CS230: Digital Logic Design and Computer Architecture

Lecture 10: ISA/Microarch.

https://www.cse.iitb.ac.in/~biswa/courses/CS230/autumn23/main.html

ISA and Microarchitecture

Microarchitecture (Not exposed to us)

- Implementation of an ISA
- Programmer cannot see/access it

ISA:

add instruction

Microarch:

Implementation of an adder (ripple carry)

ISA: What does it provide?

Opcodes,

Addressing Modes,

Instruction Types and Formats,

Registers

Access control: user/OS

Address space,

Addressability,

Alignment

ISA: What does it provide?

Instructions:

Opcodes,

Addressing Modes,

Instruction Types and Formats,

Registers

Access control: user/OS

Memory: Address space,

Addressability,

Alignment

ISA must satisfy the needs of the software: - assembler, compiler, OS,

Microarchitecture

Rest of C230 after ISA ☺

Caches

Memory Controllers

Branch Predictors, Prefetchers, …

Microarchitecture

Processor is in state S

Instruction

Processor moves to state SS

State

The information held in the processor at the end of an instruction to provide the processing context for the next instruction.

Computer Architecture

ISA + Microarchitecture

Based on lectures so far

#registers

#cycles to access a register

#Width of the register (32/64 bit)

#Instruction that uses register to access memory

#cycles to access memory

Based on lectures so far

#registers: ISA

#cycles to access a register: Microarch.

#Width of the register (32/64 bit) : ISA

#Instruction that uses register to access memory: ISA

#cycles to access memory: Microarch.

x86: It has 128/256-bit registers and one-bit too ☺

Where to place it?

- Closer to high-level language → Small semantic gap, complex instructions

  (CISC kinda? e.g. quicksort an instruction ☺)

- Closer to hardware? → Large semantic gap, simple instructions (RISC kinda?)
- Remember: Compiler+ISA defines app’s instruction count

And then the Debate of RISC vs CISC

RISC: Reduced Instruction Set of Computers

Very few simple instructions

Example: MIPS

CISC: Complex Instruction Set of Computers

Lots of complicated instructions

Example: x86 kind of ☺ [x86 is CISC with RISC mysteries]

Mystery

INTEL *CONVERTS* CISC ONES INTO RISC ONES, AND GENERATE MICROOPERATIONS.

INTELLIGENT CISC-RISC DECODER

CONSUMES AROUND 2% OF THE CHIP AREA. GOOD OR BAD?

How Easy?

A billion-dollar idea ☺

i) requires changes to microarchitecture.

ii) requires changes to ISA.

iii) both (i) and (ii)

Think about the trade-offs ☺

Does it affect the system stack?

