SAP-1

And-Or-Naur

Abdullah Iqbal (26904)
Aaraiz Masood (26988)
Afnan Anwer Kiani (26900)
Ali Uzzama (26941)
Abdullah Ahmedani (25896)
Adil Siddique (27147)
Fatima Tanveer (25183)
Ali Saeed (27153)

Contents

Introduction

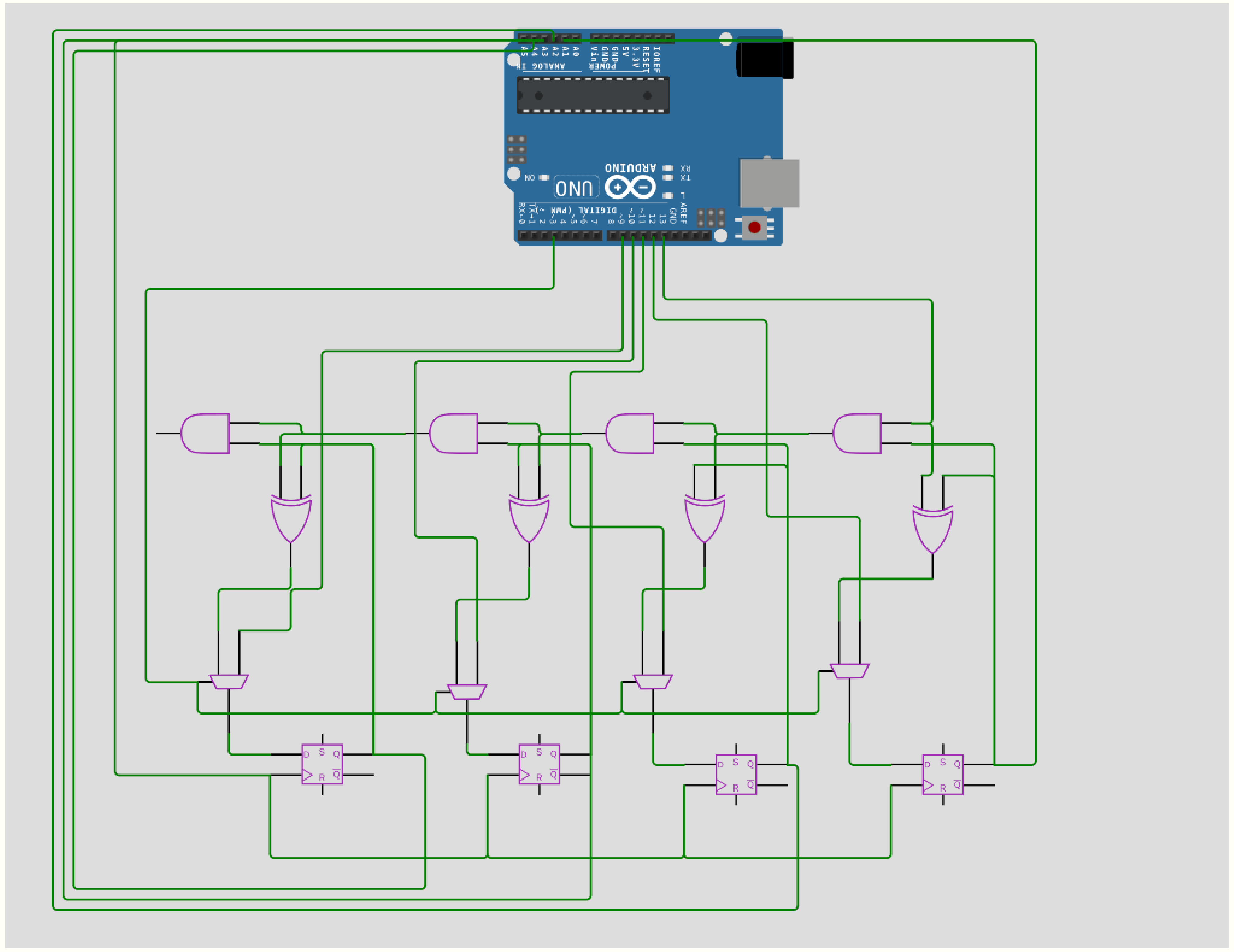
- The Simple As Possible-1 (SAP-1) architecture is a keystone in the field of digital logic design, providing a basic investigation into the complexities of early computer design.
- Based on the Von-Neumann Architecture, integrating key principles of this model.
- The SAP-1 framework simplifies intricate digital systems, presenting them in an understandable and educational format.

