Revised Division of Work for the Teaching Computer Application

Based on the feedback from our last meeting, we revised the division of the Teaching Computer Application (TCA). The goal of this new structure is to distribute the workload more evenly, ensure that each part has sufficient scientific depth, and define clear interfaces between the subprojects early on.

Instead of separating the project into simulation, user interface, and auxiliary tools, we now divide the work according to **different abstraction levels of the processor architecture**. This allows each subproject to focus on a coherent technical problem while still fitting naturally into the overall system.

Part A – Low-Level Digital Simulation

Description

This part focuses on the simulation of the basic digital elements that form the foundation of the Teaching Computer. It represents the lowest abstraction level of the system and provides reusable building blocks for the higher simulation layers.

Scope

- Implementation of basic logic gates (AND, OR, NOT, XOR, etc.)
- Construction of simple components from these gates, such as:
    - Flip-flops
    - Multiplexers
    - Adders
    - Registers
    - Memory cells
- Modeling of clocked behavior and signal changes
- Support for configurable word lengths (e.g. 10-bit and 16-bit)
- Clear specification of the input and output behavior of all components

Scientific Focus

This part allows an in-depth discussion of abstraction levels in digital simulation. In particular, it addresses questions such as how detailed a gate-level simulation needs to be for teaching purposes and how accuracy, complexity, and performance can be balanced in an educational simulator.

Interfaces

The low-level components provide well-defined interfaces (e.g. read/write operations, clock signals, ALU operations) to the higher-level simulation. There is no direct interaction with the user interface.

Part B – CPU Architecture and Control Simulation

Description

This part builds the complete processor model by combining the components from Part A. It describes the Teaching Computer at the architectural and micro-architectural level.

Scope

- Integration of registers, memory, ALU, and buses into a functioning CPU
- Implementation of the instruction cycle (fetch, decode, execute, memory access, writeback)
- Design and implementation of the control unit and microcode
- Definition and comparison of architectural variants, for example:
    - Different word lengths
    - Alternative instruction formats or addressing modes
    - Accumulator-based versus general-purpose register architectures
- Step-based execution and timing behavior

Scientific Focus

This part focuses on architectural design decisions and their impact on both functionality and teaching clarity. It also allows a comparison between different processor variants while keeping the same underlying components, which is particularly relevant in an educational context.

Interfaces

This subproject uses the component behavior provided by Part A and exposes the complete processor state (e.g. register values, current instruction, control signals, bus values) in a structured form. These data are then used by the visualization and interaction layer.

Part C – Visualization, Interaction, and Learning Support

Description

This part is concerned with how the internal processor state is presented to the user and how users interact with the system. The focus is on making the simulated behavior understandable and useful for students with different levels of prior knowledge.

Scope

- Visualization of registers, memory, buses, ALU, and control signals
- Animated representation of instruction execution and data flow
- Different levels of abstraction (e.g. simplified view for beginners and detailed view for advanced users)
- Learning-oriented features such as guided execution steps and explanations
- Assembly code input as a means of interaction, including basic validation and error feedback

Scientific Focus

The main emphasis is on the didactic design of technical visualizations. This includes decisions about which information should be shown, which details should be hidden, and how complex processes can be visualized without overwhelming the learner.

Interfaces

This part receives structured processor state data from Part B and sends user interaction commands (such as step, run, reset, or configuration changes) back to the simulation. It does not access low-level simulation details directly.

Interfaces and Overall Integration

The interaction between the three subprojects is defined as follows:

- **Part A → Part B:** component behavior and timing information
- **Part B → Part C:** complete processor state for each simulation step
- **Part C → Part B:** user-driven control and configuration commands

Defining these interfaces early helps to avoid ambiguities during implementation and allows the subprojects to be developed and tested independently.

Feedback from supervisor:

Dear students,
Thank you for joining yesterday's meeting and your revised division paper.
The following lines collect some remarks on your revised project division into subprojects and your approach to interfaces between the subprojects.
Overall, the partition of topics, workload, and research/scientific potential is now much more balanced.

- For **Part A**:
  - Scientific Focus - Include pre-existing simulations in 'related work.' They can guide your TCA implementation. By studying these and the technical components you want to simulate, you can make choices for your own implementation that are justified and well-informed.
  - Keep working on **interfaces (relevant to all parts)**. What data needs to be exchanged for a good simulation? How do you represent things like 'read-write operations' for a memory cell? (This will likely be a logic circuit with parameter input. Data at the interface could include a 1-bit read/write signal, n-bit memory data, n-bit data input, and n-bit output wires. You also need a functional description explaining how these data relate and some information on the instruction cycle, time, or time step.)
    The key question for the interfaces is: Do they contain everything I need to work independently on my subproject? If not, further specification will be needed.
- **Part B**: Here, the interface remarks apply, too.
  - Especially, the simulation needs to work seamlessly with the 'building blocks' from parts and simulate the timely behavior of the higher architecture levels based on the known (timely) behavior of the more basic components and the data provided at the interface.
  - Assessments like "which is particularly relevant ..." can be scrutinized. Be prepared to justify them by argumentation or research/literature.
- **Part C**:
  - You included supplementary features (guided execution steps, explanations), which is fine, yet their usefulness depends on the proper execution of the core tasks.

    - For a clear vision of the project, shift from the notions of showing information and hiding details to some degree to the concept of layers of abstraction. Abstraction 'hides details', too. However, abstraction does this by a clear principle. If you organize the 'hiding of details' going from most specific to most general, you can mirror the structure of the simulation: The most general functions of a computer rely on very basic mechanisms.
      For the visualization, there are two complementary dimensions of 'hiding' - the abstraction (hiding/showing internal processes) and the section of the system (presenting everything/a group of components/single component/...).
    - Regarding user inputs: Clarify that beside commands (control, configuration), users may change memory as well (programs/assembler, values, particular scenarios of system state).

All parts: While I agree with your general concept, **I don't think you can avoid an interface between parts A and C altogether**. The goal for parts A and B is a consistent, detailed, and precise simulation. To reveal the internal states and processes higher functionalities rely on, the UI in part C will additionally have to access the data of lower levels. Then, users can study the overall system at a high level, e.g., the instruction cycle. Later, they may switch to components or sub-components and watch their internal processes to understand the underlying mechanisms. One benefit of the comprehensive simulation is that it provides all the data necessary for this flexible visualization.
In the opposite direction (input), users will also affect all levels of the simulation. If they, for example, input a value into the memory, they will alter the state of logic gates/wires within flip flops, alter the state of memory cells, and alter the corresponding visualization (if the data is propagated via the interfaces properly).

Please **adapt your project description and problem statements** accordingly and send them to me next week.
If Professor Werner agrees, we will set a time and date for your **concept presentations at TUC**. Please tell me some days, maybe 2-3 weeks from now, that will work for you.

- The presentations should include a joint section. describing the overall project (TCA), with its objective, requirements, and scope, briefly.

- The main part will be the individual presentations. Include the role within the project (scope and limits, goals), the requirements, object of your work (what do you model/which phenomena do you visualize), state of the art/related work (if you don't want to re-invent the wheel: Where do you start? Or: Why are all pre-existing wheels not working for this project? :), and - eventually - your approach to the implementation, tools, and (realistic) plan for the remaining work.
- If you have questions or topics you are particularly interested in, you can emphasize them. Describe problems you encountered, or point to an area you are unsure about, to get more detailed feedback. We will discuss your presentations first and foremost to provide advice and improve your approach.

Kind regards, have a nice weekend,
Albrecht Stoye