

A PDP-11 COMPUTER SIMULATOR

Bryan Higgs, Ph.D.*
Associate Professor of Computer Science, Rivier College

Keywords: *PDP-11 computer, simulator, machine-level programming*

Abstract

Teaching an early computer organization and architecture course can be a challenge. It is difficult to enliven many of the basic computer science concepts and have students understand the relevance of such concepts to the real world. Among the topics that can be made more interesting and relevant to students are programming at the machine level, and the use of assembly language. A hands-on approach increases enthusiasm and understanding. In this article, we present a simulator for a PDP-11 computer that uses a GUI-based interface and a built-in assembler, and we argue that its use in computer organization courses can enhance student understanding for machine-level programming.

1 Motivation

Over several sessions of teaching an introductory Computer Organization course to Computer Science undergraduates, we have found that many of our students struggle with a number of the basic concepts presented in the course. In an attempt to give students a more hands-on experience, we chose a textbook [1] that provided a very user-friendly computer simulator, MarieSim [2], which is written in Java and makes excellent use of a graphical user interface (GUI) to provide a pleasant and relatively comfortable experience for students.

Unfortunately, we found the design of the (imaginary) computer being simulated ("MARIE: Machine Architecture that is Really Intuitive and Easy") to be too limited for students to program much beyond relatively trivial machine-level programs. While the machine was designed -- for pedagogical reasons -- to be extremely simple, we found its limited instruction set and its non-orthogonal provision of features such as indirect addressing (provided only on two of its 13 instructions) to be frustratingly restrictive. For example, we wanted to expose our students to the concept of stacks, and their application to expression evaluation, but we found this to be very difficult to implement using this architecture.

2 The Vision

We felt that we could apply the concepts of a user-friendly Java GUI to a more realistic, but still understandable, machine simulation, and thereby allow our students to progress further in their use and understanding of machine language programming.

It occurred to us that simulating a PDP-11 would be advantageous. The PDP-11 [3, 4, 5] was a highly influential series of 16-bit minicomputers sold by Digital Equipment Corporation (DEC) during the 1970s and 1980s.

Features of the PDP-11 architecture that we felt would lend themselves to ease of learning include:

- A general-purpose register (GPR) machine, which is the most common kind of CPU in use today.
- An orthogonal instruction set; once the concepts of one instruction are understood, other instructions are easy to learn.
- A rich set of addressing modes (Register, Register Deferred, Autoincrement, Autoincrement Deferred, Autodecrement, Autodecrement Deferred, Index, Index Deferred, PC Immediate, PC Absolute, PC Relative, PC Relative Deferred), applied in a very consistent manner over virtually all the available instructions. Again, once the concept of a particular addressing mode is understood, that knowledge can be applied across the instruction set. These addressing modes allow for very easy LIFO stack manipulation, and can be used to great effect for studying stacks and their applications.
- An instruction set using an expanding op-code approach, which could be used as a real-world example of that technique.

In general, we could imagine the course introducing features gradually, without overwhelming our students with too much information at once. For example, while the PDP-11 boasts a relatively powerful instruction set, we could gradually introduce a subset of these instructions, adding more instructions as our students become more comfortable. A similar approach can be taken with addressing modes.

The PDP-11 was a highly successful real-world architecture that was used for many kinds of applications, from banking applications to capturing data in high-energy physics experiments, and everything in between. We know that it was capable of being applied to many areas of computing, and expect that our students could take advantage of this in their programming activities.

The PDP-11 architecture was used as an extended example in an early edition of a popular textbook on Computer Organization [6].

3 The Implementation

3.1 Build, Acquire, or Adapt?

We looked into the availability of PDP-11 simulators, and found quite a few. [7-12]. However, none seemed to satisfy our needs. We were looking for a pedagogically oriented solution involving the use of a modern GUI, preferably written in Java for portability; none lent itself to this approach.

So, we decided to build our own PDP-11 simulator.

3.2 The Implementation in Java

We implemented a PDP-11 simulator in Java, which included:

- An emulation of the PDP-11 machine, including:
 - Almost a complete set of machine instructions
 - CPU fetch-decode-execute instruction cycle
 - 64Kbytes of Memory, addressable by byte or 16-bit word.
 - 8 General-Purpose Registers

- Processor Status Word flag bits
- A simple assembler, which produced 'executable' files for the emulator
- A GUI, which allows the user to view the above, to observe a loaded program in symbolic form, and to execute that program to observe its execution.