Components

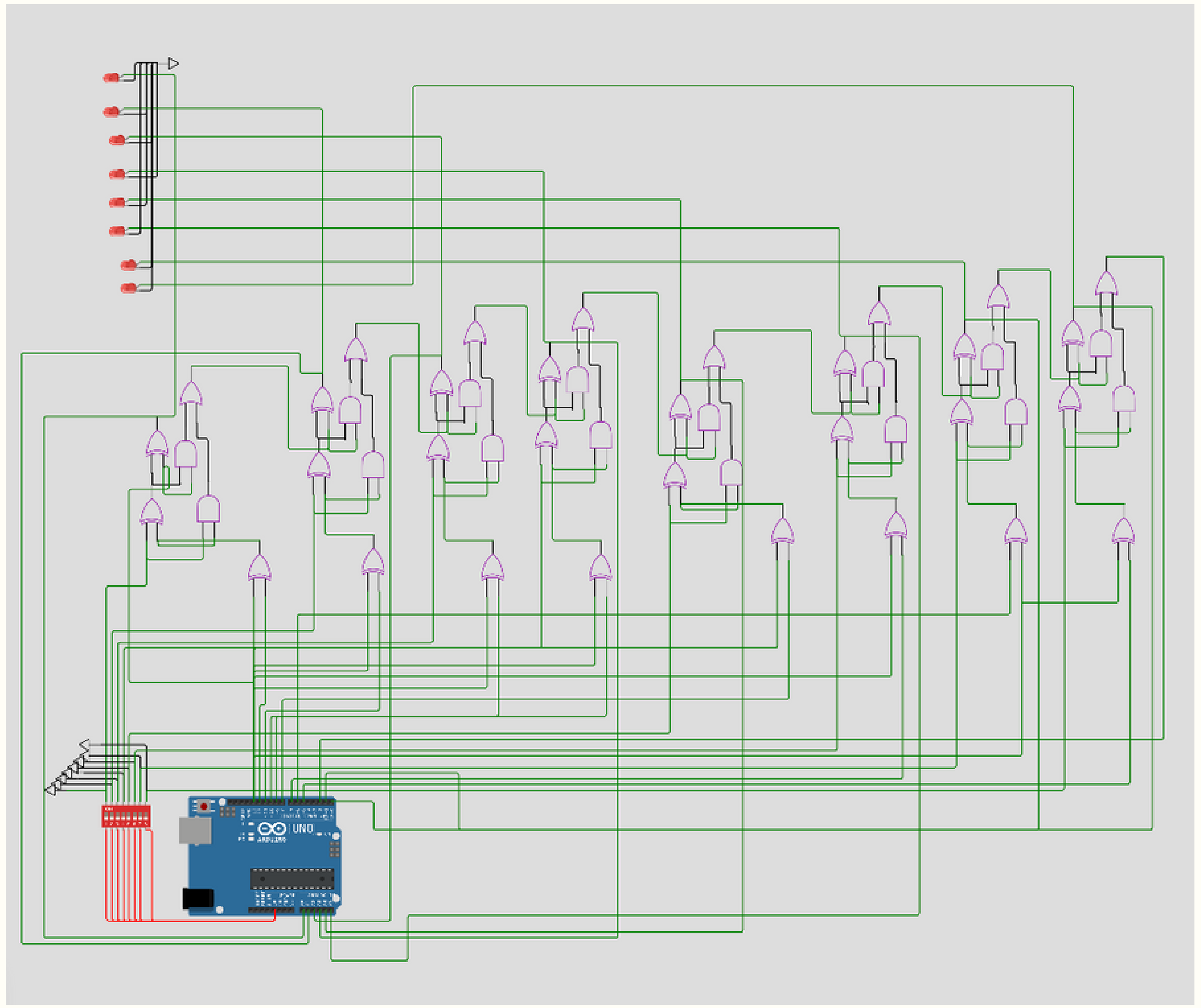
- Program Counter
- ALU
- IR
- Accumulator
- B-register
- Additional register for Comparator
- Comparator
- 16 by 1 MUX
- RAM

Program Counter

- **4-bit Binary Counter**: The Program Counter is a 4-bit binary counter capable of counting from 0 to 15 and then rolling back to 0, following the binary counting sequence.
- **Counter with Parallel Load**: The counter includes a parallel load feature, which allows for setting the counter to a specific value from the 4 bits, rather than just incrementing.
- **D Flip-Flop Utilization**: D Flip Flops are employed within the counter to hold the current count value, ensuring stable storage of the binary count.
- **Multiplexer for Parallel Load**: A multiplexer (mux) is integrated into the design to manage the parallel load operation; when the mux is set to high, it enables the parallel load function.
- **Counting Sequence and Reset Mechanism**: The counter starts its count based on the input provided through the parallel load function and will count up from that point; once it reaches the maximum count of 15, it resets back to 0.



