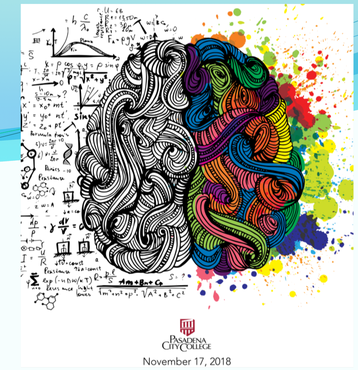
Pep9Micro:
A Microcoded CPU Implementation

Matthew McRaven, Mentor: J. Stanley Warford

Division of Natural Science, Pepperdine University, Malibu, CA 90263



Abstract:

Pep9Micro is a newly created tool to teach programming in Microcode. Pep/9 is a 16-bit CISC computer with eight addressing modes used to teach assembly language programming and computer systems concepts. Students program with the applications Pep9 and Pep9CPU, which simulate the Pep/9 computer at two different levels of abstraction. Pep9 allows students to program in assembly language and execute their programs on the virtual machine simulator. At a lower level of abstraction, Pep9CPU allows students to program microcode fragments that implement individual assembly language instructions. This poster describes the development of a third virtual machine, Pep9Micro, that extends the capabilities of Pep9 and Pep9CPU by designing a microcode implementation of the complete Pep/9 instruction set. It provides the assembler from Pep9 and the CPU simulator of Pep9CPU so that complete assembly language programs can be executed at the microcode level spanning two levels of abstraction. Pep9Micro extends the Pep9CPU microcode language with conditional microcode branch instructions and the design of a microcode store. Pep9, Pep9CPU, and Pep9Micro are support software for the text Computer Systems *by J. Stanley Warford, and are available on GitHub as open source projects.*

Background

Microcoding is a design approach for Complex Instruction Set Chips (CISC) that adds an abstraction layer between assembly level instructions and the hardware implementation of the instruction set. Microcode is heavily utilized in the implementation of x86 processor family (though many other processors have microcode). Microcode manages the flow of data through the CPU necessary to implement each instruction in the machines Instruction Set Architecture (ISA). Due to its tight coupling with hardware design, almost all microcode is closed a source implementation. Pep9Micro uses available open source CPU research with the Pep9 ISA to provide an interactive introduction to CISC CPU design.

Pep/9 is a 16-bit CISC computer with 58 instructions and 8 addressing modes. Pep/9 addressing modes mirror the C memory model; so, programs from C may easily be translated to assembly.Figure 1 is a screen shot of the Pep9 application. At a lower level of abstraction, the data section (i.e., registers, main memory access, and the ALU) of the Pep/9 CPU is described in the book *Computer Systems*. Figure 2 is a circuit diagram on which the Pep9CPU application is based.

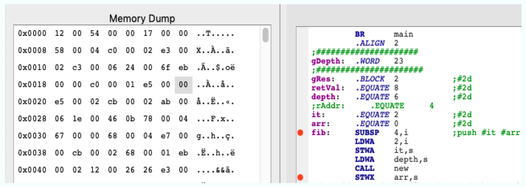


Fig. 1 - Memory Dump and Assembly Code Program for Pep9.

Without Pep9Micro

Pep9CPU is an integrated development environment (IDE) that allows students to create microcode fragments to perform various computations according to the Pep/9 specification. However, the application is unable to fully implement an arbitrary assembly program since it lacks an implemented control section and conditional microcode jumps. Therefore, a microcode programmer is always limited to writing code fragments of arithmetical and logical operations without any ability to loop.

From the view of an assembly code programmer using the Pep9 application, one could write an arbitrary application and execute it. The simulator is a complete model of Pep/9 at the assembly level, but due to the lack of the Pep/9 CPU control section, is unable to model the inner workings of the CPU. Therefore, the programmer, while able to execute arbitrary code fragments, is unable to visualize the operation of the Pep/9 hardware.

The lack of a Pep/9 control section means a programmer cannot write assembly code programs or trace its execution in hardware at the CPU level.

