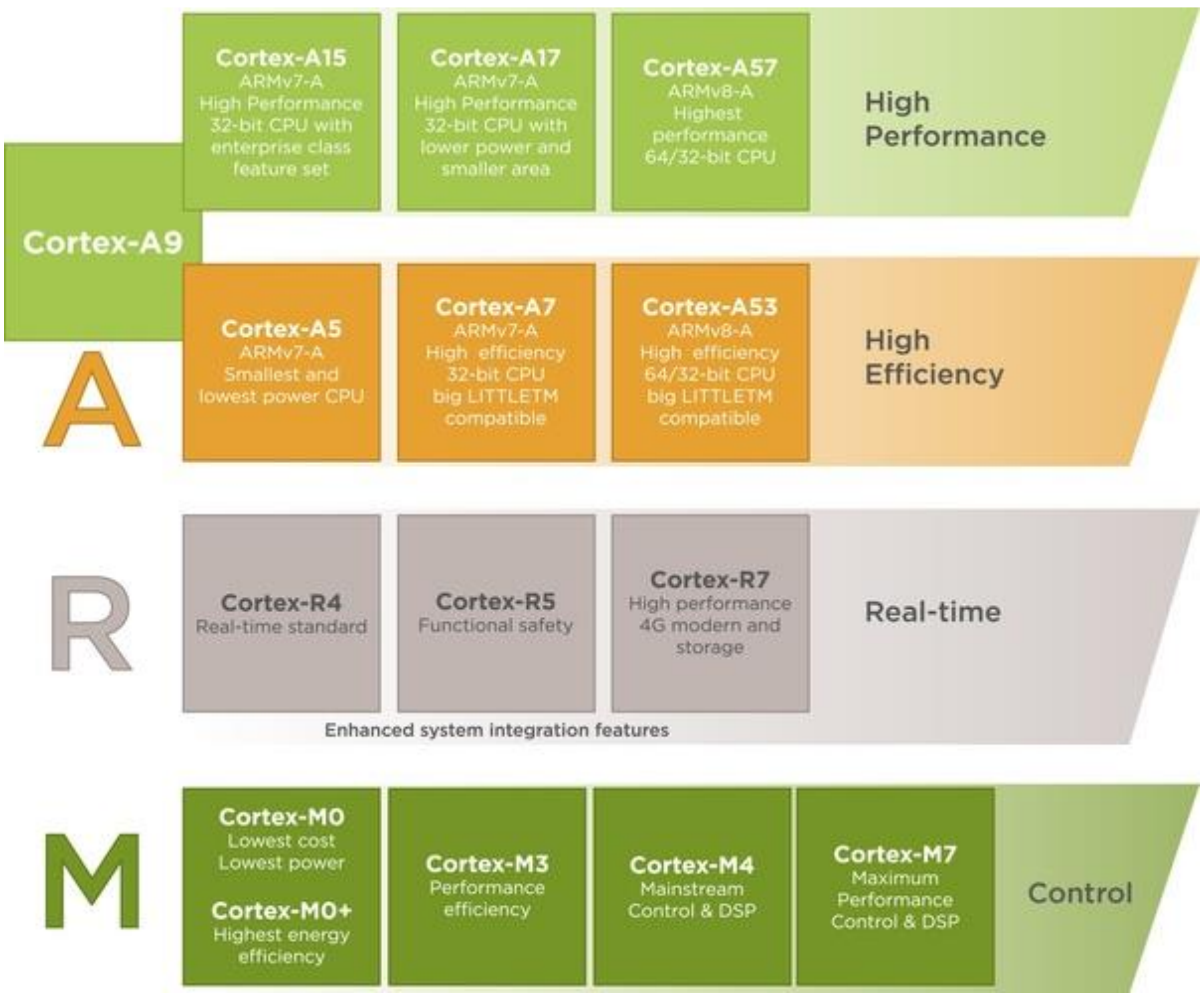
ARM

The clock speed of the processor is of great importance because the clock speed helps to determine the speed of the processor. More is the clock speed more fast is the processor.

