



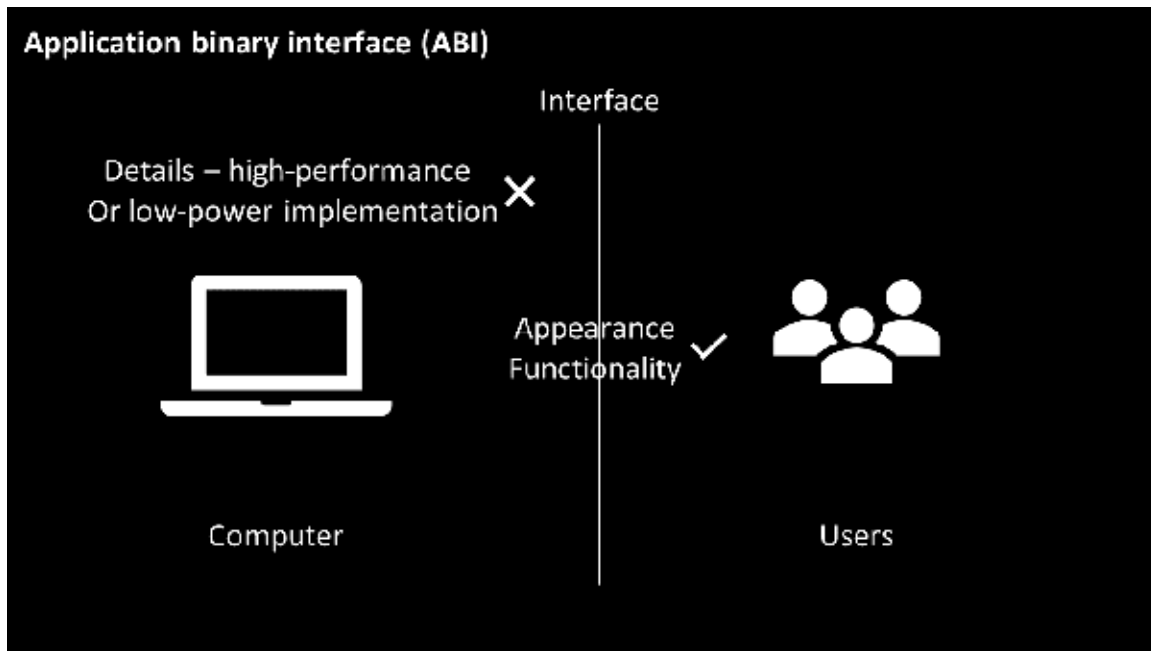
ABI – Get this one right – RISC-V is all yours.