ALU

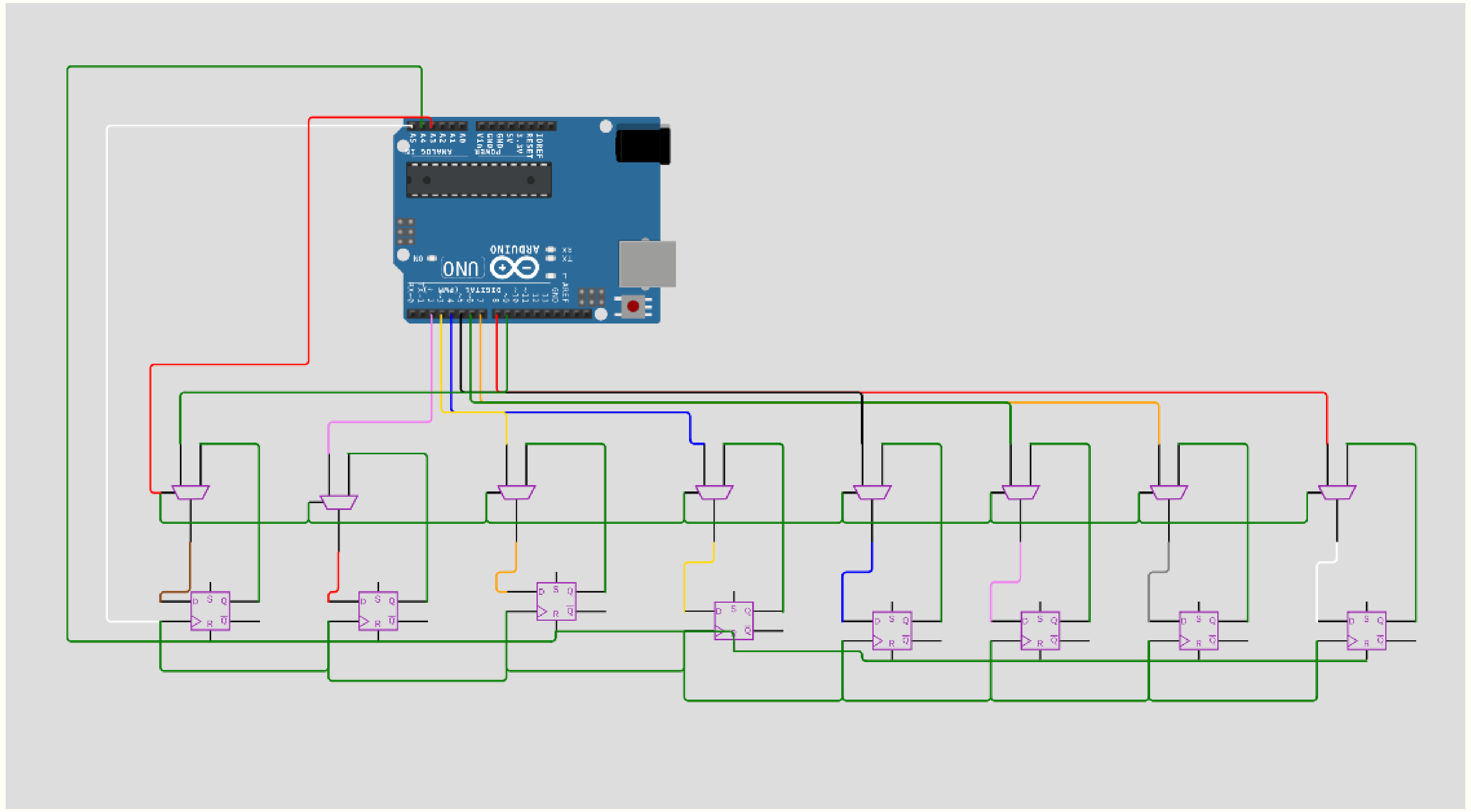


- **Dual Functionality with XOR Gates:** The 8-bit adder-subtractor in the SAP 1 ALU uses XOR gates for performing both addition and subtraction (via 2's complement method), allowing for flexible arithmetic operations.
- **Carry-Out Calculation with AND Gates**: AND gates within the unit are responsible for accurately calculating carry-outs, a crucial aspect of multi-bit arithmetic.
- **Sum Output via OR Gates**: The final sum of arithmetic operations is derived using OR gates, which consolidate the individual outcomes of addition and carry processes.

- **Versatile Computation:** The integration of basic logic gates like XOR, AND, and OR enables the ALU to efficiently perform a range of arithmetic tasks, enhancing the SAP 1's computational versatility.
- **Reliable Result Accuracy:** The design ensures clear and precise carry propagation through the gates, leading to accurate and reliable arithmetic results, thereby bolstering the overall computational performance of the SAP 1 ALU.

- **8-bit Instruction Handling:** The Instruction Register receives an 8-bit output from the RAM, which is then relayed onto an 8-bit bus system.
- **Controller/Sequencer Interface**: The separation of the instruction into nibbles facilitates the Controller/Sequencer in executing the appropriate sequence of operations based on the decoded instructions.