The clock speed is the same thing you hear that your phone has a 1.3 Ghz processor. This 1.3 Ghz is the clock speed.


Cortex-A15
ARMv7-A
High Performance 32-bit CPU with enterprise class feature set
Cortex-A17
ARMv7-A
High Performance 32-bit CPU with lower power and smaller area
Cortex-A57
ARMv8-A
Highest performance 64/32-bit CPU
High Performance
Cortex-A9
A
Cortex-A5
ARMv7-A
Smallest and lowest power CPU
Cortex-A7
ARMv7-A
High efficiency 32-bit CPU big LITTLETM compatible
Cortex-A53
ARMv8-A
High efficiency 64/32-bit CPU big LITTLETM compatible
High Efficiency
R
Cortex-R4
Real-time standard
Cortex-R5
Functional safety
Cortex-R7
High performance 4G modern and storage
Real-time
Enhanced system integration features
M
Cortex-M0
Lowest cost
Lowest power
Cortex-M0+
Highest energy efficiency
Cortex-M3
Performance efficiency
Cortex-M4
Mainstream Control & DSP
Cortex-M7
Maximum Performance Control & DSP
Control

Reduced instruction set computing (RISC) has a special place on the map of hardware development, and the family of ARM processors is based on the RISC architecture

- **Cortex-A:** built for advanced operating systems and exhibits the highest possible performance;
- **Cortex-R:** caters perfectly to the needs of real-time applications and provides its users with the fastest response times;
- **Cortex-M:** mainly built for microcontrollers;

RISC stands for reduced instruction set computation while CISC stands for complex instruction set computation.

As their names suggest one is reduced and another one is complex.

In RISC machine cycle is already defined i.e one, so it fetches the instruction quickly but in CISC instruction set multiple machine cycle is present that's why it takes more time to execute.

RISC supports pipelining, has Harvard architecture, less use of HLL,less addressing mode etc.

Also RISC has many more advantages over CISC.

Conclusion from the above is RISC is better than CISC.

Through the 1990s, RISC processors increased clock speeds substantially.

Why RISC is used in mobile processor apart from CISC?

The general reason behind this is RISC processors are smaller in size than CISC

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

Advantages of CISC: Because few lines are required, less RAM is used than what's used in RISC.

So, in comparison, when multiplying two numbers, a CISC Processor can do it in a single line of command such as "MULT 2:3, 5:2", whereas a RISC would definitely take more number of steps to complete the same task, is to load the first and the second values, multiply them and store.

CISC :

-Multi-cycle instructions

-Microcoded decoder

-Instructions define multiple operations, which can make assembly programming easier (see DEC VAX)

-Von-Neumann architecture (unified Data and instruction memory)

RISC:

-single-cycle instructions

-Hardwired decoder

-Instructions define very simple operations , which can be used together .

-Emphasis on software/compiler , more difficult code in assembly because of simpler instructions

-Harvard architecture (separate data and instruction memory)

RISC designs are usually smaller , and it’s easier to extract ILP from them (pipelining , multi-issue).

Typical examples of the “CISC vs RISC” debate are intel’s x86 and ARM , respectively. But that’s no longer true. While the original x86 implementations where indeed fundamentally CISC processors , ever since the p6 architecture (or K5 on the AMD side) , all x86 processors convert the complex CISC x86 instructions and convert them to RISC-like micro-ops to be executed out of order internally. Modern x86 chips use some sort of hardwired decoder for simpler instructions (alongside a microcode ROM) , and they also split the L1 cache into separate instruction and data caches.

Similarly ,the original 1985 ARM CPU was very much a RISC chip , but modern ARMv7a and ARMv8a chips use microcode and have support for complex operations such as division , multiplication , floating point , trigonometry functions and more , all of of which take multiple cycles to execute and are not compatible with the original RISC philosophy .

RISC uses simple instructions which can be executed within one clock cycle, whereas primary goal of CISC is to complete a task in as few lines as possible.

