Computer Architecture Probe

Yi Zhao, Yuxia Yao | Computer Science Serior Project | Supervised by Professor Olivier Marin

NYU SHANGHAI 上海纽约大学

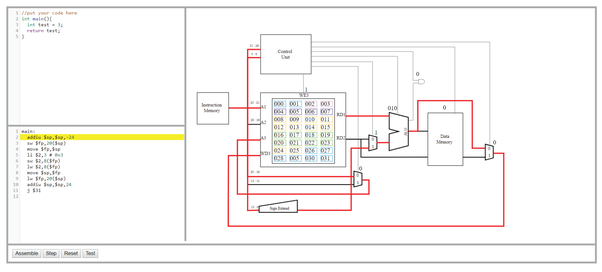
Background

In the course of Computer Architecture, we studied how a modern computer works through multiple layers of interface, connecting both hardware and software. In order to study the layers that usually work "behind the scene" of real computers, students turn to graphical simulators to observe the interaction between components. Most simulators focus on one particular layer. While such tools help grasp the theoretical concepts of particular layers, they fail in helping students master translation between two layers of interface, not to mention the relations of the internal layers themselves. Hence, there is a need for a graphical simulator that illustrates the operation of all layers during program execution, starting from a high-level programming language to its most elementary components (transistors, resistors, capacitors).

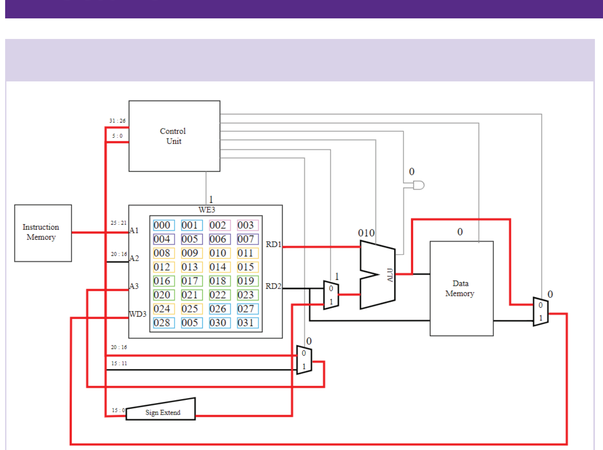
Design

The project Computer Architecture Probe (CAP) is a graphical simulator of all layers of the computer architecture based on MIPS. Given a user-input code in C language, CAP translates it into Assembly Instruction Codes (AIC) using Compiler Explorer API. Then it simulates and renders the output in microarchitecture, gate, intermediate and transistor layer. The web-based application is light-weight and thus avoids problems like installation and distribution. The simulator is programmed in Object-Oriented method.

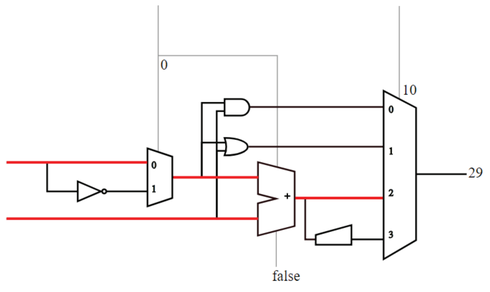
The User Interface design consists of three panels: a code editor on the top left, a panel for the assembly language on the bottom left, and the simulator on the right. Below the panels is a bar of progress controller with four buttons, Assemble, Step Forward, Step Back and Reset. After assembling their code, users then could press Step Forward button to observe step-by-step simulation in every layer. In the simulation panel, only one layer shows at one time but users can scroll mouse to switch between layers. The assembly code in execution is marked yellow, and the activated datapaths in the simulator is highlighted with its output value.