Kunal Ghosh

Hi

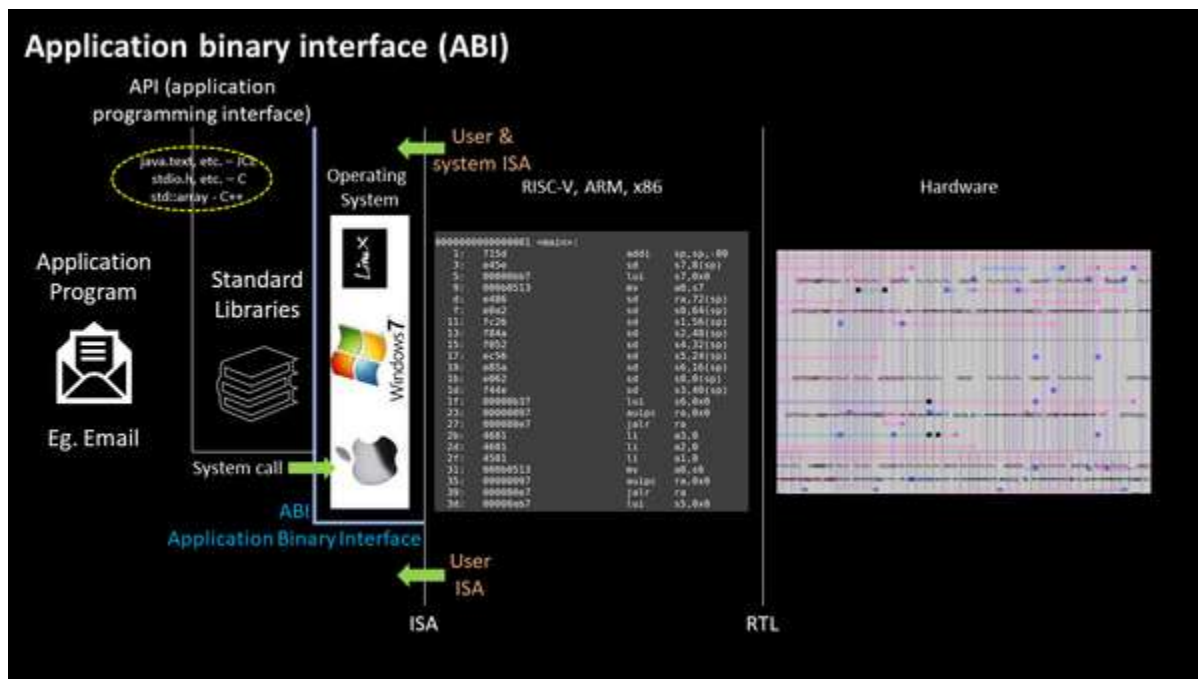
ABI (application binary interface), as the name says, is an interface, that helps programs access system hardware and services. Let's break it down

When users look forward to buying computers, they are, more likely, interested in the appearance and functionality of the computer, whether it serves their purpose. And, very unlikely, user will be interested in its physical design implementation of the chips inside the computer, like whether its high-performance or low power implementation



Now, here, the appearance and functionality, are an example of interface between users and computers. Let's consider another kind of interface – more VLSI and technical

Application program interacts with standard libraries using an interface called 'application programming interface (API)'. The very basic example, you must have seen, is while writing a C program where you use '#include stdio.h' – this interface defines the core input and output functions




Next interface is the operating system, which handles IO operations, allocates memory and some other low-level system functions. This layer is, by the way, also the one which converts the function programs into its assembly language program and/or machine language program, providing “bit-patterns” to the underlying hardware. This interface is the ISA interface (in this blog, its RISC-V ISA).

Then, the other VLSI level interface is RTL which implements the RISC-V specifications and is an important interface between the ISA and its physical design implementation

Coming back to ABI or application binary interface, it consists of 2 parts as shown in above image – one is the set of all user instructions itself, and second is the system call interface through the operating system layer. How does it do that in reality? Through registers shown in below image

Application binary interface (ABI)			
Register	ABI name	Usage	Saver
x0	zero	Hard-wired zero	-
x1	ra	Return address	Caller
x2	sp	Stack pointer	Callee
x3	gp	Global pointer	-
x4	tp	Thread pointer	-
x5-x7	t0-2	Temporaries	Caller
x8	s0/fp	Saved register/frame pointer	Callee
x9	s1	Saved register	Callee
x10-11	a0-1	Function arguments/return values	Caller
x12-17	a2-7	Function arguments	Caller
x18-27	s2-11	Saved registers	Callee
x28-31	t3-6	Temporaries	Callee



RISC-V architecture has 32 registers (we explained, in detail, why this architecture has 32 registers in our online course). Application programmer, can access each of these 32 registers through its ABI name, for example, you need know the value of stack pointer or move the stack pointer, all you need to do is “addi sp, sp, -16”, where ‘sp’ is the ABI name of stack pointer.

This blog should give a good head-start towards understanding ABI. Now, once you get the functioning of all ABI names in table shown in above image, believe me, RISC-V is all yours. We have done that in our online course, explaining each ABI using an example. You finish the examples, and you conquer the battle.

“When in doubt, just take the next small step”. One step that you can take today, and get closer to this architecture is through the below online course on RISC-V ISA:

<https://www.udemy.com/vsd-riscv-instruction-set-architecture-isa-part-1a/>

TAKE THE FIRST STEP NOW...

I will see you in class and happy learning...