Pep/9 for iPad:
A 2016-17 Academic Year Seaver Research Council Grant Application

Principal Investigator: Stan Warford, Professor of Computer Science

Seaver Undergraduate Student: Josh Haug, Senior Computer Science/Mathematics major

Research Focus: An iPad app for the Pep/9 Assembler/Simulator

Background

Computer science students in the second-year course COSC 330, Computer Systems, use a desktop app named Pep/9 to learn assembly language and machine language programming. The current version of the application was developed by Prof. Warford and a former undergraduate student six years ago. It is an implementation of a virtual machine and is used in conjunction with the textbook, *Computer Systems*. [1]

Research goal

The specific research goal is to begin production of a software app that we will name "Pep9Pad", which stands for "Pep/9 for iPad". To duplicate all the functionality of the current desktop version of the app is not feasible for a single undergraduate student in a single year. We anticipate this research effort will result in a partially functional app that will be completed by future undergraduate students.

The Current Pep/9 Desktop Application

The current application is a free open-source package used by students at Pepperdine and other universities around the world. It is available at the textbook web site [2] and is described in the paper by Warford and Dimpfl. [3]. The application is available for MS Windows, Mac OS X, and Unix/Linux systems.

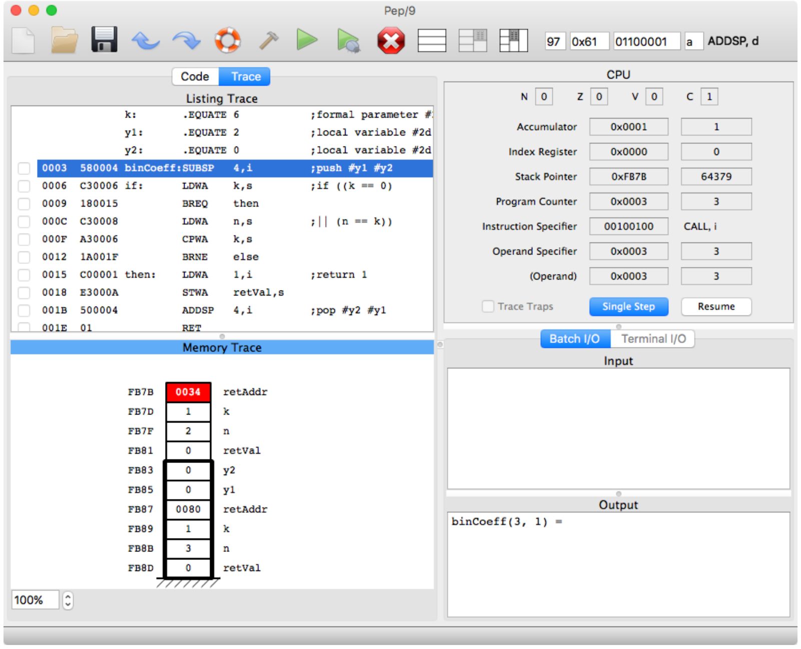
The two major components of the app are the assembler and the simulator. The assembler features the following:

- An integrated text editor
- Error messages that are inserted within the source code at the place where the error is detected
- Student-friendly machine language object code in hexadecimal format
- The ability to code directly in machine language, bypassing the assembler
- The ability to redefine the mnemonics for the unimplemented op-codes that trigger synchronous traps

The simulator features the following:

- Simulated ROM that is not altered by store instructions
- A small operating system burned into simulated ROM that includes a loader and a trap handler system
- An integrated debugger that allows for break points, single-step execution, CPU tracing, and memory tracing
- The option to trace an application, the loader, or the operating system in any combination
- The ability to recover from endless loops
- The ability to modify the operating system by designing new trap handlers for the unimplemented opcodes
- Every example from the text built into the application, making it a useful tool for class demonstrations

The following screenshot shows the debugger feature of the simulator. The upper left pane shows the trace of the program source written by the student. The lower left pane shows the debugger graphical output. The upper right pane shows the content of the Pep/9 central processing unit. The lower right panes show the program input and output.



97 0x61 01100001 a ADDSP, d

```
             k:        .EQUATE 6          ;formal parameter #
             y1:       .EQUATE 2          ;local variable #2d
             y2:       .EQUATE 0          ;local variable #2d
0003  580004 binCoeff:SUBSP    4,i        ;push #y1 #y2
0006  C30006 if:       LDWA    k,s        ;if ((k == 0)
0009  180015           BREQ    then
000C  C30008           LDWA    n,s        ;|| (n == k))
000F  A30006           CPWA    k,s
0012  1A001F           BRNE    else
0015  C00001 then:     LDWA    1,i        ;return 1
0018  E3000A           STWA    retVal,s
001B  500004           ADDSP   4,i        ;pop #y2 #y1
001E  01               RET
```

```
FB7B  0034  retAddr
FB7D  1     k
FB7F  2     n
FB81  0     retVal
FB83  0     y2
FB85  0     y1
FB87  0080  retAddr
FB89  1     k
FB8B  3     n
FB8D  0     retVal
```

N 0 Z 0 V 0 C 1

Accumulator	0x0001	1
Index Register	0x0000	0
Stack Pointer	0xFB7B	64379
Program Counter	0x0003	3
Instruction Specifier	00100100	CALL, i
Operand Specifier	0x0003	3
(Operand)	0x0003	3

```
binCoeff(3, 1) =
```

Project Objective

The complete project is too ambitious to be completed by a single undergraduate student in a single year. The objective for this research is to implement the top-level user interface for all parts of the app, implement the editor, and implement the assembler. If time permits, the next objective would be to complete the simulator and the memory pane (not shown in the above figure). If time permits after that, the next objective would be to complete the debugger graphical output.

Research expenses

Because this research involves open-source software and is easily carried out on personal computers, there is no need for additional funding other than the credit to the student's tuition.

References

[1] Warford, J. Stanley, *Computer Systems*, Jones and Bartlett Publishers, LLC, Fourth edition, 2010.
[2] `http://ComputerSystemsBook.com`
[3] Warford, J. Stanley, and Dimpfl, Christian, The Pep/8 Memory Tracer: Visualizing Activation Records on the Run-Time Stack, *Forty-First ACM Technical Symposium on Computer Science Education*, vol. 42, no. 1, March, 2010.