The Other ISAs

World of ISAs

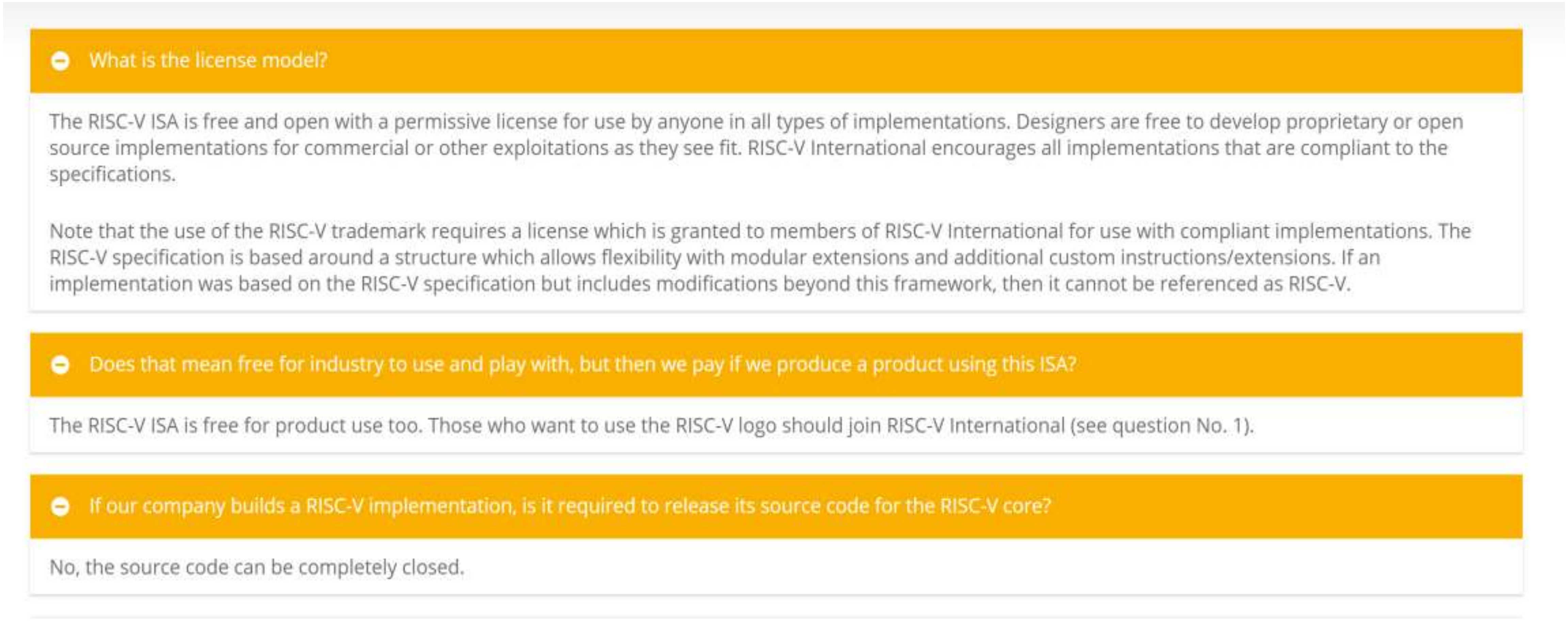
x86: Intel, AMD: Laptops, Desktops, Servers

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?

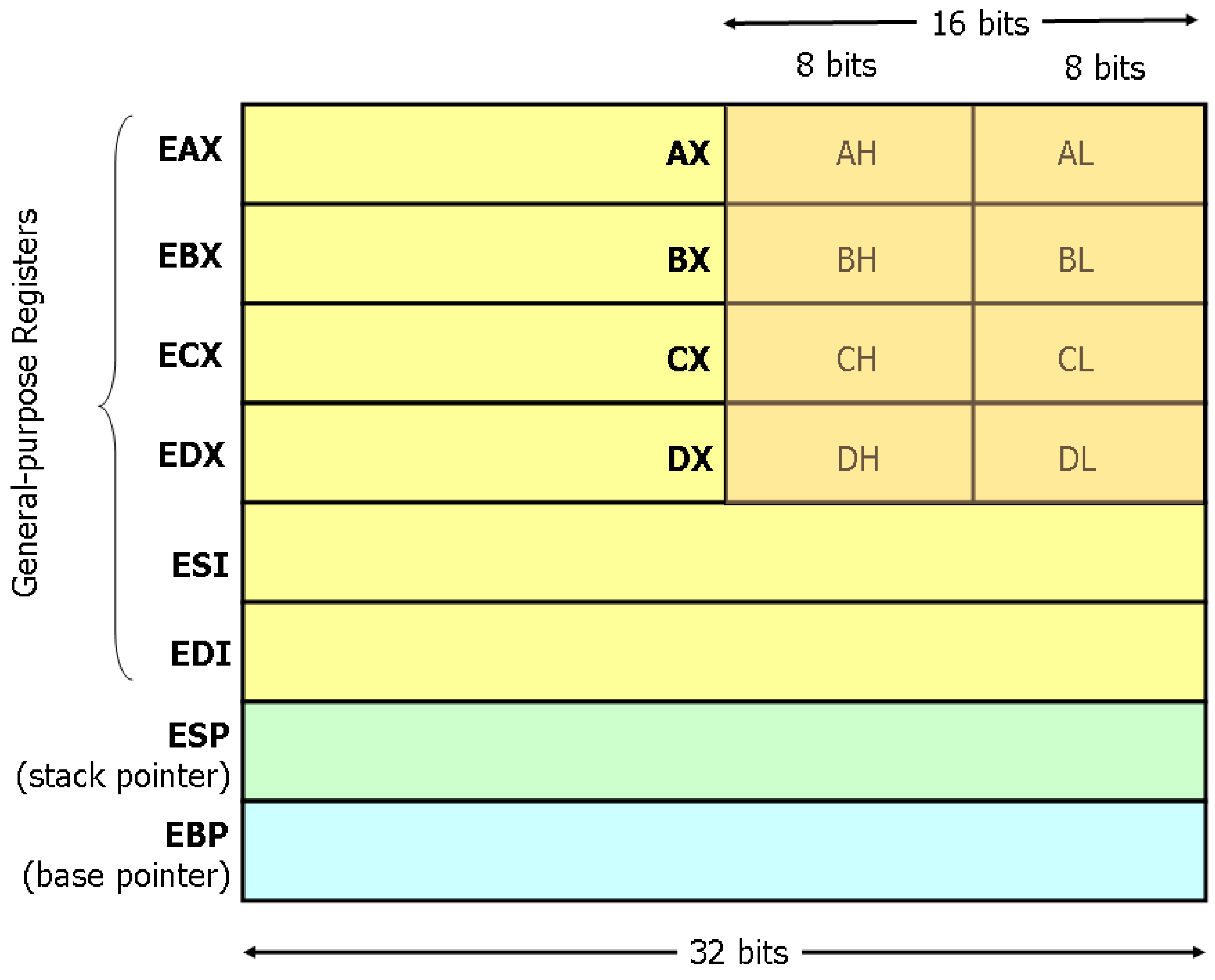
The RISC-V ISA is free for product use too. Those who want to use the RISC-V logo should join RISC-V International (see question No. 1).