As RISC architectures have a smaller set of instructions than CISC architectures in a pipeline architecture the time required to fetch and decode for CISC architectures is unpredictable.

Intel and AMD processor instruction set is CISC but internally they are RISC , you can consider it as CISC

your cell phone processor ARM is RISC (not purely)

RISC is better since intel and amd consumes a lot of power for decoding instructions

Why RISC is used in mobile processor apart from CISC?

The general reason behind this is RISC processors are smaller in size than CISC.

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

Advantages of CISC: Because few lines are required, less RAM is used than what’s used in RISC.

RISC: RISC processors only use simple instructions that can be executed within one clock cycle. No extra hardware is implemented onto the die of the processor to complex instructions saving die size of the processor.

So, in comparison, when multiplying two numbers, a CISC Processor can do it in a single line of command such as “MULT 2:3, 5:2”, whereas a RISC would definitely take more number of steps to complete the same task, is to load the first and the second values, multiply them and store.

Advantages of RISC: Since each command execution requires just one cycle of CPU, the time taken by the processor to do the same task would be almost the same as taken by a CISC processor. This makes it equivalently powerful as if CISC. However, x86 is the only architecture which retains CISC because of numerous improvements in its architecture.

One of the main issue in choosing mobile processors is Energy Consumption. No mobile manufacturing company will design a mobile which will last only for 2–3 hours. RISC processors are designed to have simple Instruction Set Architecture with low power consumption while CISC processors are designed are high performance processors consuming a lot of power as compared to RISC processors. Thats why RISC processors have been dominating the market in mobile computing.

RISC uses simple instructions which can be executed within one clock cycle, whereas primary goal of CISC is to complete a task in as few lines as possible.

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

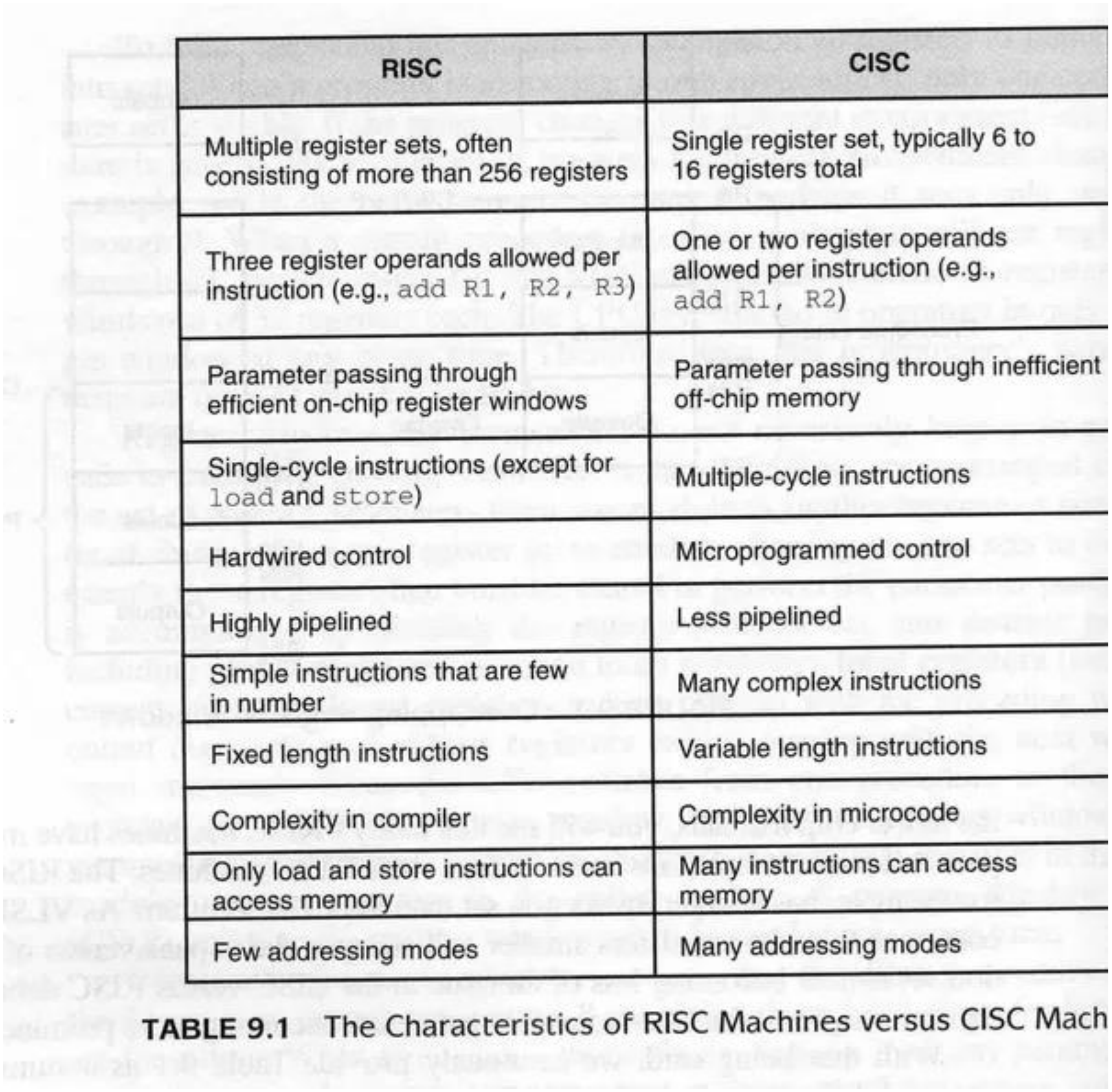
Advantages of CISC: Because few lines are required, less RAM is used than what’s used in RISC.

