A green and yellow logo

Description automatically generatedA close-up of a logo

Description automatically generated

**SAP-3 Processor Core**

**Design and Implementation**

Abderahman Mohamed Khalil V23010331

Ahmed Kamal V23010450

Mohaned Tarek V23010449

Mohamed Gamal V23010636

**Delivered to Dr. Islam Yehia**

Contents

[**Introduction** 3](#_Toc154060559)

[**Design plan and team plan** 3](#_Toc154060560)

[**Blocks details** 4](#_Toc154060561)

[**Memory** 5](#_Toc154060562)

[**ALU** 5](#_Toc154060563)

[**Register File** 6](#_Toc154060564)

[**Clock Controller** 6](#_Toc154060565)

[**Output Controller** 7](#_Toc154060566)

[**Controller and instruction register** 7](#_Toc154060567)

[**Control unit and instruction decoding** 7](#_Toc154060568)

[**Results** 10](#_Toc154060569)

[Programming Examples 10](#_Toc154060570)

[Addition code 10](#_Toc154060571)

[Stack Testing Code 11](#_Toc154060572)

[**Further development** 13](#_Toc154060573)

# **Introduction**

The project discusses the design of simple as possible (SAP) 3 processor. The instruction set of the processor is similar to 8088-8086 but in a smaller version. Additional hardware is added to the main processor core to communicate with external peripherals. The specification sheet shows the processor specifications, details, instructions with timing diagram, examples of codes to be run by the processor and finally further development that intended to be made for the processor.

# **Design plan and team plan**

The plan is to divide the design into sub-blocks that can be designed independently and then integrate them and decide how each block will be tested. Before each design, the team must meet and agree on the inputs and outputs of each block and how the rhythm will flow.

Here is the team plan:

* Abdelrahman Khalil: the instruction register and the controller, clock, writing the testing assembly code.
* Mohaned: the ALU
* Ahmed Kamal: Register file, memory
* Mohamed Gamal: output controller, clock controller

# **Blocks details**

The design of the processor follows Von Neumann architecture. As a result, all sub blocks of the processor communicate through a single 16-bit bus transferring both address and data as shown in figure 2. The controlling lines form 35-bit word size controlled by the processor controller.

A diagram of a computer

Description automatically generated

Figure 1 Processor Core Overall Block Diagram

### **Memory**

**Function Brief:**

The ram memory of the processor is 64Kb size with 16-bit addressing size. The address data that comes from the bus is buffered into the memory address register (MAR). controller controls this block using write enable and external multiplexer to control the flow of the data into the MAR.

### **ALU**

**Function Brief:**

ALU which is the brain of the processor, can perform several arithmetic and logical operations including:

* 8-bit addition
* Addition with carry
* 8-bit subtraction
* Subtraction with borrow
* AND, OR, Complement, XOR
* Rotate right and left
* Rotate right and left with carry
* Quick increment and decrement by 1

Also provide control to set or reset the carry. The ALU has three main registers which are:

* Accumulator
  + The main register of the ALU and stores the output of the ALU and its state is represented by the flags register
  + All single operand operations like increment and decrement operates on it
  + Can be modified by the ALU or the bus
* Temporary register
  + The second operand of the ALU
  + Can be modified from the bus only
  + Gives flexibility to do more operations (Add other registers to the Accumulator)
* Temporary register 2
  + This register is modified by the controller only and the end user has no control over it.
  + Stores only the value of the Accumulator during some instructions that operate with the value of the Accumulator without affecting the value of the Accumulator.

### **Register File**

**Function Brief:**

The register file includes all the registers mentioned in the programming model. The register file has an internal increment and decrement circuit to increment and decrement frequently used registers such as PC or SP or operate on other registers with simple control steps and without affecting the flags.

### **Clock Controller**

**Function Brief:**

This block is used for clock gating after the program finishes. In further development this block won’t cut the clock after the program ends because there will be an operating system but will cut it during idle state.

### **Output Controller**

**Function Brief:**

This block enables the processor to communicate with the other peripherals and with the outside world.

