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) **FSM in SystemVerilog:** Consider the finite state machine specified in SystemVerilog below:

   (a) The table to the right specifies the output logic. Specify their respective logical formulas.

   (b) Show the corresponding ASM diagram of the state machine.

   (c) Show the structural diagram of the FSM featuring logic blocks, flip flops, and wires.

   (d) Name the type of state machine in this example. What is the name of the second type of FSM and explain the difference between them.

   (e) Change one value in the `out_o` column to create the other type of FSM. The resulting FSM does not need to be functionally equivalent.

state_p [1]	state_p [0]	in	out_o [1]	out_o [0]
0	0	0	1	0
0	0	1	1	0
0	1	0	0	0
0	1	1	0	0
1	0	0	1	1
1	0	1	1	1
1	1	0	0	0
1	1	1	0	1

```
module state_machine(
  input logic clk_i,
  input logic reset_i,
  input logic in_i,
  output logic [1:0] out_o
);
  logic [1:0] state_p, state_n;

  always @(posedge clk_i or posedge reset_i) begin
    if (reset_i) state_p <= 2'b00;
    else         state_p <= state_n;
  end

  always @(*) begin
    case (state_p)
      2'b00:   state_n = state_p + 2;
      2'b01:   state_n = 2'b11;
      2'b10:   state_n = state_p - 1;
      default: state_n = 0;
    endcase
  end

  always @(*) begin
    // output logic is left out (refer to the table above)
  end
endmodule
```

→

→

2. (10 points) **Memory and Cache:** Assume a directly-mapped data cache with a total size of 64 bytes, organized in 8 blocks, and 256 bytes of byte-addressable main memory. The latency for accessing the main memory is 200ns and the latency for accessing the cache is 20ns.
   (a) Why are caches used in modern processors, *i.e.*, what problem do they solve?
   (b) Name and explain the two types of locality that caches exploit.
   (c) Sketch the directly-mapped cache and explain how a cache access to the address 0x56 is performed. What checks are performed on which data and what are the expected values for a cache hit?
   (d) Assume the following piece of code being executed on a system. The local variables `i` and `sum2` are kept in registers. The data cache is empty at the beginning of the execution. What is the total latency for accessing *all* elements? For every access, state whether it is a cache hit `H` or miss `M`.

```
// Located at memory address 0x56
char array[4] = {1, 2, 3, 4};

int sum2 = 0;
for(int i = 0; i < 8; ++i)
  sum2 += array[i % 4]
```

→

3. (10 points) **Assembly:**
   (a) What is a calling convention and why is it needed? Explain what a calling convention covers.
   (b) Transform the following C-code to RISC-V assembly. All local variables of the C-code **must** be allocated on the stack. The global variable `g` is located at address `0x7F0`. 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
// Located at memory address 0x7F0
int g;

int addglobal(int* p) {
  return *p + g;
}

int main() {
  int a = 3;
  g = 4;
  return addglobal(&a);
}
```

Assembly Reference

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

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

  addglobal:
```

```
  main:
```

4. (10 points) **Ethernet switching:**
   (a) (2 points) Modern Ethernet networks are built around central devices called *switches*. What is the basic functionality of a switch? *(Use approx. 2-3 sentences.)*
   (b) (3 points) In the past, Ethernet networks were built using *hubs* instead. How were they different from today's switches? What problems did this cause? *(Use approx. 3-4 sentences.)*
   (c) (3 points) Professional switches offer *Virtual LAN* functionality. Describe how this works. *(Use approx. 3-4 sentences.)*
   (d) (2 points) If three switches (A, B, and C) are each connected to each other, this causes a problem. Describe why this is, and give a workaround. *(Use approx. 2-3 sentences.)*

5. (10 points) **DNS & NAT:**
   (a) (4 points) The *Domain Name System* allows human-readable hostnames to be translated into IP addresses. Explain step-by-step how the hostname `secure.web.example.org` is resolved at the application layer. What devices beside the user's are involved? How does the user device know who to contact? *(Use approx. 4-6 sentences.)*
   (b) (2 points) Explain the concept of *Network Address Translation (NAT)*. What issue with IPv4's design is it working around? *(Use approx. 2-3 sentences.)*
   (c) (2 points) When querying DNS for data, that data has to somehow make its way back to the client. If a client at *192.168.0.1* sends a DNS query to *1.1.1.1* on UDP port *53*, how does a NAT router process this datagram? What fields are modified? When the DNS server sends a response, how is it delivered back to the client? *(Use approx. 3-4 sentences.)*
   (d) (2 points) Is NAT still necessary when communicating via IPv6? Explain why, or why not. *(Use approx. 2-4 sentences.)*

Good Luck!