CASPER

CS305: Computer Architecture

X86, ARM, other ISAs

https://www.cse.iitb.ac.in/~biswa/courses/CS305/main.html

https://www.cse.iitb.ac.in/~biswa/

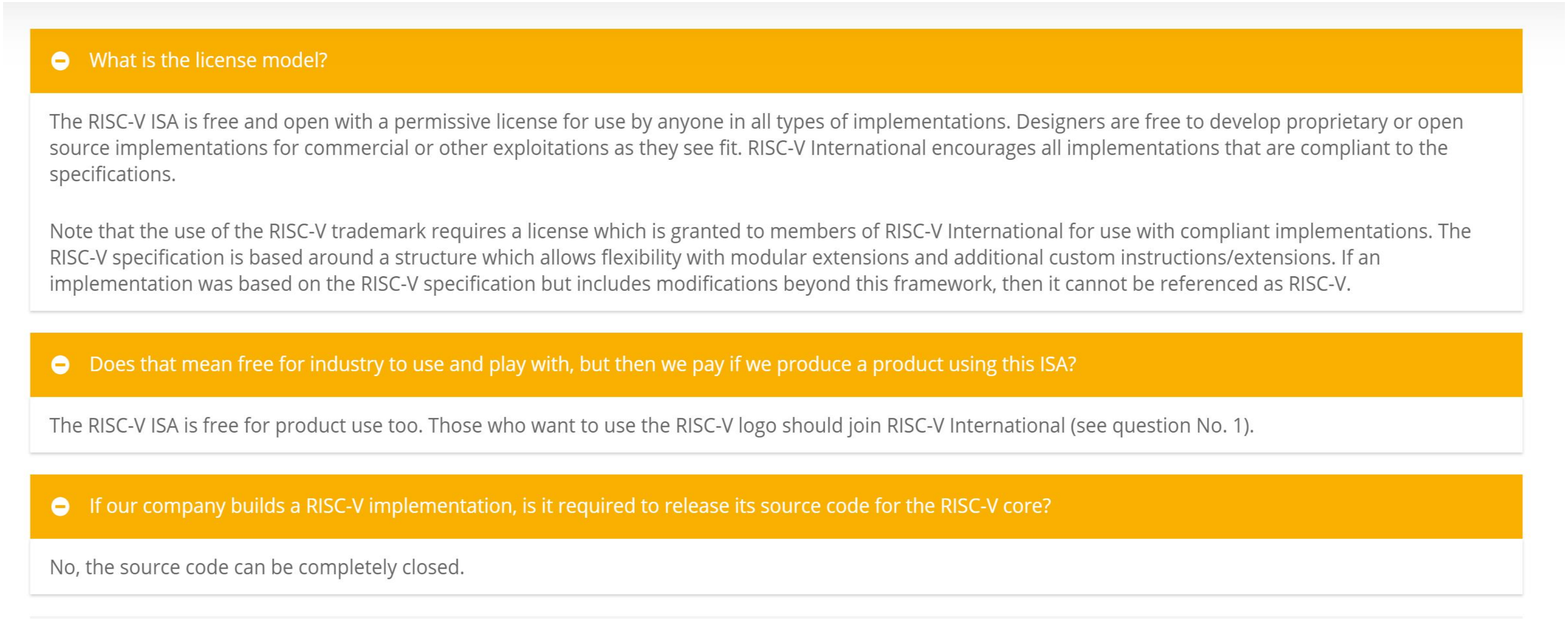
World of ISAs

- x86: Intel, AMD: Laptops, Desktops, Servers
- ARM: Arm, Qualcomm, Apple, Samsung: Mobiles

btw ARM: Advanced RISC Machines ☺

- RISC-V: Open-source ISA, what does it mean?

RISC-V, opensource



What is the license model?

The RISC-V ISA is free and open with a permissive license for use by anyone in all types of implementations. Designers are free to develop proprietary or open source implementations for commercial or other exploitations as they see fit. RISC-V International encourages all implementations that are compliant to the specifications.

Note that the use of the RISC-V trademark requires a license which is granted to members of RISC-V International for use with compliant implementations. The RISC-V specification is based around a structure which allows flexibility with modular extensions and additional custom instructions/extensions. If an implementation was based on the RISC-V specification but includes modifications beyond this framework, then it cannot be referenced as RISC-V.