### **Controller and instruction register**

**Function Brief:**

The Instruction register stored the instruction to be executed. The controller decodes the instruction stored in the instruction register and controls all the other blocks of the processor through the control lines (called controlled word).

# **Control unit and instruction decoding**

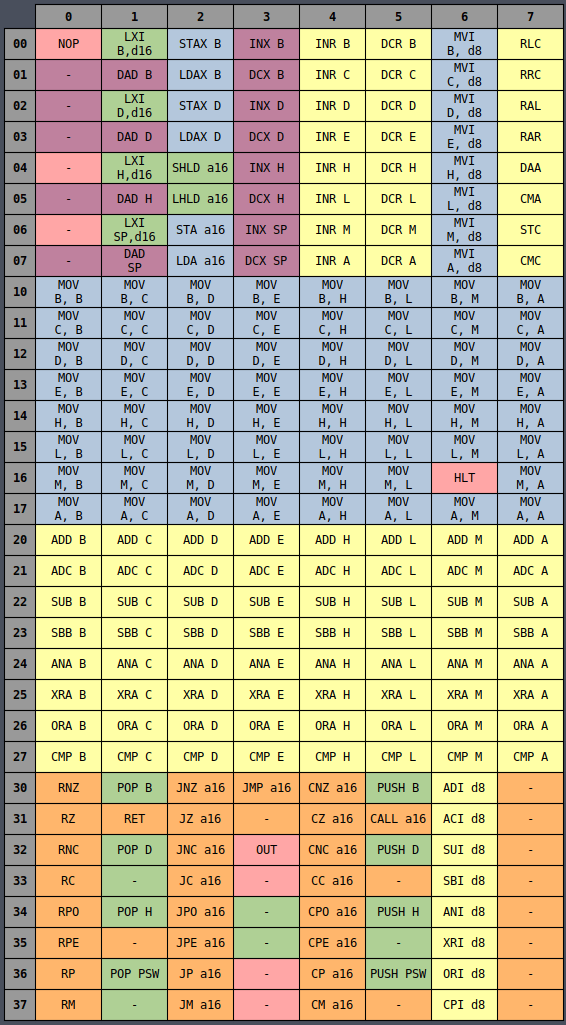
The control unit took most of the effort. The use of ”casez” was very powerful since it provided the ability to group multiple of similar instruction in the same case.

We found the idea on the internet and this image which shows a summary for 8085 instructions in compact way.

A table with different colored squares

Description automatically generated

We also found that octal representation is much easier in Verilog so we followed the next table



The reference could be found here:

<https://pastraiser.com/cpu/i8085/i8085_opcodes.htm>

# **Results**

We tested each block individually to make sure that there is no error with the hardware. All blocks then tested to work properly with each other via writing simple testing assembly programs. The results obtained from the simulation and implemented on the FPGA.

## Programming Examples

To make sure that all the processor works in harmony, 2 simple programs (Addition program and stack testing program) were written to test some of the rest of the instructions.

### Addition code

**MVI** A, 08

**MVI** B, 04

**ADD** B

**OUT** 02

**HLT**

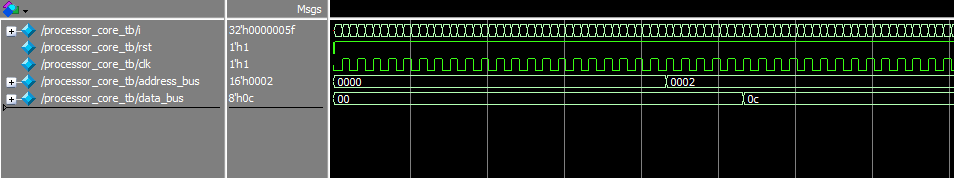


Figure 2 addition program results

A computer with wires connected to a piece of paper

Description automatically generated

### Stack Testing Code

**INX** B

**INX** B

**INX** B

**PUSH** B

**POP** D

**MOV** A, E

**OUT** 20

A screen shot of a computer screen

Description automatically generated

A close-up of a circuit board

Description automatically generated

A screenshot of a computer

Description automatically generated

# **Further development**

Further development could be made as follows:

* Implement dual core processor to investigate this kind of architecture
* Implement approximate instructions targeting high speed and low power