3.2.1 The Graphical User Interface

The style of presentation was influenced by the MARIE Simulator [2], and so owes much to its philosophy.

An example of how the PDP-11 simulator appears is shown in Figure 1.

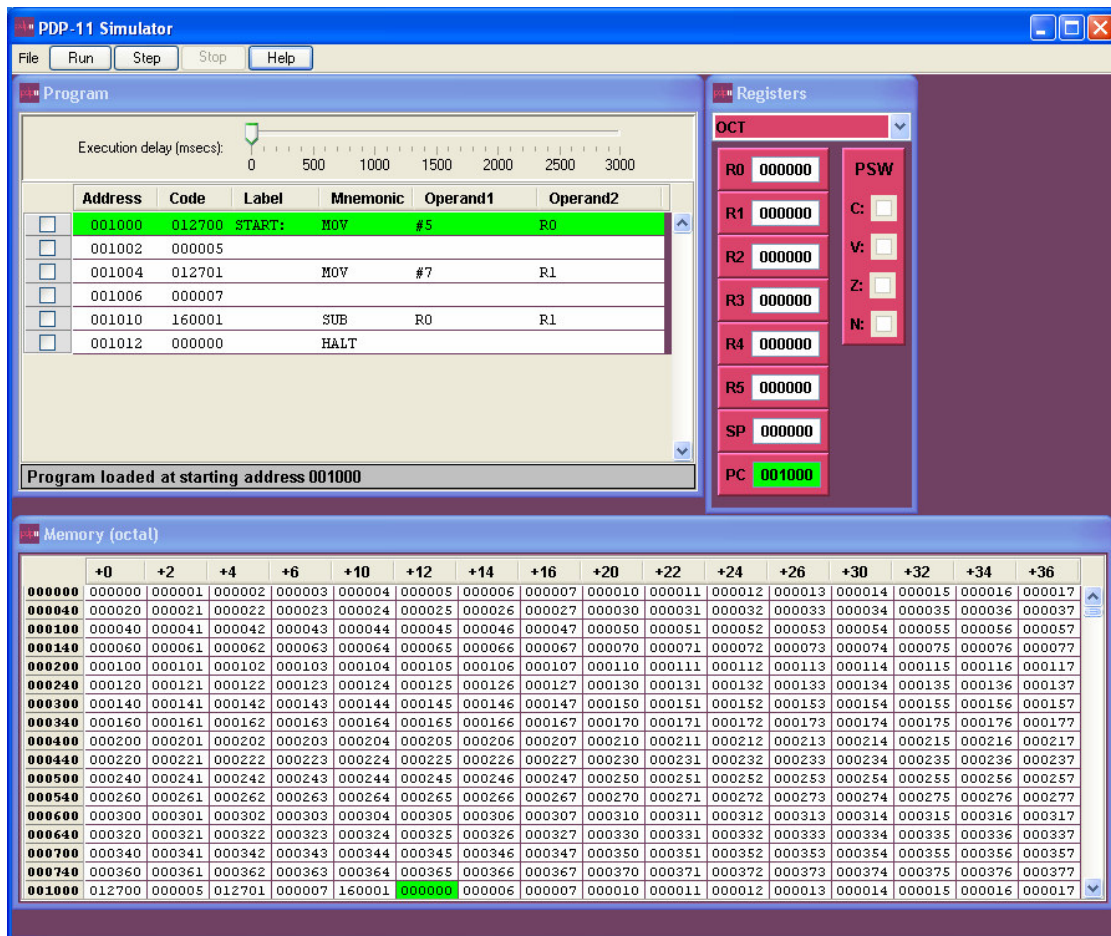


Figure 1: The PDP-11 Simulator Environment

3.2.2 The Program Panel

The panel labeled 'Program' displays a symbolic version of a loaded program. For each instruction, it displays the octal address, the instruction code in octal, the instruction mnemonic, and operands (if any). Note that instructions that use Immediate addressing mode consume additional words of memory; this explains those lines which contain no instruction mnemonic or operands. During program execution, the next instruction to be executed is highlighted in green.

The top of the panel shows a scrollbar that may be used to slow program execution, to allow convenient observation of program flow.