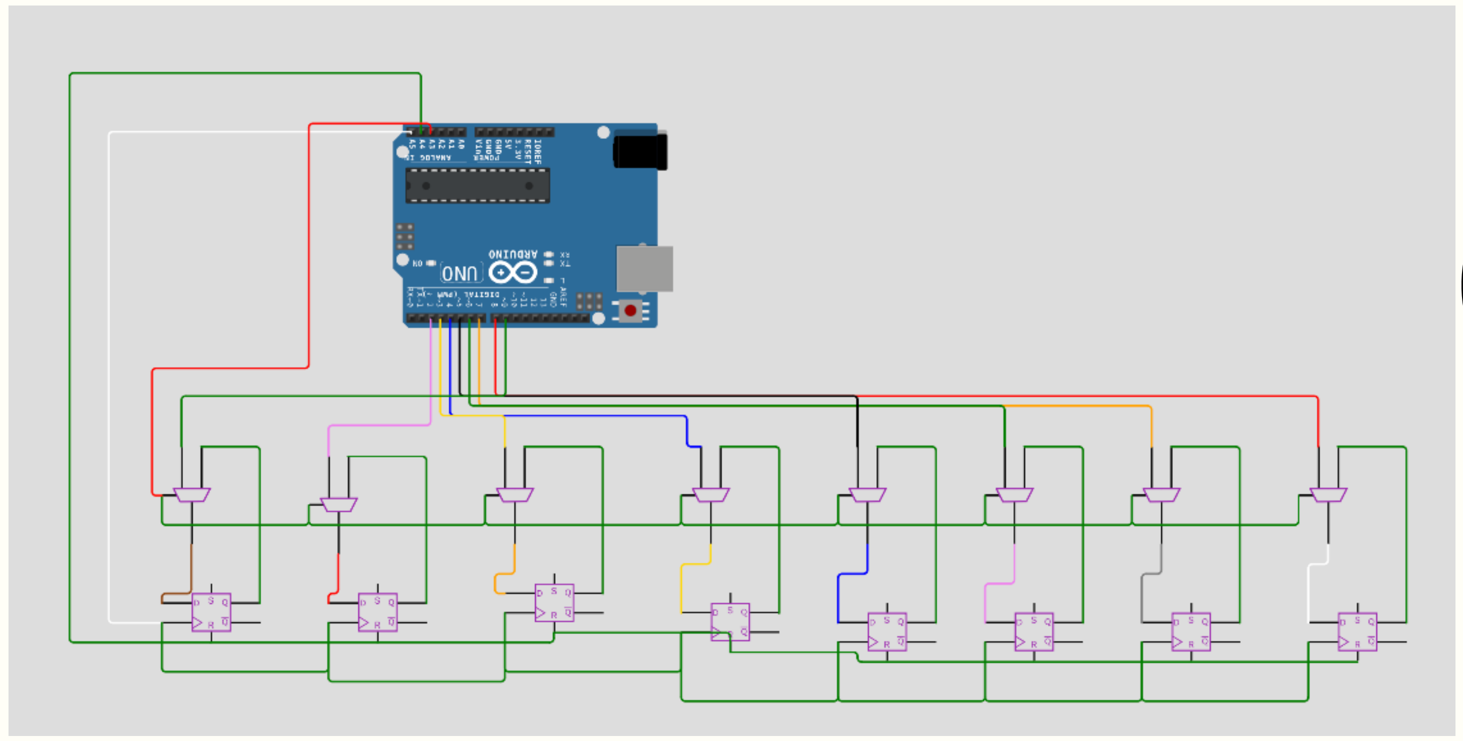
Instruction Register



- **Nibble Division:** Its primary role is to divide the 8-bit instruction into two separate nibbles - the upper nibble and the lower nibble.
- **Instruction Decoding:** The upper nibble, consisting of the most significant 4 bits, is sent to the Controller/Sequencer where the instruction set is decoded.
- **Memory Location Identification:** The lower nibble, made up of the least significant 4 bits, is directed to the select line of multiple 16x1 multiplexers (muxes) to determine the specific memory location to be accessed in RAM.

Accumulator

- **Shift Register with Parallel Load:** The circuit includes an 8-bit shift register capable of parallel loading, allowing for simultaneous data input across all bits.
- **Data Storage from RAM**: The register is designed to store values fetched from RAM, positioning it as a primary stage in data handling before processing by the ALU.
- **Controlled Data Accessibility**: The use of control signals in the circuit ensures that the register's contents are selectively placed on the bus, managing the accessibility of the accumulator's data for subsequent operations.

