B: a few large concrete blocks.

Which method is better?

The amount of work done in either method is equal

Method A: more blocks to stack but easier and faster to carry.

Method B: fewer blocks to stack but the process is slowed by the weight of each block.

The distinction is in the dispersal of the work done.

RISC versus CISC

RISC machines: SUN SPARC, SGI Mips, HP PA-RISC

CISC machines: Intel 80x86, Motorola 680x0

What really distinguishes RISC from CISC these days lies in the architecture and not in the instruction set.

CISC occurs whenever there is a disparity in speed

- between CPU operations and memory accesses
- due to technology or cost.

What about combining both ideas?

Intel 8086 Pentium P6 architecture
is externally CISC but internally RISC & CISC!

Intel IA-64 executes many instructions in parallel.

Pipelining (Designing…,M.J.Quinn, ‘87)

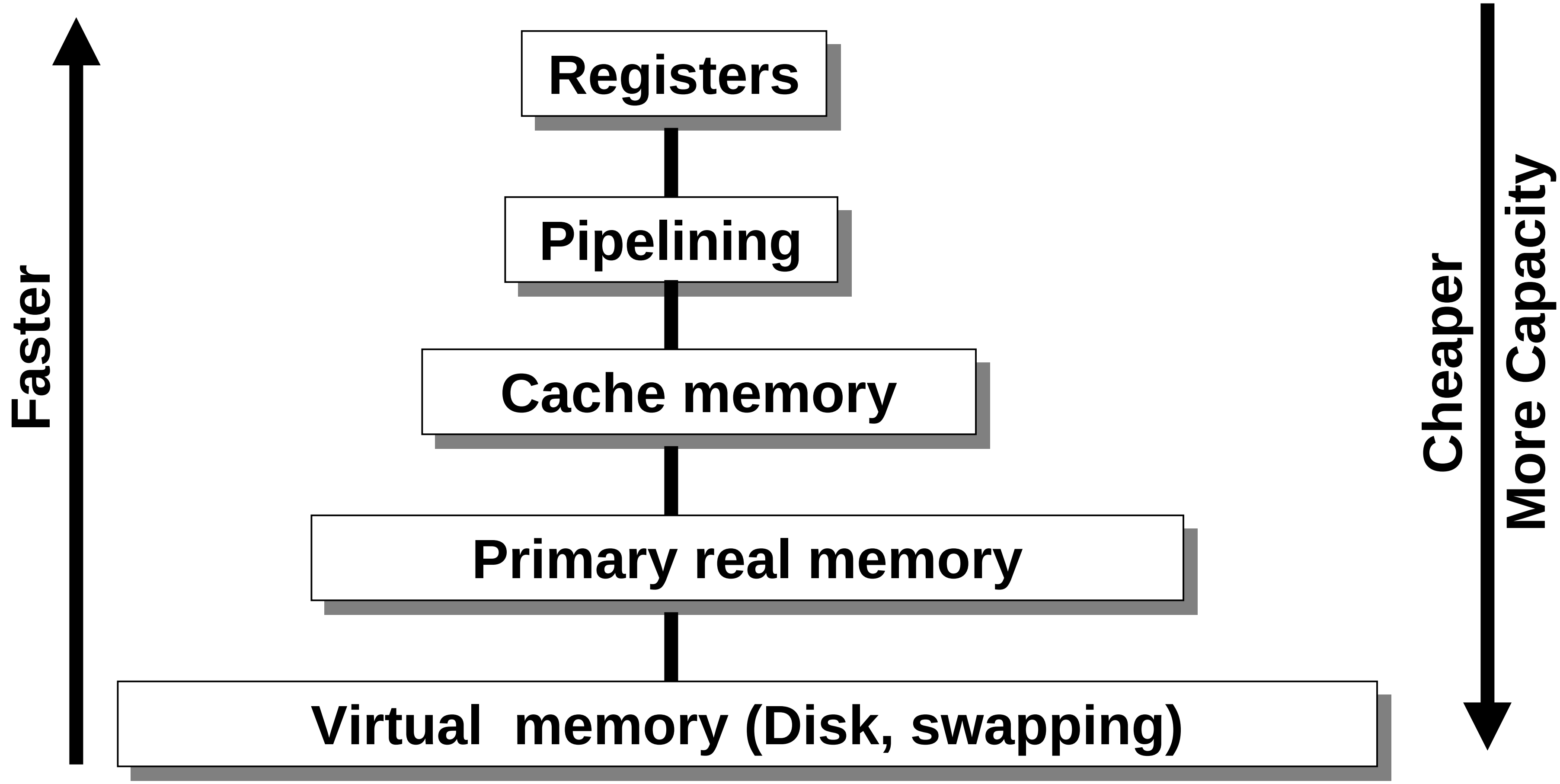
Instruction Pipelining is the use of pipelining to allow more than one instruction to be in some stage of execution at the same time.

Cache memory is a small, fast memory unit used as a buffer between a processor and primary memory

Ferranti ATLAS (1963):

- Pipelining reduced the average time per instruction by 375%
- Memory could not keep up with the CPU, needed a cache.

Memory Hierarchy



Faster

Registers

Pipelining

Cache memory

Primary real memory

Virtual memory (Disk, swapping)

Cheaper

More Capacity

Pipelining versus Parallelism (Designing…,M.J.Quinn, '87)

Most high-performance computers exhibit a great deal of concurrency.

However, it is not desirable to call every modern computer a parallel computer.

Pipelining and parallelism are 2 methods used to achieve concurrency.

Pipelining increases concurrency by dividing a computation into a number of steps.

Parallelism is the use of multiple resources to increase concurrency.