3.2.3 The Registers Panel

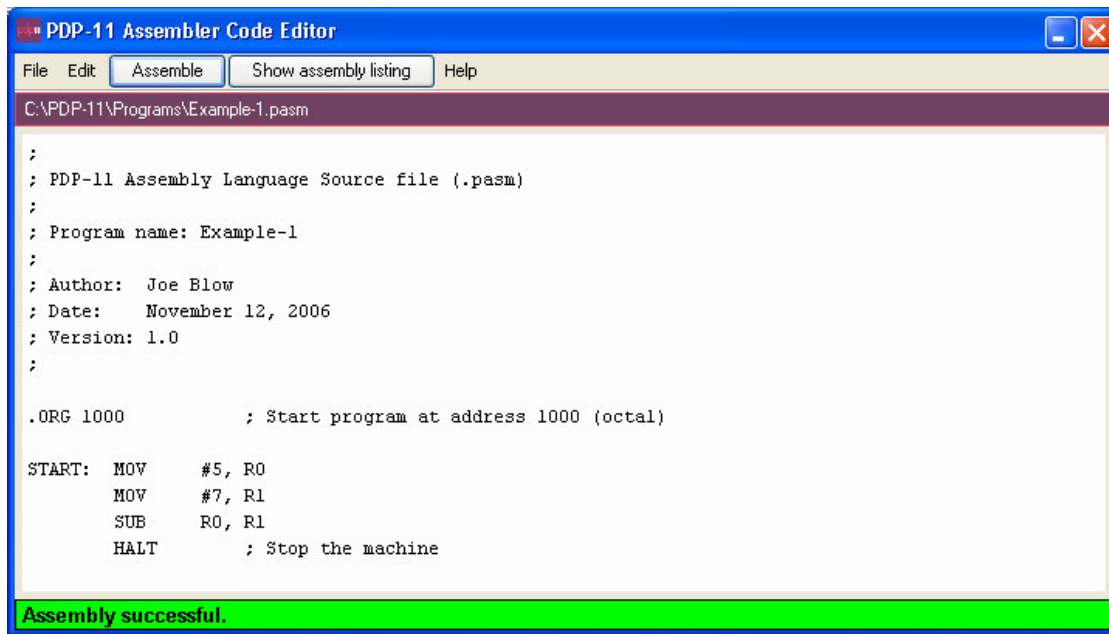
The panel labeled 'Registers' displays the contents of the machine's general-purpose registers, R0 – R7, in octal, decimal, or hexadecimal. The panel also displays the state of the flag bits C, V, N and Z from the machine's Processor Status Word. These flag bits indicate the result of each instruction, on which conditional branch instructions depend. When an executed instruction causes changes to the contents of any of these registers, and/or the flag bits, those changes are highlighted in green.

3.2.4 The Memory Panel

The panel labeled 'Memory (octal)' displays the contents of the machine's memory words, with addresses and contents displayed in octal. When an executed instruction causes changes to the contents of memory, those changes are highlighted in green.

3.2.5 The Assembler Code Editor Panel

When a user requests to create or change a source program, the simulator displays an assembler editor code panel. An example of how this appears is shown in Figure 2.



```

PDP-11 Assembler Code Editor
File Edit Assemble Show assembly listing Help
C:\PDP-11\Programs\Example-1.pasm
;
; PDP-11 Assembly Language Source file (.pasm)
;
; Program name: Example-1
;
; Author: Joe Blow
; Date: November 12, 2006
; Version: 1.0
;
.ORG 1000 ; Start program at address 1000 (octal)

START: MOV #5, R0
      MOV #7, R1
      SUB R0, R1
      HALT ; Stop the machine

Assembly successful.

```

Figure 2: The Assembler Editor Panel

The user may invoke the assembler directly from the editor panel. A successful assembly is indicated by a message highlighted in green at the bottom of the panel. If the assembly fails, a message is displayed, and an assembler listing panel is automatically shown. The user may also ask to see the assembler listing. An example of an assembler listing is shown in Figure 3.

```

PDP-11 Assembler Listing
Close Print C:\PDP-11\Programs\Example-1.lst
Assembly listing for Example-1.lst
Fri Nov 17 21:01:37 EST 2006

Line | Addr.  Code Source
-----|-----
1 |          ;
2 |          ; PDP-11 Assembly Language Source file (.pasm)
3 |          ;
4 |          ; Program name: Example-1
5 |          ;
6 |          ; Author:  Joe Blow
7 |          ; Date:   November 12, 2006
8 |          ; Version: 1.0
9 |          ;
10 |         ;
11 |         .ORG 1000          ; Start program at address 1000 (octal)
12 |         ;
13 | 001000 012700 START: MOV   #5, R0
    | 001002 000005
14 | 001004 012701          MOV   #7, R1
    | 001006 000007
15 | 001010 160001          SUB   R0, R1
16 | 001012 000000          HALT          ; Stop the machine

Symbol Table:
-----|-----
Symbol | Value (octal/decimal) | Line References
-----|-----
START  | 001000/512            | 13
-----|-----

Assembly successful

```

Figure 3: The Assembler Listing Panel

4 Project Status

The PDP-11 Simulator implementation is nearing completion. What remains is:

- Completing the implementation of breakpoints.
- Thorough testing of the entire simulator.
- Relatively minor cleanup.