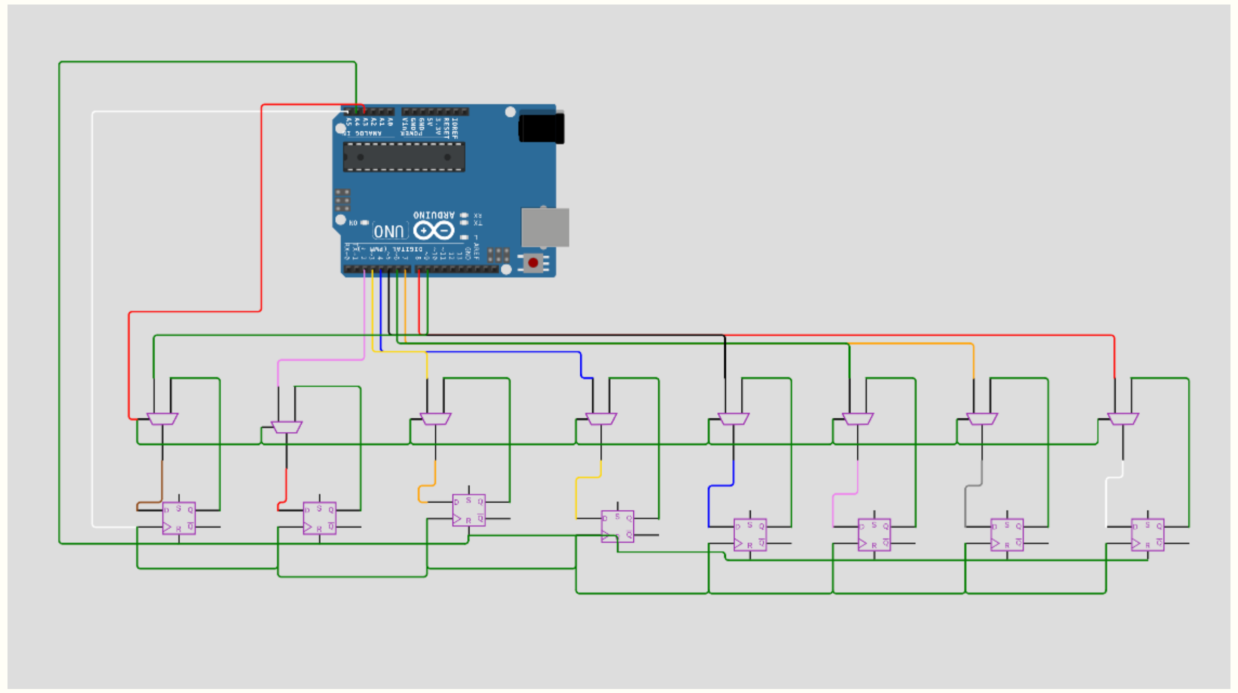


- **Multiplexer-Controlled Loading**: The state of the multiplexer (mux) dictates the loading behavior; when the mux signal is high, the register performs a parallel load of data. When the mux is low, the register retains its previous value.
- **Accumulator after ALU Operations**: Following ALU operations, the register can store the resulting value in the accumulator, serving as a temporary holder for intermediate results.

B - Register

- **Accumulator Update:** Post-operation, the ALU's result is stored back into the accumulator, updating it with the new value derived from the computation.

- **Conditional B Register Load**: The B register loads data from the bus when its load signal is in the high state, indicating readiness to receive data.
- **Operand Storage**: The B register serves as a storage unit for 8-bit numbers which are one of the operands for subsequent arithmetic operations.
- **ALU Processing**: The Arithmetic Logic Unit (ALU) performs operations using the number from the B register in conjunction with the number in the accumulator.
- **RAM to Bus Transfer**: Data output from the RAM is transferred to an 8-bit bus.



