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 fundamental building block in digital logic design, used to perform arithmetic operations in binary arithmetic. It’s capable of adding three binary digits: A, B, and a carry-in (Cin), and produces two outputs: a sum (S) and a carry-out (Cout).
   (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) What is the purpose of a *Reorder Buffer* in an out-of-order pipeline?

```
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.

## Assembly Reference

| 31–25 | 24–20 | 19–15 | 14–12 | 11–7 | 6–0 | |
|---|---|---|---|---|---|---|
| imm[11:0] | | rs1 | 010 | rd | 0000011 | `LW rd,imm(rs1)` |
| imm[11:5] | rs2 | rs1 | 010 | imm[4:0] | 0100011 | `SW rs2,imm(rs1)` |
| 0000000 | rs2 | rs1 | 000 | rd | 0110011 | `ADD rd,rs1,rs2` |
| imm[11:0] | | rs1 | 000 | rd | 0010011 | `ADDI rd,rs1,imm` |
| 0100000 | rs2 | rs1 | 000 | rd | 0110011 | `SUB rd,rs1,rs2` |

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

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

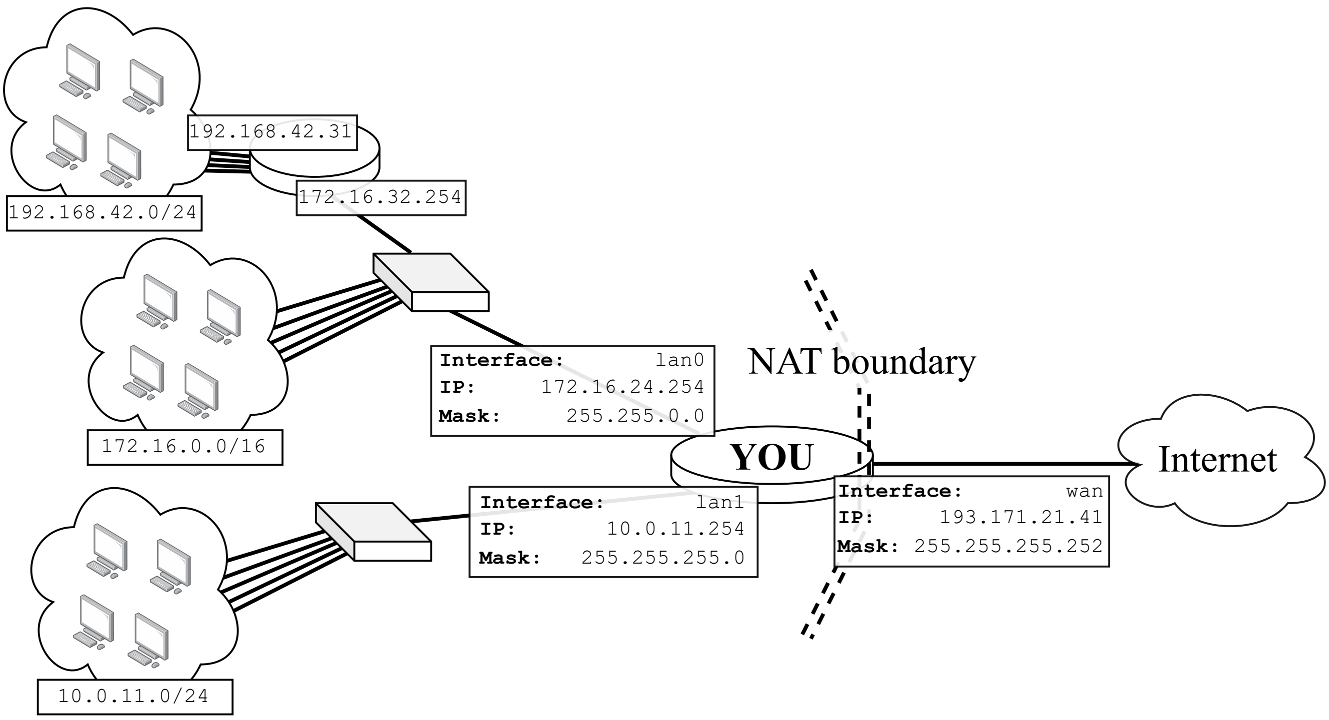
```
_start:
  ADDI sp, zero, 0x700
  JAL ra, main
  EBREAK

times3:
```

```
main:
```

→

4. (10 points) **Routing & NAT:**

   You are acting as a Network Address Translation (NAT) router in the pictured network.




   Your default gateway for any networks not listed is `193.171.21.42`. Your task is to correctly route packets, and to apply NAT to allow devices to communicate with the internet.

   (a) (6 points) On the next page, you will find multiple incoming IPv4 packets, in the order of receipt. Fill in the missing values in the corresponding forwarded packet:

   - Which **network interface** to use to forward the packet (*lan0*, *lan1*, or *wan*).
   - Which **source** and **destination IP address** the forwarded IP packet should have.
