Grading scale: 00–25: insufficient 26–31: sufficient 32–38: satisfactory
39–44: good 45–50: very good

The use of examination aids (e.g., calculators) is prohibited. Answers can be given in German or English. Please refrain from using lead pencils and red ink pens.

Some questions specify an approximate sentence count for your answer. This should give you an idea of how much detail is expected, and is *not* a hard limit. Avoid giving a two-paragraph answer to a question with "approx. 2 sentences."

Matr. number: **Last name:**

---

1. (10 points) **Logic Design:** A full-adder is
   (a) State the truth table for a 1-bit full adder.
   (b) Draw the gate logic of the 1-bit full adder.
   (c) Use the 1-bit full adder to build a 4-bit full adder. Draw the circuit on module level.
   (d) How can this 4-bit full adder be used for subtraction? What steps are needed?

A	B	Cin	S	Cout

2. (10 points) **Pipelining:**
   (a) Explain the motivation and concept of pipelining based on the following example. You have a single cycle machine with a critical path of 600ps. You are able split the path into three pipeline stages of 200ps each. What does it mean to insert a pipeline stage? Explain how this leads to a speedup. How much is the speedup (assuming a full and ideal pipeline)?
   (b) Assume the three pipeline stages `IF`, `ID` (includes reading the operands from the register file), and `EX`. This pipeline executes the three instructions below. Explain the problem arising in the pipeline with this code snippet. Explain two approaches to resolve this problem.
   (c) Complete the diagrams below for both approaches and show which pipeline operation is performed at what clock cycle for each instruction of the code snippet. Assume an empty pipeline before starting the execution. State the total number of needed clock cycles for the execution of the code snippet for both approaches.
   (d) Explain the concept of *Scoreboarding* and discuss why its usage for pipelining.

```
ADDI x2, x1, 0  # I1
ADDI x3, x1, 5  # I2
ADDI x4, x3, 3  # I3
```

→

3. (10 points) **Assembly:**
   (a) Explain the RISC-V instructions `JAL` and `JALR` and describe what they do internally in the CPU. What is the difference? Where are these instructions used (provide one application for each of the two instructions)?
   (b) Transform the following C-code to RISC-V assembly. All local variables of the C-code **must** be allocated on the stack. The RISC-V calling convention must be followed. The assembly startup code including the initialization of the stack is provided below. Write the assembly code for the two functions at the foreseen locations.

```c
int times3(int* p) {
  return *p * 3;
}

int main() {
  int a = 3;
  return a + times3(&a);
}
```

Assembly Reference

```
LW    rd,imm(rs1)
SW    rs1,imm(rs2)
ADD   rd,rs1,rs2
ADDI  rd,rs1,imm
SUB   rd,rs1,rs2
```

```
_start:
  ADDI sp, zero, 0x700
  JAL ra, main
  EBREAK

times3:

main:
```

4. (10 points) **TCP/IP & Ethernet:**
   (a) (2 points) The TCP/IP networking model consists of four layers: the *Link layer*, the *Internet layer*, the *Transport layer*, and the *Application layer*. Briefly describe the role each of the four layers plays in the communication stack. *(Use approx. 1-2 sentences per layer.)*
   (b) (2 points) Some Link layer technologies employ *checksums*. Without going into detail on any particular algorithm[1], describe the basic concept of a checksum. What does the sender do? What is transmitted? How does the receiver handle this? What problems are avoided by using a checksum? *(Use approx. 3-4 sentences.)*
   (c) (3 points) In the lecture, we covered two Link layer technologies – *Ethernet* and *Wi-Fi*. Characterize and compare these two Link layer technologies. How do they differ? *(Use approx. 4-5 sentences.)*
   (d) (3 points) In an Ethernet network connected by a central hub, what is the role of a MAC address? If a frame is received by a device, how does it process the addressing information? Describe the role of the special MAC address `FF:FF:FF:FF:FF:FF`. *(Use approx. 4-5 sentences.)*

[1] We don't expect you to explain how a CRC32 calculation works

5. (10 points) **HTTP & TCP:**
   (a) (2 points) The *HyperText Transfer Protocol* is a foundational protocol of modern Internet, allowing your web browser to retrieve pages. Requests can be made using one of many *methods* – explain how a `GET` and `HEAD` request for the same resource differ. Give an example where a `HEAD` request might be useful. *(Use approx. 2-3 sentences.)*
   (b) (3 points) Explain the difference between read-only ("idempotent") requests, and requests that may modify resources. Could a `GET` request be used for a button that adds items to a shopping cart? How would this be different from using a `POST` request for the same purpose? *(Use approx. 3-4 sentences.)*
   (c) (3 points) Even though HTTP/1.1 can make multiple requests over a single TCP connection, it still suffers from inefficiencies in loading modern web pages. This was a factor in the development of HTTP/2. Explain the issue, give a scenario where this matters, and briefly sketch how HTTP/2 addresses it. *(Use approx. 4-6 sentences.)*
   (d) (2 points) If a packet in a TCP connection is lost, this introduces so-called *TCP head-of-line blocking delay*. Explain this phenomenon, and how it affects HTTP/2. How does HTTP/3 avoid it? *(Use approx. 3-4 sentences.)*

Good Luck!