Additional register for Comparator

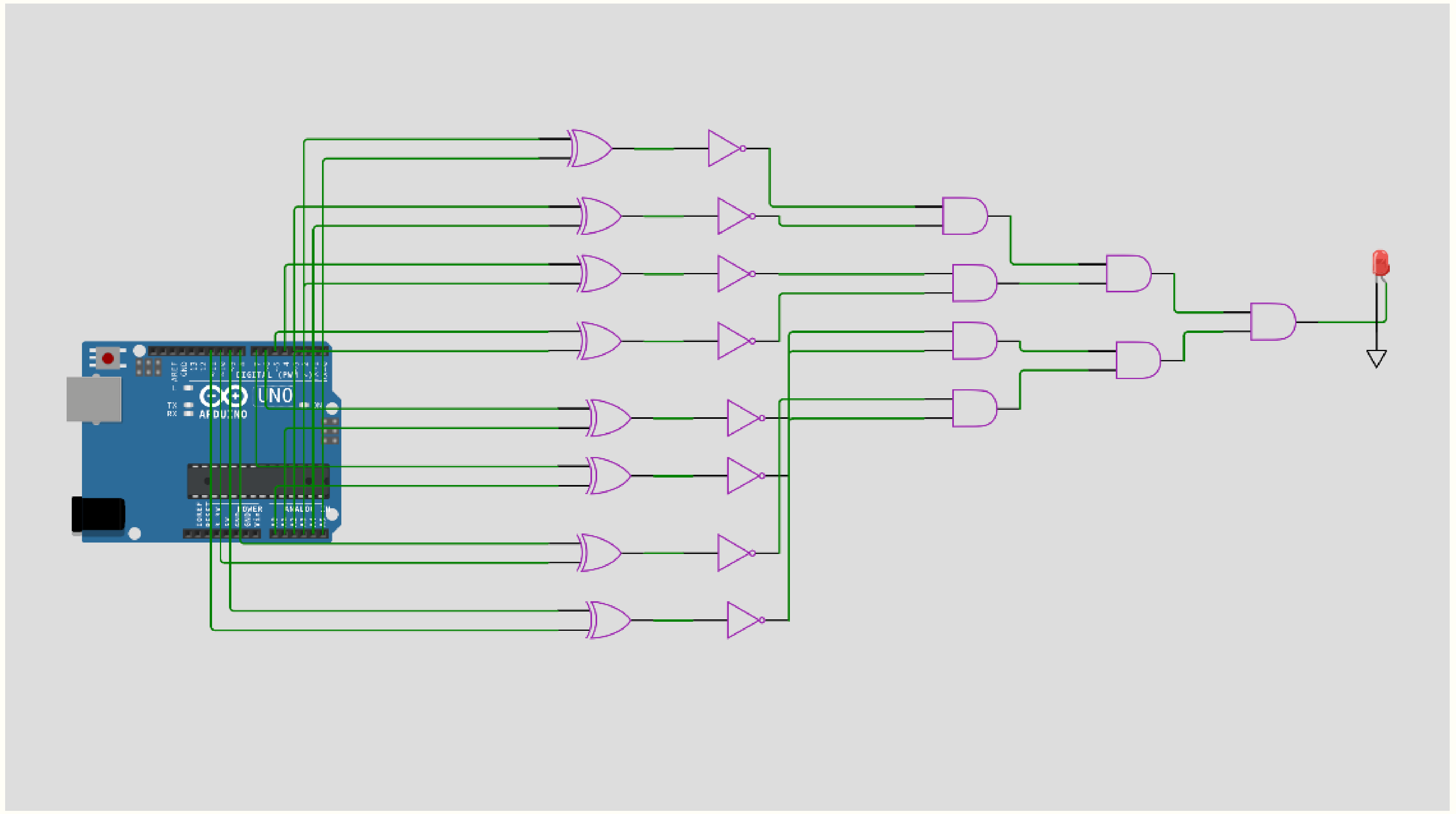
Intermediate Result Storage:
The 8-bit buffer register is primarily used for storing intermediate results during the execution of instructions.

Dual Output Configurations:
It is equipped with two types of outputs: a two-state output for direct input to the Adder for arithmetic operations, and a three-state output for transferring data to the bus.

Controlled Bus Access:
A control signal governs the conditional transfer of the register's contents to the W bus, determining the availability of data on the bus.

Comparator Interface:
Unlike the B register which feeds the ALU, this register provides its output to a comparator for comparison operations.