The actual differences between the three are too many for an answer here. Most of these subtle differences lie in the way memory is addressed, exceptions are handled, branches are executed etc. There are many subtle differences too that are beyond the scope of a brief answer here. From a high level though, I can think of these:

1. RISC vs CISC: This is the classic difference between ARM/MIPS and x86. RISC stands for Reduced Instruction Set Computer and CISC is Complete ISC. x86 is a CISC processor and both ARM/MIPS are RISC. The philosophy behind CISC processors is that a single instruction can do multiple things; like add an immediate number with a register, store this value to an address computed using some other register and also set arithmetic flags. RISC processors on the other hand would have two separate instructions for this; one to add two numbers and the other to store the result. This increases the instruction count but makes the instructions simpler. Which brings me to the second point.
2. Power vs Performance: Simpler instructions tend to consume lesser power. x86 processors are typically designed for high performance applications like servers, while RISC processors are used in mobile applications where performance is sometimes compromised for better power efficiency. It should be noted however that these lines are thinning down everyday. ARM has added more complexity to its instructions to get better performance and Intel breaks down its opcodes into micro-ops that are ARM like to achieve better power. MIPS perhaps remains the simplest ISA around and is used more heavily in embedded applications where ultra low power consumption is needed.

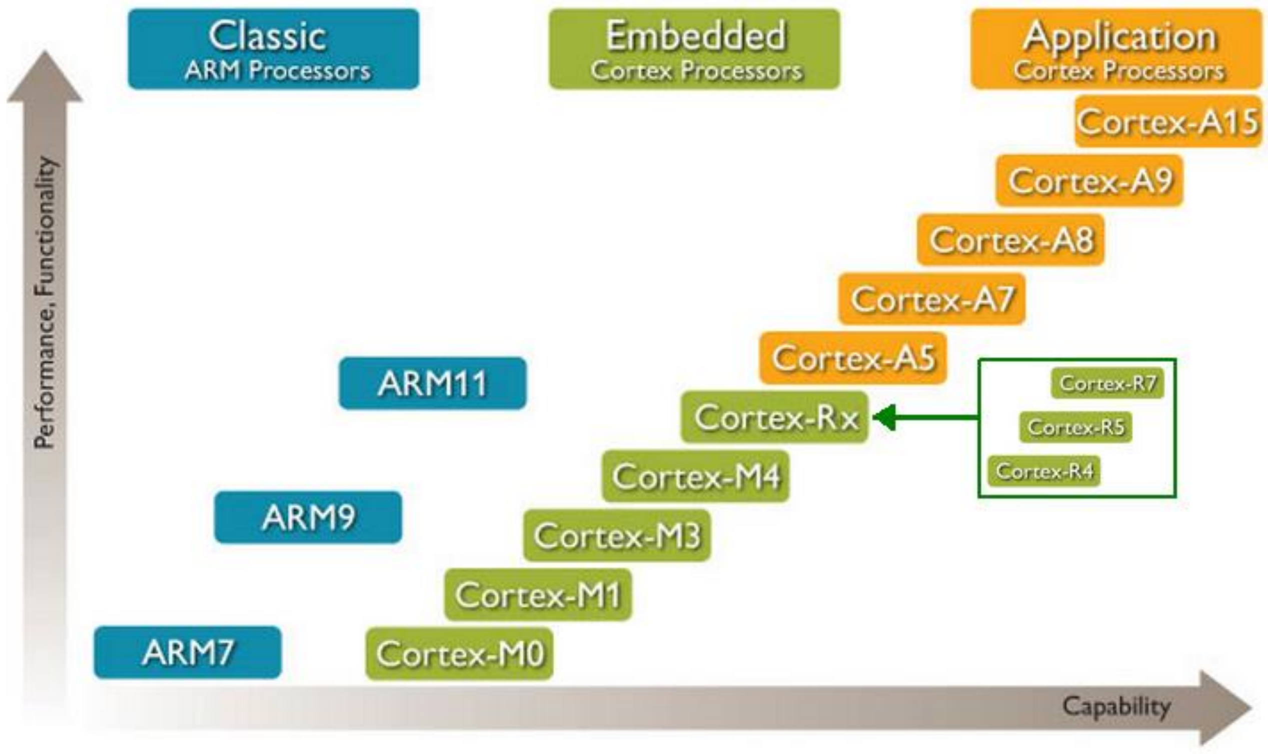
