# Architecting MIPS Processor

Students: Joel Kahn, Max Eskenazi, and Omer Agam;
Supervisors: Nishil Talati, *M.Sc.* and Prof. Shahar Kvatinsky
SciTech 2017

## Objective

- In this project, we designed a CPU based on a MIPS processor architecture shown in Fig. 2
- **MIPS** stands for **M**icroprocessor without **I**nterlocked **P**ipelined **S**tages
- For this, we learnt Boolean algebra, digital circuits, modern computer architectures, and Verilog Hardware Description Language (HDL)
- We designed MIPS processor using Verilog HDL and tested it by running a Fibonacci number generator on the designed processor

## Background–Boolean Algebra

**Logic operators**

- Boolean algebra is a field of mathematics concerning the algebra of Boolean (truth/false) variables represented as one and zero. For each variable n, there are $\mathbf{2^{2n}}$ functions possible. For example, with 2 variables (x,y) there are 16 possible Boolean functions.
- Possible functions include NULL, AND, Inhibition, XOR, OR, NOR, Equivalence, Implication, Complement, NAND (Fig. 1), *etc.*
- To reduce the complexity of such functions, methods such as Karnaugh Maps [1], QM method [2], etc. are used to simplify the function

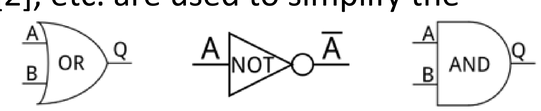


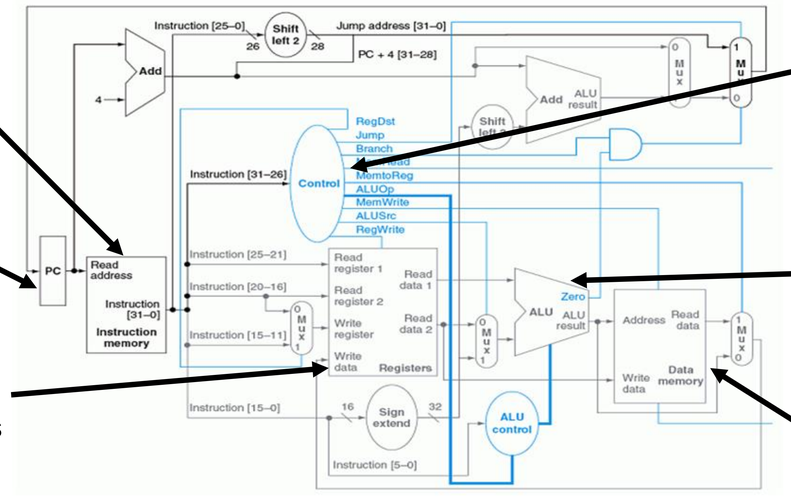

**Fig. 1. Schematic of basic gates used to implement Boolean functions - (from left to right) OR gate, NOT gate, and AND gate.**

## MIPS Processor Architecture

**Instruction Memory**
Stores the instructions, i.e., what operation to perform in each cycle

**Program Counter**
Stores the memory address of instruction

**Register File**
Uses multiple registers (fast memory elements) to store either data used in instructions or address of memory locations




**Main Control**
Based on the instruction, it generates the control signals to monitor the activity of different units

**ALU – Arithmetic and Logic Unit**
Performs the arithmetic and logic operations on the data, for example, addition, substation, AND, OR, XOR, *etc.*

**Data Memory**
Stores data to be processed in different instructions

**Fig. 2. Architecture of a single-cycle MIPS processor without pipeline.**

## Fibonacci Number Generator

We wrote a assembly program, converted it to 1s and 0s, and loaded them into the instruction memory to execute it on the modeled MIPS processor using Verilog. The program outputs the first 16 Fibonacci numbers into the registers as shown in Fig. 3.

**Fibonacci Series**

$S_n = S_{n-1} + S_{n-2}$
$S_0 = 0$
$S_1 = 1$

```
Assembly program:
lw $0,  $1,  #00  (loading  1)
add $0, $1, $2 (adding 1+0)
add $1, $2, $3 (adding 1+1)
. . .
```

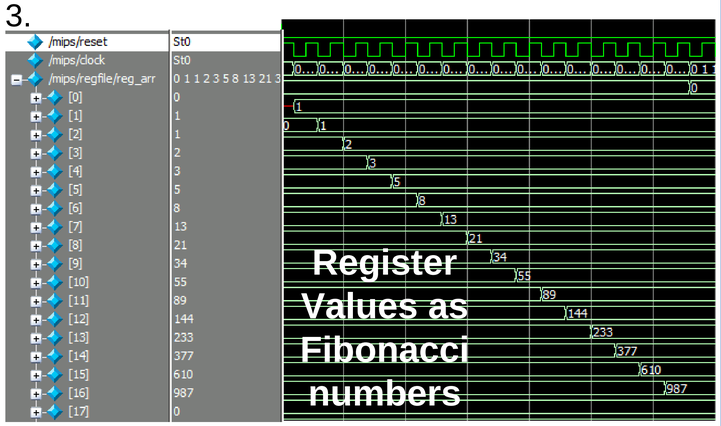



**Fig. 3. Assembly program loaded to the instruction memory (left) and the register file showing Fibonacci numbers (right).**

## Acknowledgements

We would like to thank ***Nishil Talati, M.Sc.*** and ***Prof. Shahar Kvatinsky, PhD*** for hosting and guiding us through our research in his laboratory.
We would also like to thank the foundations and donors for their generous support of the SciTech Program.

## References

[1] Karnaugh, M. "The Map Method for Synthesis of Combinational Logic Circuits," Trans. of the American Institute of Electrical Engineers. 72 (9): 593–599, 1953.
[2] Quine, W. "The Problem of Simplifying Truth Functions," The American Mathematical Monthly. 59 (8): 521–531, 1952.