8-Bit Equality Comparator

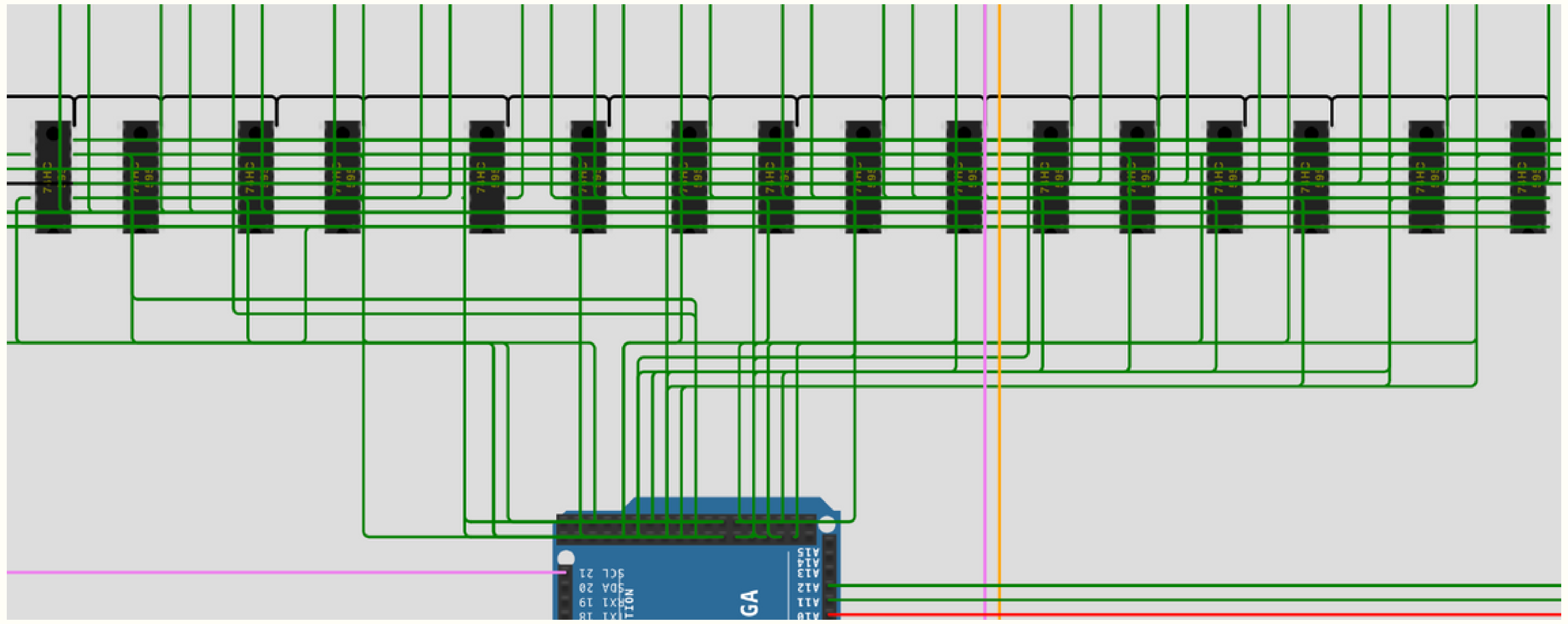


- **XNOR Gate Utilization**: XNOR gates are employed in the design to enable effective comparison, with the first comparator dedicated to the four most significant bits and the second to the four least significant bits.
- **Integration with 8-bit Bus**: The comparator system is linked to an 8-bit bus, facilitating its connection with an Arduino board for broader computational functions.

- **Comprehensive Comparison Function:** Both 4-bit comparators work in tandem to determine if a number is greater than, less than, or equal to another, covering all basic comparison operations.
- **Output Signaling:** Depending on the result of the comparison, the comparator outputs a high signal for the condition: A equal to B (A=B), enabling seamless integration into the SAP-1 computer architecture.

RAM

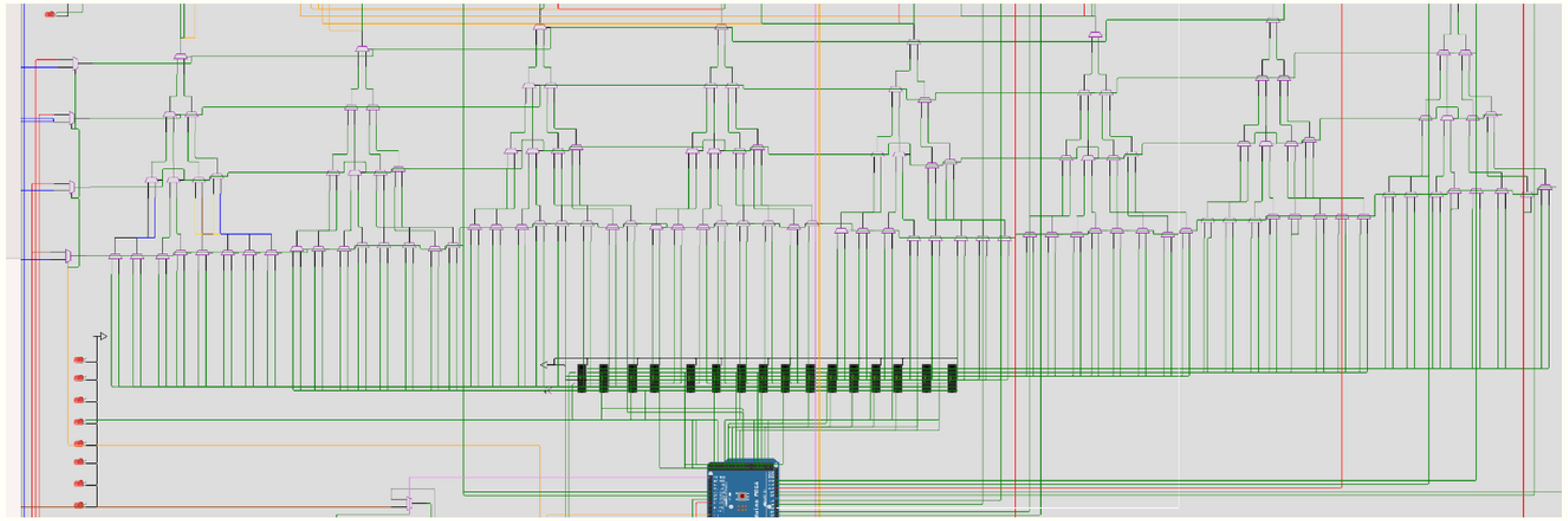
- **Bit-Shifting Mechanism:** Data is stored and then shifted into each register using the ShiftOut command, a process known as bit-shifting.
- **Output to Multiplexer:** The outputs from the shift registers (Q0 to Q7) are connected to eight 16x1 multiplexers, with each Q output from all registers connected to the corresponding input of one of the multiplexers.