   - Which **source** and **destination port** the contained TCP segment should have.
   - What **next hop** (IP address) the Ethernet frame should be forwarded to.

   (b) (4 points) After transmitting the above packets, you receive an incoming packet from source address `54.239.28.85` and source port `80`.

   - What **source address** and **source port** will the packet have after translation?
   - What **destination address** and **destination port** will you set?
     How do you determine this?

| INCOMING | FORWARDED | |
|---|---|---|
| **IPv4:** | **IPv4:** | **TCP:** |
| **Source:** 10.0.11.116 | **Source:** ______ | **Source:** ______ |
| **Dest.:** 172.16.24.180 | **Dest.:** ______ | **Dest.:** ______ |
| **TCP:** | | |
| **Source:** Port 51333 | | |
| **Dest.:** Port 53979 | **Next hop:** ______ | **Interface:** ______ |

| INCOMING | FORWARDED | |
|---|---|---|
| **IPv4:** | **IPv4:** | **TCP:** |
| **Source:** 10.0.11.116 | **Source:** ______ | **Source:** ______ |
| **Dest.:** 172.16.32.129 | **Dest.:** ______ | **Dest.:** ______ |
| **TCP:** | | |
| **Source:** Port 51333 | | |
| **Dest.:** Port 53979 | **Next hop:** ______ | **Interface:** ______ |

| INCOMING | FORWARDED | |
|---|---|---|
| **IPv4:** | **IPv4:** | **TCP:** |
| **Source:** 192.168.42.3 | **Source:** ______ | **Source:** ______ |
| **Dest.:** 54.239.28.85 | **Dest.:** ______ | **Dest.:** ______ |
| **TCP:** | | |
| **Source:** Port 55522 | | |
| **Dest.:** Port 80 | **Next hop:** ______ | **Interface:** ______ |

| INCOMING | FORWARDED | |
|---|---|---|
| **IPv4:** | **IPv4:** | **TCP:** |
| **Source:** 172.16.15.91 | **Source:** ______ | **Source:** ______ |
| **Dest.:** 54.239.28.85 | **Dest.:** ______ | **Dest.:** ______ |
| **TCP:** | | |
| **Source:** Port 55522 | | |
| **Dest.:** Port 80 | **Next hop:** ______ | **Interface:** ______ |

| INCOMING | FORWARDED | |
|---|---|---|
| **IPv4:** | **IPv4:** | **TCP:** |
| **Source:** 192.168.42.3 | **Source:** ______ | **Source:** ______ |
| **Dest.:** 54.239.28.85 | **Dest.:** ______ | **Dest.:** ______ |
| **TCP:** | | |
| **Source:** Port 55522 | | |
| **Dest.:** Port 80 | **Next hop:** ______ | **Interface:** ______ |

| INCOMING | FORWARDED | |
|---|---|---|
| **IPv4:** | **IPv4:** | **TCP:** |
| **Source:** 172.16.15.91 | **Source:** ______ | **Source:** ______ |
| **Dest.:** 54.239.28.85 | **Dest.:** ______ | **Dest.:** ______ |
| **TCP:** | | |
| **Source:** Port 55523 | | |
| **Dest.:** Port 80 | **Next hop:** ______ | **Interface:** ______ |

→

5. (10 points) **TCP, DNS & HTTP:**

   (a) (3 points) A client establishes a TCP connection to the server at `https://en.wikipedia.org`. HTTPS uses the well-known port `443`.

   *For each statement, check whether it is true or false.*
   +1 *point for correct choice,* −1 *point for wrong choice,* 0 *points for no choice.*

| **Statement** | **True** | **False** |
|---|---|---|
| The source port of any TCP packets the server sends will be `443`. | ◯ | ◯ |
| If two different clients connect to this server at the same time, they must use different source ports for their TCP packets. | ◯ | ◯ |
| The server chooses a unique destination port for each of its clients. The clients then send packets to the assigned port. | ◯ | ◯ |

   (b) (3 points) The ***D**omain **N**ame **S**ystem* allows the client to translate the hostname `en.wikipedia.org` to an IP address. Explain how this happens. What device(s) does the client communicate with? How does it discover the right device(s)? What queries are made? Focus on the Application layer; ignore any details of the other layers.

   (c) (4 points) *For each statement, check whether it is true or false.*
   +1 *point for correct choice,* −1 *point for wrong choice,* 0 *points for no choice.*

| **Statement** | **True** | **False** |
|---|---|---|
| The client needs to ensure that all of its packets take the same route to the server, to avoid them arriving out of order. | ◯ | ◯ |
| If a packet from the client is lost, the server notifies the client. The client then re-sends only that packet. | ◯ | ◯ |
| If a packet from the client is lost, the server notifies the client. The client then re-sends that packet and all subsequent packets. | ◯ | ◯ |
| Even if packets from the server arrive out of order, the client will be able to re-order them, and process them in the correct order. | ◯ | ◯ |

Good Luck!