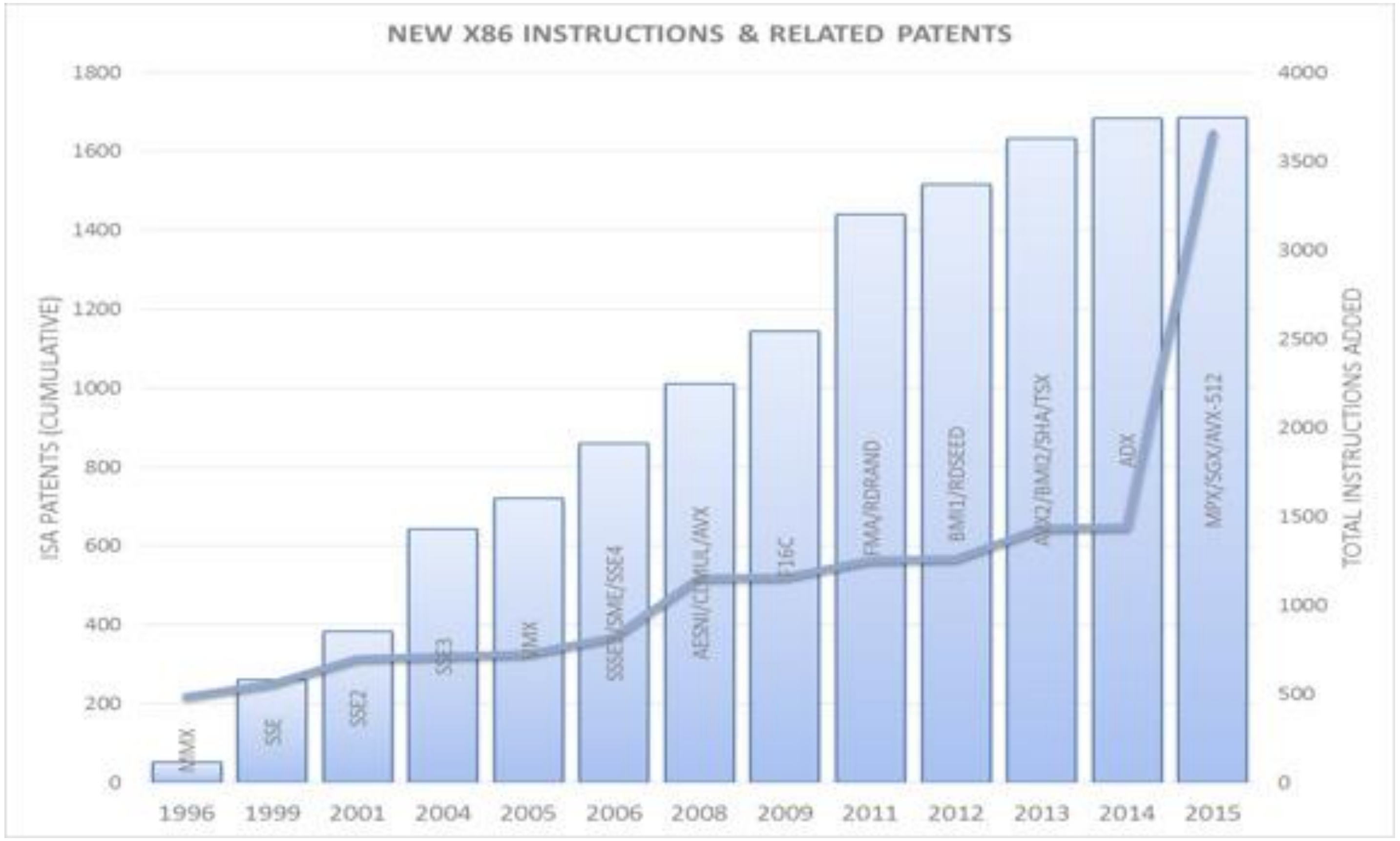
Does that mean free for industry to use and play with, but then we pay if we produce a product using this ISA?

The RISC-V ISA is free for product use too. Those who want to use the RISC-V logo should join RISC-V International (see question No. 1).