3. Security and virtualization: Not sure about MIPS here, but ARM has something called TrustZone and Intel has vPro. Both these features target hardware security. Both ARM and Intel have support for virtualization and memory management, and they differ only in their implementation. These are rather new features that are needed in newer applications and thus, both ISAs have one or the other flavor. So you see, the differences are trimming down.

All in all, the ISAs matter only from the programming point at this time. The mobile software ecosystem is way better developed and maintained for ARM than it is for x86. The picture reverses when it comes to desktop applications. That is the main reason why Intel is struggling to get into the mobile market and ARM is having a hard time to get into servers. In all of this, MIPS is there somewhere along with Sparc, PowerPC etc being used in rather niche applications.



RISC	CISC
Multiple register sets, often consisting of more than 256 registers	Single register set, typically 6 to 16 registers total
Three register operands allowed per instruction (e.g., `add R1, R2, R3`)	One or two register operands allowed per instruction (e.g., `add R1, R2`)
Parameter passing through efficient on-chip register windows	Parameter passing through inefficient off-chip memory
Single-cycle instructions (except for `load` and `store`)	Multiple-cycle instructions
Hardwired control	Microprogrammed control
Highly pipelined	Less pipelined
Simple instructions that are few in number	Many complex instructions
Fixed length instructions	Variable length instructions
Complexity in compiler	Complexity in microcode
Only load and store instructions can access memory	Many instructions can access memory
Few addressing modes	Many addressing modes

TABLE 9.1 The Characteristics of RISC Machines versus CISC Mach


Classic
ARM Processors
Embedded
Cortex Processors
Application
Cortex Processors
Cortex-A15
Cortex-A9
Cortex-A8
Cortex-A7
Cortex-A5
ARM11
Cortex-Rx
Cortex-R7
Cortex-R5
Cortex-R4
Cortex-M4
ARM9
Cortex-M3
Cortex-M1
ARM7
Cortex-M0
Performance, Functionality
Capability