Program CPU Microcode with Pep9Micro

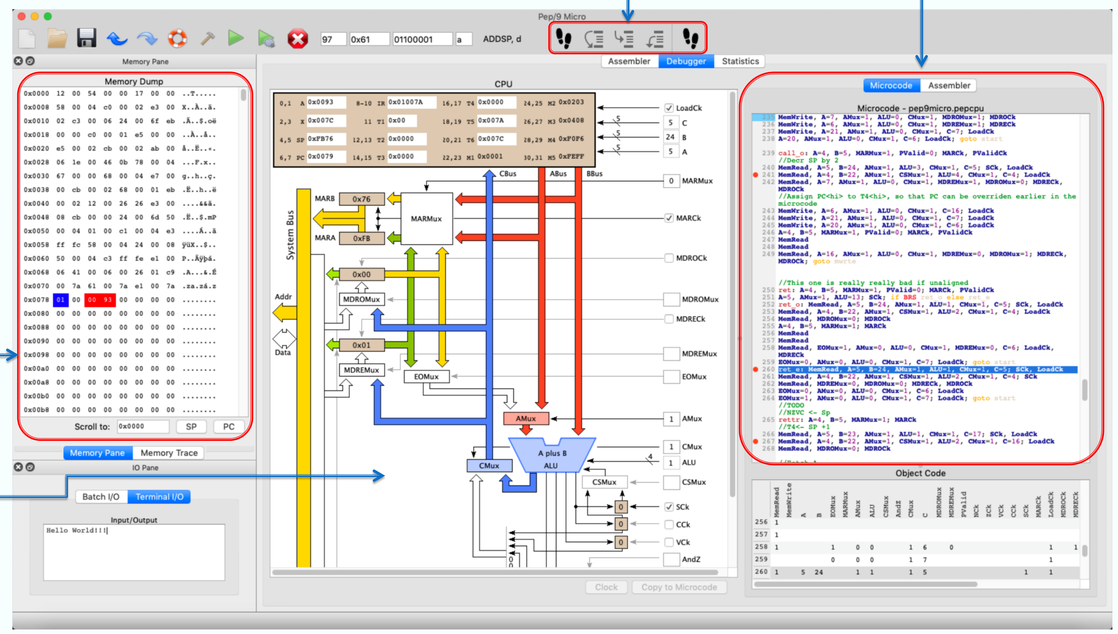
With Pep9Micro, students can now program and debug across both microcode and assembly language abstraction levels. Students can step through code in any of these levels. New features include setting break points, and use standard debugging comments such as step over, step into, and step out.

Debug Actions: Supports common debugging features like step-into, step-out, and step-over.

File Editor: Break points work for both microcode and assembly. Students can switch between assembly and microcode at will.

Editable Memory Dump: View and edit memory on-the-fly. Highlights current program counter and any changed values.

Interactive CPU: Watch change in data flow while executing microcode.



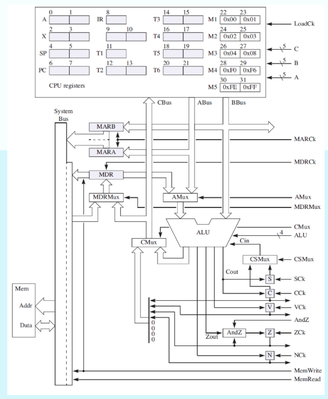


Fig. 2 – Data section of Pep/9 CPU

With Pep9Micro:

The Pep/9 CPU was extended by adding a control section and conditional branching shown in Figure 3. By designing a fully functionally Pep/9 control section, programmers may simultaneously work at the at the microcode and assembly levels. Each conditional branch enumerates both the target when the condition is true and when it is false.

The design avoids microjump and microreturn instructions, and instead uses lookup tables to handle the translation of an assembly instruction or addressing mode to the first line of microcode needed to implement it. When these tables are combined with a decoder for a branch function, it becomes trivial to select the next microinstruction.

Pep9Micro is an application that is a working implementation of the assembly code simulator, residing atop data section powered by the newly designed control section and augmented microcode. The Pep9Micro IDE allows programmers to evaluate programs written in assembly code and watch the programs step along at both the microcode and assembly levels.

Fig. 3 — The newly designed CPU control section.

Acknowledgements:

Support for this project was provided by the Natural Science Division at Pepperdine University and the Academic Year Undergraduate Research Initiative (AYURI) Fall 2018.

References:

[1] Computer Systems, J. Stanley Warford, Jones and Bartlett Learning, 5th edition, 2017.
[2] Structured Computer Organization, Andrew Tanenbaum, Pearson 5th edition, 2005.
[3] Superscalar Microprocessor Design, Mike Johnson, Prentice Hall, 1990.

Conclusions:

Pep9Micro provides students with tools to program and troubleshoot CPU microcode, while using IDE tools that are similar to popular IDE environments. Since most legacy architectures are closed source, this tool provides students the ability to see the magic inside the CPU.

Pep9Micro is also a stepping stone to the future. With the advent of open source processors, microcode can become open sourced just like other software projects. This trend is already starting with open platforms, such as RISC-V.