SAP-1 Digital Logic Design Project Report Phase-1

GROUP NAME: AND-OR-NAUR

Group Members:

Lead	Abdullah Iqbal 26904
Co-Lead	Aaraiz Masood 26988
Member 1	Abdullah Ahmedani 25896
Member 2	Afnan Anwer Kiani 26900
Member 3	Adil Siddiqui 27147
Member 4	Ali Saeed 27153
Member 5	Ali Uzzama 26941
Member 6	Fatima Tanveer 25183

Contents

Chapter 1

Work Breakdown

The process of designing a Simple As Possible-1 (SAP-1) computer is a multi-faceted undertaking, necessitating a systematic breakdown of tasks to ensure comprehensive coverage and understanding of the underlying principles. The SAP-1, conceived as an educational tool for elucidating fundamental digital computer design concepts, demands a rigorous and methodical approach to its creation.

1.1 Design Considerations

1. **Conceptual Framework and Requirements Specification:**
    - Undertake the design with a meticulous definition of the conceptual framework and specification of requirements.
    - Determine critical parameters such as the word size, instruction set architecture, and memory size and addressing capabilities.
2. **Register Architecture Design:**
    - Design the accumulator register (AC) and general-purpose register (B register) with careful consideration.
    - Define roles and functionalities for any auxiliary registers if needed.
3. **Memory Subsystem Design:**
    - Specify the type of memory (e.g., RAM) and determine memory size and organizational structure.

- Design the memory addressing scheme for effective data retrieval and storage.

4. **Arithmetic and Logic Unit (ALU) Design:**
   - Define specific operations for the ALU (e.g., addition, subtraction, logical functions).
   - Intricately design the ALU circuitry to accommodate these operations.

5. **Control Unit Formulation:**
   - Define the instruction set and opcode structure for the control unit.
   - Create a control unit capable of interpreting instructions and generating necessary control signals.
   - Devise sequencing logic for orderly instruction execution.

6. **Instruction Register (IR) Design:**
   - Design the Instruction Register (IR) to house the current instruction under execution.

7. **Clock and Timing Circuitry Development:**
   - Specify the clock frequency and design circuitry for precise timing signals.
   - Orchestrate synchronous execution of operations within the computer.

8. **Input/Output (I/O) System Considerations:**
   - Consider input and output systems, including device selection and interface design.
   - Design circuitry to facilitate seamless input and output operations.

9. **System Interconnectivity Planning:**
   - Design the data bus and address bus for system interconnectivity.

- Plan communication pathways to facilitate coherent interactions between components.

10. **Component Implementation and Testing:**
    - Implement each component on a physical breadboard or using simulation tools.
    - Rigorously test each component in isolation to ensure functionality.
11. **System Integration:**
    - Integrate all designed components into a cohesive SAP-1 system.
    - Thoroughly verify the system's functionality.
12. **Debugging and Optimization:**
    - Iteratively debug and optimize the design.
    - Address identified issues and explore opportunities for performance or efficiency optimization.
13. **Comprehensive Documentation:**
    - Compile comprehensive documentation, including schematics, diagrams, and detailed explanations of the design process.
    - Provide a user manual or guide for individuals engaging with or studying the SAP-1 computer.

1.2 Design Steps

The proposed workflow delineated herein outlines a comprehensive and methodical approach for the systematic design of the Simple As Possible-1 (SAP-1) computer architecture. This well-considered sequence of steps endeavors to provide a rigorous framework that encompasses the intricacies of digital computer design, ensuring a holistic exploration of each facet of the SAP-1 system. The meticulous progression through these stages not only underscores the complexity inherent in computer architecture but also serves to establish a foundation for an in-depth and scholarly exploration of the design process.

1. **Define Specifications**
    - Define the instruction size to be processed by the computer system.
    - Determine instruction set architecture.
    - Define memory size and addressing capabilities.
2. **Register Design Stage**
    - Design Accumulator.
    - Design general purpose register.
3. **Memory Design Phase**
    - Specify memory type.
    - Specify memory size.
    - Design memory addressing scheme.
4. **ALU Design Phase**
    - Specify operations the ALU will support.
    - Design the ALU circuitry according to functional requirements.
5. **Control Unit Design Phase**
    - Define the instruction set and opcode structure.
    - Create a control unit that interprets instructions and generates control signals.
    - Design the sequencing logic for instruction execution.
6. **Instruction Register Design**
    - Design the instruction register to hold the current instruction being executed.
7. **Clock Design Phase**
    - Specify the clock frequency.
    - Design the clock circuitry and timing signals.