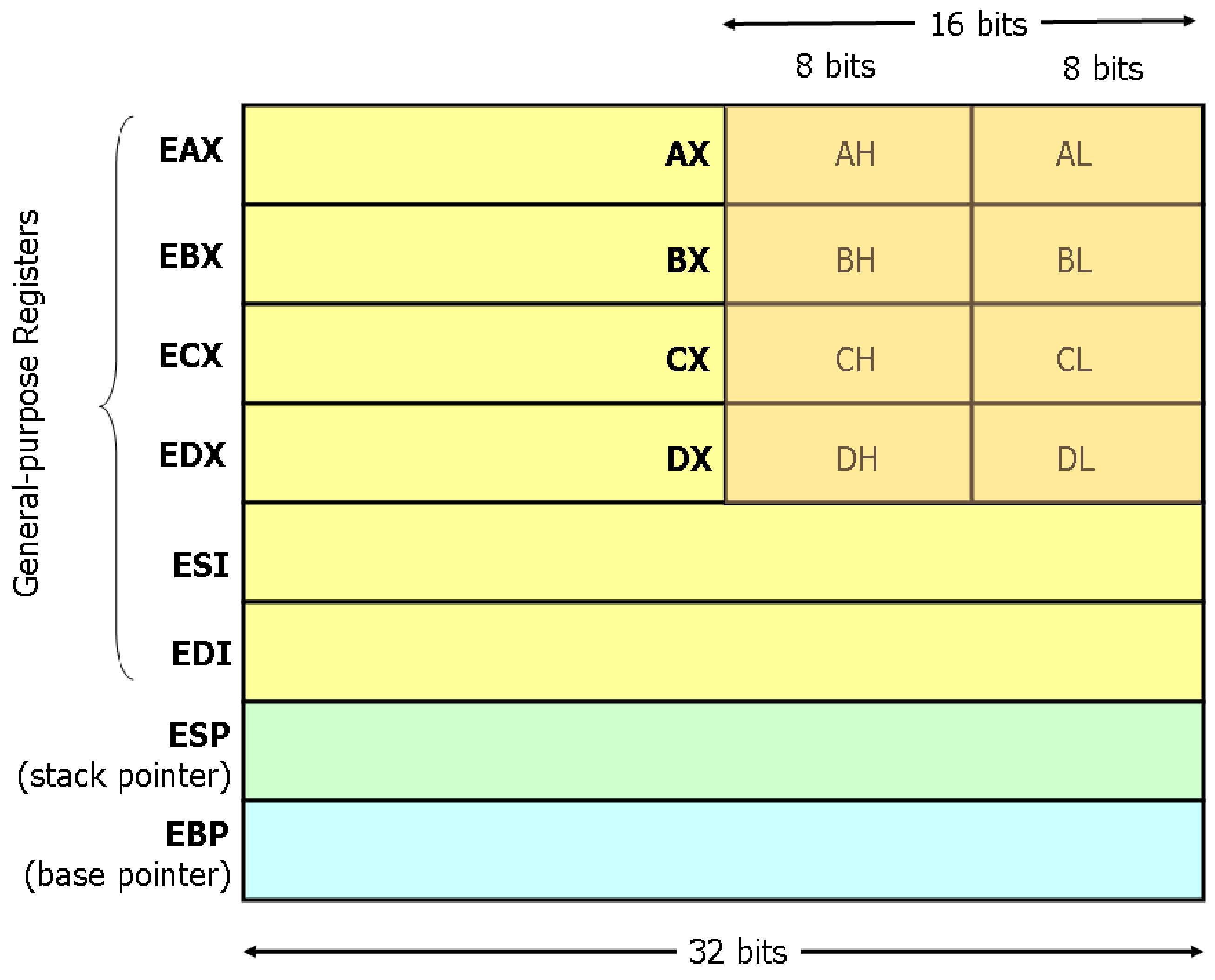
If our company builds a RISC-V implementation, is it required to release its source code for the RISC-V core?

No, the source code can be completely closed.

World of Intel and AMD (One instruction per month)



x86 Registers: 80386



Subtle Differences

- x86 arithmetic/logic instructions: one operand should act as both source and destination

add $s0, $s1 // Add $s0 and $s1 and put it in $s0 ☺

- One of the operands can be in memory:

wow (programmer) or oh no (instruction size ☹) !!

add $s0, Mem[$s1] ☺

- No more fixed-length instructions ☹ can be 4/8/X bytes

Why?

Fixed Width or Variable Width

Variable: No fixed size

6 bytes for add, 2 bytes for load

Smaller code footprint (compact)

Fixed:

4 bytes for all, Larger code footprint, simple decoding

Some more points of interest

- CISC vs. RISC
    - Initially motivated by "not good enough" code generation
    - RISC→ John Cocke, mid 1970s, IBM 801, Goal: enable better compiler control
- RISC motivated by
    - Memory stalls (no work done in a complex instruction when there is a memory *stall*?)
    - Enabling the compiler to optimize the code better
        - Find fine-grained parallelism to reduce *stalls*

ARM’s Compressed Thumb Instructions

The biggest reason to look for an ARM with the Thumb instruction set is

If you need to reduce code density, embedded domain

Normal instructions (32 bits) – Thumb ones, 16 bits

32-bit registers have compressed 16-bit counterparts

Fallacies

CISC instructions provide higher performance

Assembly language codes provide higher performance

Nandri