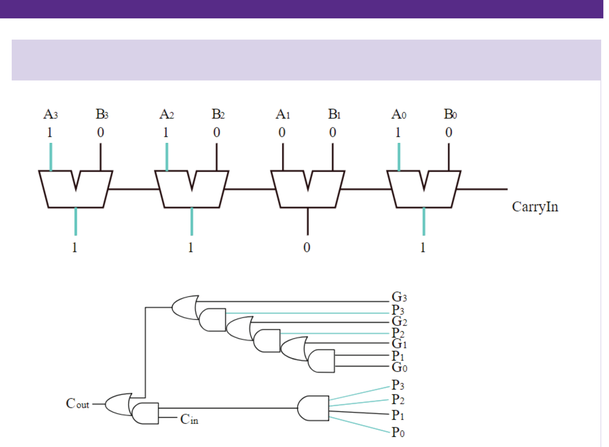
Details



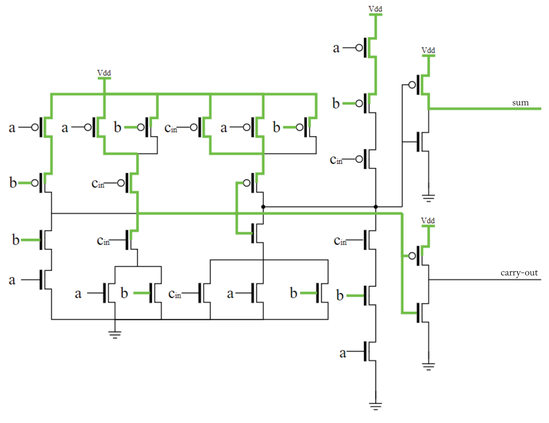
The 1st layer is an overview of all microarchitecture components in a single-cycle MIPS processor. The input is a 32-bit instruction code. Datapaths are highlighted only if their data count toward the calculation. In this example, it is executing `addiu $sp, $sp, -24`.



The 2nd layer shows ALU component in gate level. It takes inputs A and B and generates the result based on a 3-digit ALU Control Code. Datapaths are highlighted only if their data count toward the calculation.

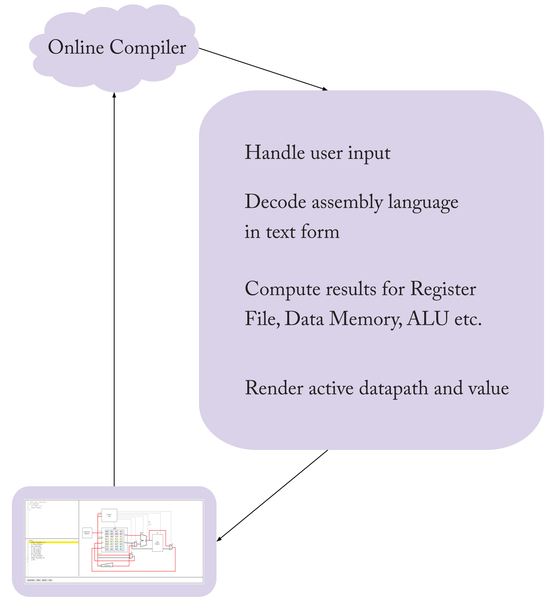


The 3rd layer gives a diagram of a 4-bit additon block in gate level. It is the very last four bits taken from a 32-bit carry lookahead adder that computes carry-out indepent of the result to achieve a faster calculation. The path is highlighted in cyan when its value is 1.



The 4th level reveals a conventional design of a full adder in transistor level. A full adder takes three binary input *a*, *b* and a carry-in and calculates the sum and carry-out. The green lines represent current flow. Depending on the input, individual transistors will be on or off. In this specific example, this full adder sums a=0 and b=1, with a carry-in=0.

Structure



Reference

The existing computer architecture simulators listed below have demonstrable achievements in education, though they hardly cover the simulation over all the layers. Building on their advantage, the Computer Architecture Probe is designed to be graphical and web-based for the sake of completeness, user-friendliness and accessibility.

Simulator	Level	Program does	Visual way
Little Man Computer	microarchitecture with memory addressing (princeton architecture)	fetch-execute cycle	color + ball movements
Visual6502	transistor	transistor highlights	color
Marie.js	microarchitecture	datapath + register values	color + text
SIMAS	microarchitecture	machine code -> cpu, memory, io	text
The very simple CPU simulator	gate	within register and ALU	color
EDCOMP	microarchitecture modules circuits inside a module	register, memory, IO, pipelines	color+text
HASE	depend on the architecture file	datapath, register values	text
Easy CPU	microarchitecture	datapath, register, memory(data+stack)	color + animations