- **4-bit Program Counter:** The Program Counter is 4 bits in length, enabling access to 16 distinct memory locations within the RAM.
- **RAM Structure**: Each memory location in the RAM is 8 bits wide, and the RAM itself comprises 16 individual memory slots.
- **74HC595 Shift Registers**: To construct the RAM, 16 instances of the 74HC595 shift registers are utilized, which convert serial input into parallel output from Q0 to Q7.
- **Pin Efficiency**: The 74HC595 register is designed to control eight output pins using only three input pins by sequentially shifting data into the register.

16 x 1 MUX

- **Synchronized Data Selection:** The common select line ensures synchronized data selection across all muxes, streamlining the process of choosing corresponding bits from each register, essential for coordinated instruction handling.



- **Multiplexer Array in SAP-1**: The SAP-1 system incorporates a set of eight 16-to-1 multiplexers (muxes) to manage and direct instructions from RAM to various system components.
- **Structured Connection Scheme**: Each 74HC595 register's output pins (Q0 to Q7) are systematically connected to the multiplexers. The connection scheme involves linking Q0 from each 74HC595 register to the first 16x1 mux, Q1 from each register to the second 16x1 mux, and so forth.
- **Common Select Line for Muxes**: All eight multiplexers share a common select line. This shared control allows for simultaneous selection of the same bit (Q0, Q1, Q2, Q3, Q4, Q5, Q6, or Q7) from each of the connected 74HC595 registers across all muxes.
- **Facilitated Instruction Transfer**: This arrangement enables the efficient transfer of specific instruction components from the RAM to the intended destinations within the SAP-1 system.

Instructions for Operations

- **Load:** Load the data from memory to accumulator.
- **Add:** Add the contents of accumulator and B-register and store the result back to accumulator.
- **Jump**: Jump to address specified by operand from instruction register.
- **HALT:** change enable of program counter from high to low.

- **Subtract:** Subtract the contents of accumulator and b register and store back the result to accumulator.
- **Compare:** Compare if contents of accumulator and additional register are equal and give the resulting output to controller.
- **Jump if Equal**: if the result of comparator is true jump to memory location specified.

```
 1.B00011111;//LOAD 15
 2.B00111110;//SUBTRACT 14
 3.B01101101;//CMP 13
 4.B01110101;//JPE 5
 5.B01010001;//JMP 1
 6.B00101110;//ADD 14
 7.B01101111;//CMP 13
 8.B01111001;//JPE 10
 9. B01010101;//JMP 5
10.B00000011;//HALT
11.B10111111;
12.B10111110;
13.B11011111;
14.B00000000;
15.B00000001;
16.B00001000;
```

Code of updown counter:

- Starting from decimal value 8.
- Subtracting 1 in each clock cycle.
- Comparing from memory location 13 (Zero).
- If the comparison is true, jump to memory loc 5, else jump to 1.
- Repeat until comparison is true.
- When it is true then start incrementing from Zero.
- When the value of accumulator is 0, then it will start incrementing.

Working

- **Step 1: Initialization:** Set initial values for the RAM, program counter (PC), and other registers. Initialize control signals to ensure the system is in a known state.

- **Step 2: Fetch Instruction:** Use the program counter (PC) to fetch the next instruction from RAM, load it into the instruction register (IR), and increment the PC for the next cycle.
- **Step 3: Decode Instruction:** Decode the upper 4 bits of the instruction in the IR using an Arduino as a sequencer controller to determine the operation and generate the required control signals.
- **Step 4: Execute Instruction**: Perform the operation specified by the instruction, which includes storing to or loading from the accumulator (AC), adding or subtracting contents of the B register, jumping to a new address, or comparing values and potentially causing a jump based on a comparison result.

- **Step 5: Repeat Cycle:** Return to the fetch stage to increment the PC and continue the fetch-decode-execute cycle with the next instruction.

Conclusions

Comprehensive Design and Functionality:
The SAP-1 architecture, with its well-integrated components like the Program Counter, ALU, Instruction Register, Accumulator, B-register, Comparator, multiplexers, and RAM, showcases a comprehensive design. This design effectively demonstrates the fundamental aspects of digital computing, from basic data handling to complex arithmetic and logical operations.

Instruction Handling and Execution:
The system's ability to handle a variety of instructions, including load, add, jump, compare, and conditional operations, highlights its versatility in processing and executing commands. This is further exemplified in the updown counter code, showcasing practical applications of SAP-1's capabilities.

Educational and Practical Value:
SAP-1 serves as an excellent educational tool for understanding the basics of computer architecture and digital logic design. Its simplicity, combined with the incorporation of fundamental concepts like bit-shifting and logical operations, makes it an invaluable resource for students and enthusiasts beginning their journey into computer science.

Wokwi link of project

https://wokwi.com/projects/385837427922289665

Thank you