1. Computer architecture.
   * What main parts do modern computers include?
   * Explain the stored program concept and how a computer executes a program.
   * What is computer architecture? What is computer micro-architecture?
   * What instruction set architectures do you know?
   * What are performance challenges of modern computers?
2. Integer data formats and operations.
   * What is a byte and what is a machine word? What is byte ordering (which ones do you know)?
   * Describe unsigned integer format and 2's complement signed integer format.
   * How unsigned and signed values are converted to decimal numbers?
   * How unsigned values are converted to signed and vice versa?
   * What is the difference between integer sign- and zero-extension?
   * What is the difference between arithmetical and logical shift?
3. Floating-point format.
   * Why floating-point format is needed? Name the standard that covers it.
   * Explain the floating-point format (sign, exponent, fraction). Describe single and double precision.
   * What is implicit 1. in fraction? Why exponents are biased (and what is bias)?
   * Explain how the following value types are encoded: zero, normalized number, denormalized number, infinity, NaN.
   * What are floating-point overflow and underflow?
   * How does addition of two floating-point numbers works (the main steps)?
   * How floating-point format is supported in RISC-V (registers, main instructions)?
4. ISA and assembler language.
   * What is instruction set architecture (ISA)?
   * Give definitions and examples of the following ISA types: RISC, CISC, and VLIW.
   * What are machine code, assembly language, and assembler? What tool converts machine code to assembly language?
   * Describe of the structure of an assembly program (when in text) and its memory layout (when in machine code).
   * Describe main assembly directives (.text, .data, .align, .space, etc.). What else do you know?
5. RISC-V.
   * Brief history and advantages of the RISC-V ISA. Design principles of RISC-V.
   * List main RISC-V registers and main instruction types with examples.
   * What is program counter (PC)? What RISC-V instruction can be used to read its value?
   * Briefly describe 6 types of RISC-V instruction encodings (R-type, I-type, etc.).
   * Explain immediate addressing, register addressing, base addressing, and PC-relative addressing.
6. RISC-V assembly programming.
   * Give a definition of a register. What is the difference between registers and memory?
   * How you you swap values of two registers without using a temporary registers?
   * Give an example of a logic and arithmetical shift instruction. Explain the differences.
   * What load and store instructions do you know? Explain the difference between the lh and lhu instructions.
   * What control-transfer instructions do you know?
   * Explain the idea of pseudoinstructions. Give examples of RISC-V pseudoinstructions.
   * Explain the idea of macros. When would you use macros? How to reuse macros defined in other .s files?
7. Functions and stack.
   * What is a function? What are caller and callee?
   * How functions are implemented in assembly language? Describe what exception are performed by a function call.
   * Explain the idea of return address and jump-and-link instructions?
   * What are stack, stack pointer, stack (function) frame, and frame pointer? What is stored in the stack?
   * Explain the idea of caller- and callee-saved registers (give examples of such registers).
8. Interrupts and exceptions.
   * What is an interrupt and what is an exception? What RISC-V exceptions do you know?
   * What is the role of Control and Status Registers (CSRs) in handling exceptions?
   * What system instructions do you know?
   * What happens when an exception occurs (how the CPU handles the event)?
   * What is an exception handler? What actions does it perform? How does the CPU know how to call a handler?
   * What is a system call and how does it work?
9. Memory-mapped I/O (MMIO).
   * How I/O devices are connected to CPU and managed (control, data, and status signals)?
   * Explain the idea of Memory-Mapped I/O (MMIO).
   * Explain the idea of Direct Memory Access (DMA).
   * Explain difference between Interrupt-Driven I/O and Polling?
   * What is a device driver?
10. Pipelining.
    * List the 5 stages and give brief descriptions for them.
    * What pipeline hazards are? List the types of hazards and the ways to prevent them (with brief definitions).
    * Give an example of a hazard situation and how it can be handled.
    * What is branch prediction is needed for? How does it work?
11. Caches.
    * Describe how caching mechanism works (block, index, tag, valid bit, dirty bit).
    * Give the definition of associativity (direct-mapped, set associative, fully associative).
    * What is the difference between write-through and write-back?
    * What is replacement policy (what type of policy do you know)?
    * How many cache levels are typical for modern processors?
    * What problem can caches create for multicore processors?
12. Virtual Memory.
    * What is virtual memory (vs. physical memory)?
    * How does address translation work?
    * What is a page table and what information does it contain?
    * What is TLB and why is it needed? That is a TLB miss and how is it handled?
    * What is a page fault?
    * How does memory protection work?
13. Thread-level parallelism.
    * Why do we need thread-level parallelism? What are the challenges of parallel programming?
    * What is Amdahl’s Law?
    * Briefly describe how multi-threading works with: hardware multithreading (hyperthreading), multicore, multiprocessors.
    * What are context and context switch?
    * What is memory coherency problem?
14. Multiple issue processor. Data-level parallelism. Domain-specific architectures.
    * Explain the ide of multiple issues and superscalar microprocessors.
    * How do static multiple issue and dynamic multiple issue work? What is speculation?
    * What are SISD, SIMD, MISD, and MIMD?
    * Summarize the idea of SIMD. How does it help improve performance? Give examples of the SIMD approach in modern computers.
    * Why do we need domain-specific processors? Main principles of modern DSAs. Give an example of a DSA processor.
15. Optimizations.
    * Goal of optimizations? Algorithmic optimizations vs. compiler optimizations (advantages and limitations)?
    * How to assess performance?
    * What optimizations do you know?
    * How does the loop unrolling optimisation work (how it improves performance)?