Important Interfaces

- Instruction Set Architecture (ISA)
  - Interface 4: User ISA.
  - Interface 3: System ISA.
- Application Binary Interface (ABI)
  - Interface 4: User ISA.
  - Interface 2: System calls.
- Application Programming Interface (API)
  - Interface 4: User ISA.
  - Interface 1: HLL library calls.

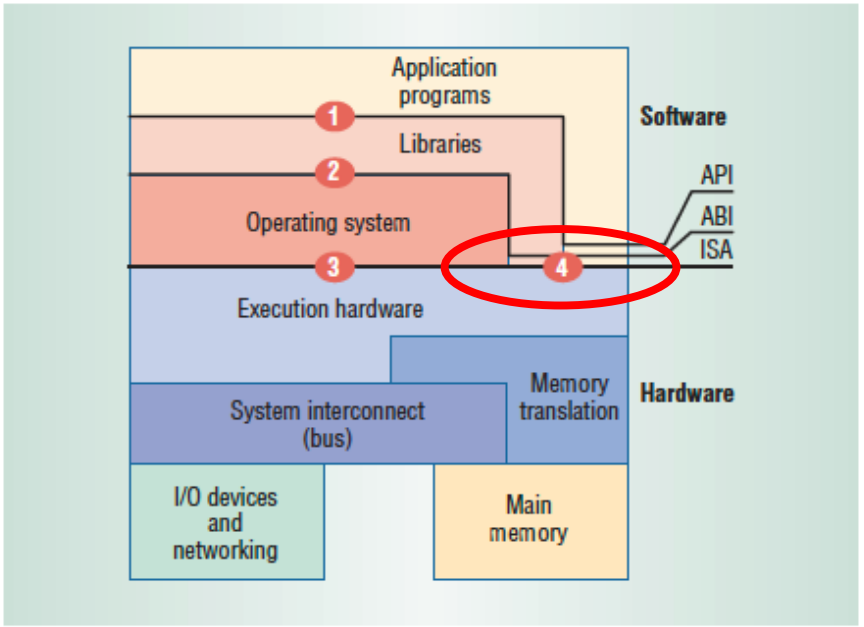


Figure 2. Computer system architecture. Key implementation layers communicate vertically via the instruction set architecture (ISA), application binary interface (ABI), and application programming interface (API).

From James E. Smith and Ravi Nair, "The Architecture of Virtual Machines", *Computer*, May 2005, 32-38. IEEE Computer Society.

ISA Taxonomy

Based on the maximum number of operands *explicitly* specified in instructions.

- 0-operand machine, aka “stack machine”
  - Operands come implicitly from the top items in a stack of values.
  - E.g., Java® Virtual Machine, SaM.
- 1-operand machine, aka “accumulator machine”
  - Single implicit accumulator register that is both the left operand and the result. The right operand is specified explicitly.
  - E.g., early machines, small microcontrollers (MOS 6502).
- 2-operand machine
  - Two named operands, one of which also serves as the result.
  - E.g., x86-64.
- 3-operand machine
  - Three named operands, two for inputs, one for result.
  - E.g., Armv8.

Instruction Encoding

- At the ISA representation level, an instruction is a bit string that is interpreted as a *verb* rather than as a *noun*.
  - Encoding specifics vary widely across machines.
- At the ISA semantics level, an instruction is a mapping from machine states to machine states.
- An instruction needs to specify (at minimum)
  - Source operands: What input values it needs to read (how many there are, how wide they are, where they come from).
  - Opcode: What operation it performs on those input values.
  - Destination location: Where the output value of the operation is stored.
  - What side-effects, if any, occur.

The RISC-CISC Debate

- The dominant ISA style in the mid-1970s featured complicated operations that combined memory access with data manipulation.
  - [PDP-10] `HLROM A, LOC`: Take the left half of the contents of register `A`, place them in the right half of memory location `LOC`, and replace the left half of the memory location with ones.
  - [PDP-10] `TLCE A, LOC`: Using the contents of `LOC` as a mask, select the corresponding bits in the left half of register `A`. If all those bits are equal to zero, skip the next instruction; and in any case, replace those bits by their boolean complement.
  - [IBM 360] `MVCL R1, R2`: Operands `R1` and `R2` each represent the even register of an even-odd consecutive register pair (2-3, 4-5, 6-7, …). The even registers contain the address of target and source fields, while the odd registers contain their lengths. Additionally, a pad character can be placed in the high order byte of the odd register containing the length of the source field. This character is used to fill up the target field when the source field is shorter and all of its contents have been transmitted. The instruction detects destructive overlap and indicates it by setting the condition code.
- Such instructions made sense when programs were written in assembly language and memory space was scarce, but were difficult as target ISAs for compilers of HLLs. They were also difficult to create fast implementations of, leveraging Moore's Law.

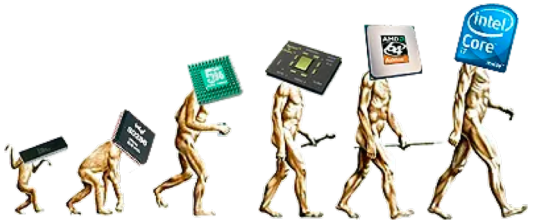
The RISC-CISC Debate

- Measurements of actual usage of instructions revealed that such complicated instructions were used rarely.
  - IBM (801 project), Berkeley, Stanford.
- This led to a new style of ISA design, characterized by:
  - Separating memory access from data manipulation.
  - Providing (orthogonal) primitives rather than solutions.
  - Co-designing ISA and compiler.
  - Amenable to fast implementations leveraging Moore's Law.
- From John Cocke's Turing Award lecture (1988)

  "Another satisfying aspect of the 801 project was that many of our ideas were considered sufficiently interesting by others and stimulated considerable additional research and experimentation in many universities. Berkeley coined the name "RISC" (Reduced Instruction Set Computing) for similar work."
- The term CISC was coined retroactively to mean "pre-RISC".

Modern CISC Exemplar: x86

- Today's x86 is the second-oldest line of computer architecture.
  - Intel launched the original 4-bit 4004 in 1971.
  - Has survived many industry cycles, self-inflicted wounds, mis-steps, and competition from AMD.
- For more details, see
  - https://www.computerworld.com/article/2535019/timeline--a-brief-history-of-the-x86-microprocessor.html
  - https://www.pcgamer.com/a-brief-history-of-cpus-31-awesome-years-of-x86/
- ISA reference
  - Download from https://software.intel.com/content/www/us/en/develop/download/intel-64-and-ia-32-architectures-sdm-combined-volumes-1-2a-2b-2c-2d-3a-3b-3c-3d-and-4.html
  - Warning: 5038 pages.



Modern RISC Exemplar: Armv8

- ARM (originally Acorn RISC Machine, then Advanced RISC Machines)
  - Primarily sold as IP cores to licensees, who use the core to create microcontrollers, CPUs, and SoCs based on those cores.
  - Low cost, minimal power consumption, lower heat generation than competitors make it desirable for light, portable, battery-powered devices.
  - Also used in desktops, servers, and supercomputers.
  - Dates back to the 32-bit ARM1 from 1985.
  - Additional instruction sets for specialized purposes: Thumb for improved code density, Jazelle for directly handling Java bytecodes, security extensions.
- ISA reference
  - Download from https://developer.arm.com/documentation/ddi0487/fc/.
  - Warning: 8248 pages.