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.

x86 Registers: 80386

For 64-bits, rax, rbx



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:

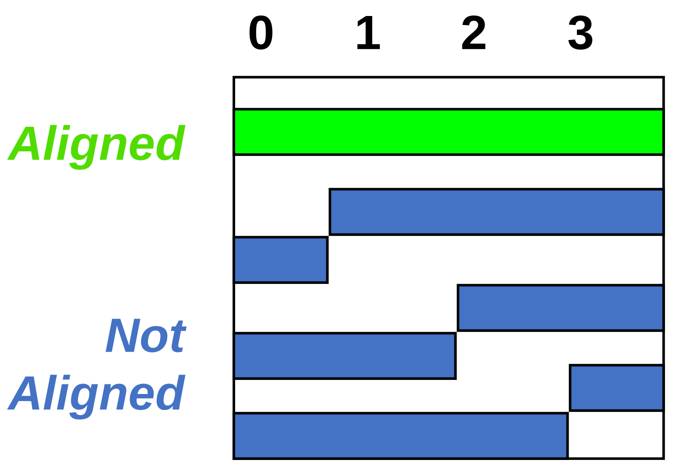
4 bytes for all, Larger code footprint, simple decoding

Fallacies

CISC instructions provide higher performance

Assembly language codes provide higher performance

Instruction Alignment: Why we need it?



Aligned:

x-byte access starting from an address y: y % x must be zero.

MIPS vs X86

MIPS does not allow unaligned accesses

x86 does not enforce alignment ☺

Whose job is to generate aligned/unaligned accesses?

MIPS vs X86

MIPS does not allow unaligned accesses

x86 does not enforce alignment ☺

Whose job is to generate aligned/unaligned accesses?

Compiler

Let’s go a bit deeper

Object of size s bytes at byte add. A is aligned if A mod s = 0

Alignment for faster transfer of data ?

Why fast ??

Think about memory (caches if you know).

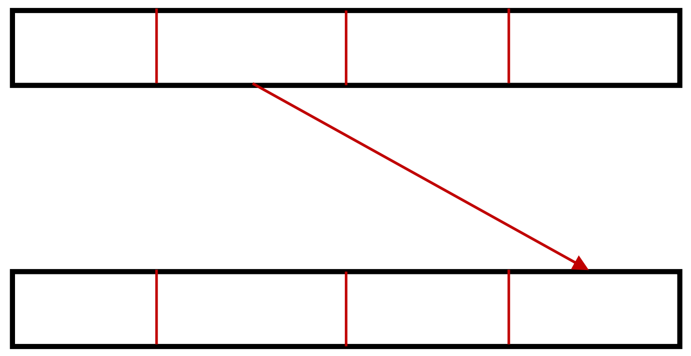
Memory operations and alignment network

LOADs and STOREs need an alignment network that makes sure data loaded/written are aligned.

lb R1, 1($s3)

4-byte chunk

Register R1



For the Curious ones

https://lemire.me/blog/2012/05/31/data-alignment-for-speed-myth-or-reality/