SAP-1 Digital Logic Design Project Report Phase 4

GROUP NAME: AND-OR-NAUR

Group Members:

Lead	Abdullah Iqbal 26904
Co-Lead	Araiz 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

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.

Chapter 2

What Works

2.1 SAP-1 Components

The components that make up the SAP-1 are pivotal in its operation and are described in detail below. *SAP-1 Computer* [3]

2.1.1 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.

2.1.2 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.

2.1.3 IR

- **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.
- **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.

2.1.4 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.

2.1.5 B-register

- 1. When ADD instruction is met the controller enables the B register, 8 bit data values from the location of the operand in RAM are stored in B-register 2. Transfer from RAM to B-Register: Facilitate the transfer

of data output of address specified by the operand from RAM to B-Register when Add instruction is met which will be used by ALU to perform arithmetic and logic operations.

2.1.6 16 x 1 MUX

- **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.
- **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.

1.6.1 Multiplexer using 2:1 multiplexers [1]

2.1.7 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.

2.1.8 8-Bit Equality Comparator

- **Dual 4-bit Comparators Configuration:** The 8-bit comparator is constructed using two 4-bit comparators connected in cascade, enhancing its capability to handle 8-bit comparisons.
- **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 three different conditions: A greater than B (A¿B), A less than B (A¡B), or A equal to B (A=B), enabling seamless integration into the SAP-1 computer architecture.

8 Bit Equality Comparator [2]

2.1.9 RAM

- **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.
- **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.

2.1.10 BCD-TO-7-SEGMENT-DECODER

- **4-bit decoder:** BCD numbers are made up using just 4 right data bits from accumulator but unlike hexadecimal numbers that range in full from 0 through to F, BCD numbers only range from 0 to 9, with the binary number patterns of 1010 through to 1111 (A to F) being invalid inputs for this type of display and so are not used as shown below.

2.2 Instructions for Operations

The operation of the SAP-1 computer follows a basic instruction cycle, which includes the following steps:

Instructions (opcodes)	Description
Load (0001)	Data is loaded from memory into the accumulator.
Add (0010)	The contents of the accumulator and the B-register are added, and the result is stored back in the accumulator.
Jump (0101)	The program counter jumps to the address specified by the operand from the instruction register.
Compare (0110)	The contents of address specified by operand are loaded from RAM to additional register, the contents of RAM and additional register are compared and the result is output to the controller.
Jump if Equal (0111)	If the result of the comparison is true, the program counter jumps to the memory location specified by IR.
Subtract (0011)	The B-register's contents are subtracted from the accumulator, and the result is stored back in the accumulator.
Halt (0000)	Stop incrementing program counter.

Table 2.1: Sequential Process Steps

2.3 Example Program Implemented: Down-Up Counter

The up-down counter operates by starting at a specified value, incrementing or decrementing in each clock cycle, and making jumps based on comparisons. Below is the assembly code for the counter:

```
B00011111;//LOAD 15
B00111110;//SUBTRACT 14
B01101101;//CMP 13
B01110101;//JPE 5
B01010001;//JMP 1
B00101110;//ADD 14
B01101111;//CMP 13
B01111001;//JPE 10
B01010101;//JMP 5
B00000011;//HALT
B10111111;
B10111110;
```

```
B11011111;
B00000000;
B00000001;
B00001000;
```

Operation Description: The code initializes the counter at decimal number 8. It subtracts 1 in each cycle, compares the result with decimal number zero, and jumps to a new location based on the comparison. This loop continues until the comparison is true, after which it starts incrementing from zero and compares with decimal number 8. if comparision is true program halts else continuous to inrement. BCD-TO-7 degment decoder is used to display right 4 bits of accumulator to 7 segment display

2.4 Working of SAP-1

2.4.1 Initialization

1. Set up the initial values of the RAM, program counter (PC), and other registers.
2. Initialize the control signals and ensure the system is in a known state.

2.4.2 Program Counter (PC) Operations

1. The output of the PC is the input to the select lines of RAM.
2. Use the PC to access the memory location in RAM pointed to by the PC.
3. As select lines of the 16x1 mux are the same, Q0 to Q7 values of a 74HC595 register are loaded to the IR.
4. Increment the PC to prepare for the next instruction.

2.4.3 Instruction Register (IR) and Control Signals

1. The upper 4 bits of the IR are decoded to determine the operation to be performed.
2. Control signals are generated based on the decoded instruction to coordinate the subsequent steps of the cycle.

2.4.4 Decoding of Instructions

The decoding process involves several steps where the contents of the accumulator and various registers interact with memory locations specified in the instructions:

1. The content of the accumulator (AC) is stored in the specified memory location.
2. The specified memory content is loaded into the B register, and the binary adder/subtracter processes it with the content of the accumulator.

3. The specified memory address is loaded into the program counter (PC), resulting in a jump to that address.

4. The specified memory address is loaded into an additional register for comparison with the accumulator's content, with the result being output to the Arduino controller.

5. If the comparison is true, the program counter (PC) is updated with the specified address, causing a conditional jump.

2.4.5 Repeat the Cycle

1. Return to the fetch stage, increment the PC, and fetch the next instruction from memory.

Chapter 3

Conclusion

The SAP-1 architecture provides a comprehensive design and functionality, integrating key components like the Program Counter, ALU, Instruction Register, and others, to demonstrate core digital computing operations. Its capacity for handling and executing a diverse set of instructions underscores its versatility. Serving as an educational tool, SAP-1 offers practical value in teaching the fundamentals of computer architecture and digital logic, making it a foundational resource for learners in the field.

Bibliography

[1] *1.6.1 Multiplexer using 2:1 multiplexers*. https://www.researchgate.net/figure/161-Multiplexer-using-21-multiplexers_fig18_327425781.

[2] *8 Bit Equality Comparator*. URL: https://easyeda.com/module/8_Bit_Equality_Comparator-fe5bc189884141869ec343ee07bc12fc.

[3] *SAP-1 Computer*. URL: https://karenok.github.io/SAP-1-Computer/.