We plan to use this simulator in the next session of our Computer Organization course, and to measure how easily our students learn in this potentially more sophisticated simulation environment. We also plan to produce more challenging assembly language programming assignments, and investigate how easily our students are able to expand their assembly language programming skills.

Additional possibilities exist for this simulator, including studying its source code in courses in object-orientation, advanced Java, GUIs, compilers, and data structures.

5 Conclusions

We have almost completed implementation of a PDP-11 Simulator written in Java. The simulator provides a user-friendly GUI interface to assembly language programming in a real-world environment.

The simulator holds much potential promise in our Computer Organization course, and potentially elsewhere.

References

- [1] Null, L. and Lobur, J. The Essentials of Computer Organization and Architecture, Second Edition, Sudbury, MA: Jones and Bartlett, 2006
- [2] Null, L. and Lobur, J. MarieSim: The MARIE Computer Simulator, Journal of Educational Resources in Computing, 3, (2), June 2003.
- [3] PDP-11, <http://en.wikipedia.org/wiki/Pdp-11>, retrieved November 12, 2006.
- [4] PDP-11, <http://www.pdp11.org/>, retrieved November 12, 2006.
- [5] Bell, G. and Strecker, W., "Computer Structures: What have we learned from the PDP-11?", November 1975, [http://research.microsoft.com/users/GBell/Digital/Bell Strecker What we%20 learned fm PDP-11c%207511.pdf](http://research.microsoft.com/users/GBell/Digital/Bell%20Strecker%20What%20we%20learned%20from%20PDP-11c%207511.pdf), retrieved November 12, 2006.
- [6] Tanenbaum, A., Structured Computer Organization, First Edition, Englewood Cliffs, NJ: Prentice-Hall, 1976.
- [7] The Computer History Simulation Project, Bob Supnik, <http://simh.trailing-edge.com/>, retrieved November 12, 2006.
- [8] DEC PDP-11 Simulators, <http://www.xsim.com/bib/index4.d/Tool-DEC-11-43.html>, retrieved November 12, 2006.
- [9] The PDP-11 Simulator System, Hankinson, D. W., and Bidulock, D.S., <http://pages.cpsc.ucalgary.ca/~dsb/PDP11/manual/>, retrieved November 12, 2006.
- [10] The PDP-11 Processor Handbook, <http://pages.cpsc.ucalgary.ca/~dsb/PDP11/>, retrieved November 12, 2006.
- [11] Wolffe, G. S., Yurcik, W., Osborne, H., and Holliday, M. A., Teaching computer organization/architecture with limited resources using simulators, ACM SIGCSE Bulletin, Proceedings of the 33rd SIGCSE technical symposium on Computer Science education, SIGCSE '02, 34, (1), 176-180, 2002.
- [12] Computer Architecture Simulators, <http://sosresearch.org/caale/caalesimulators.html>, retrieved November 12, 2006.

* **Dr. BRYAN HIGGS** is an Associate Professor of Computer Science at Rivier College. He earned a B.Sc. (Honors) in Physics from the University of Liverpool, United Kingdom, and subsequently M.Phil. and Ph.D. degrees in Physics from Yale University. He then changed fields, and joined Digital Equipment Corporation where he worked for 20 years in a variety of positions, culminating in the Database Engineering group. In 1994, Digital sold its Database business to Oracle Corporation, and Dr. Higgs transferred to Oracle, together with the rest of the Database group. He worked for Oracle in their Software Engineering group for 7 years. In 2002, Dr. Higgs decided to return to academia, and joined the full-time Computer Science faculty at Rivier College, where he has worked ever since. His major interests are Database Systems, programming languages (especially Java and C/C++), Computer Security, and Web development. Recently, he has become interested in scripting languages, including Perl, Python, JavaScript, PHP, and Ruby. Dr. Higgs is a member of the Association for Computing Machinery. He is Chair of the Faculty Development Committee of the Faculty Senate.