8. **Component Testing**
    - Implement each designed component on a breadboard or using simulation tools.
    - Test each component individually for functionality.
9. **Integration**
    - Connect all components together on a single platform.
    - Verify the integrated system's functionality.
10. **Debugging and Optimization**
    - Identify and fix any issues in the design.
    - Optimize the design for performance or efficiency.
11. **Documentation**
    - Provide a user manual or guide for anyone working with or studying the SAP-1 computer.

Chapter 2

Inclusions and Exclusions

The SAP-1 design, being inherently flexible, is amenable to diverse modifications to cater to a spectrum of functionalities. In our endeavor to tailor the SAP-1 architecture to specific purposes, strategic inclusions and exclusions have been carefully implemented. The refined design incorporates bespoke functionalities that enhance its capabilities, while simultaneous exclusions ensure a focused and optimized computational framework. These meticulous adjustments reflect a cognizant approach to the adaptability of the SAP-1 computer, allowing for its alignment with distinct computational requirements. The deliberate choices in function inclusions and exclusions underscore the versatility inherent in the SAP-1 design, transforming it into a dynamic and purposeful computing platform.

2.1 Inclusions

The meticulously devised SAP-1 computer encompasses a nuanced architectural design, featuring an 8-bit framework enhanced by various key components. The array-based memory system is complemented by the strategic inclusion of a program counter, an instruction register, and an accumulator. Notably, the instruction register is configured with a 4-bit architecture, aligning with a precise encoding scheme for instructions. Simultaneously, the program counter, governing the sequence of operations, is designed with a 4-bit structure, optimizing its capacity for sequential instruction management. The Arithmetic and Logic Unit (ALU), a pivotal component in computational processes, boasts a 9-bit architecture, providing extended capability

for diverse arithmetic and logical operations. Complementing this, the accumulator, responsible for accumulating results, is configured with an 8-bit architecture, ensuring compatibility and efficiency in data processing. This refined architectural composition, marked by meticulous considerations of bit sizes across key components, not only aligns with a specific operational focus, particularly centered on the execution of addition operations but also stands as a testament to precision and purpose within the realm of digital computation.

2.2 Exclusions

The design of our SAP-1 computer is distinguished by strategic exclusions that define its operational scope. Specifically, the computational framework is intentionally constrained to solely execute addition operations, a deliberate exclusion that streamlines the system for efficiency and precision in arithmetic processing. Furthermore, the omission of the capability to jump to a given address in the memory ensures a focused and straightforward execution flow, aligning with a specific use case that prioritizes sequential and additive computational tasks. These exclusions, thoughtfully incorporated into the design, underscore a commitment to tailored functionality and purposeful limitations, reflecting a strategic approach to computational architecture that prioritizes clarity, reliability, and optimal performance in the specified operational domain.

Chapter 3

Functional Requirements

3.1 Requirements for each component

1. **Program Counter**

   - The ability to read the current content of the PC and write new values to it, allowing for the sequential progression of the program counter.
   - An increment operation to move the program counter to the next sequential memory address, pointing to the next instruction in the program.
   - Program counter requires a clock signal, D-Flip flops, And and Or gates. At every rising edge of clock, value of program counter is incremented by 1.

2. **Instruction Register**

   - The instruction register have a storage capacity of 2 bits to hold the entire instruction being executed. Leading to 4 types of instruction added in SAP-1.
   - The ability to read from and write to the instruction register is essential for loading instructions from memory and storing intermediate results.
   - The instruction register needs to be connected to the Arduino Uno controller for the transfer of instruction data between the register and CPU or ALU.

3. **Accumulator**

   - The accumulator have storage capacity of 9 bits to hold the result of arithmetic and logic operations.
   - The ability to read from and write to the accumulator is crucial for loading initial values, storing results, and performing operations.
   - The accumulator needs to be connected to the data bus for the transfer of data between the accumulator and other components in the CPU, arithmetic logic unit (ALU).
   - Clock Input: The accumulator should be able to latch data at the appropriate time, synchronized with the CPU clock.
   - Clearing and Initialization: The ability to clear the contents of the accumulator, either manually or through control signals, is important for preparing the accumulator for new operations.

4. **Arithmetic Logic Unit**

   - ALU takes two inputs of eight bits. ALU make use of 8 full adder circuits to add two inputs resulting an output of 9 bits. ALU makes use of accumulator to store or read bits. One of input bits comes from accumulator and other from controller(Arduino). The sum is calculated by ALU and resulting bits are stored back to accumulator.