Evaluating and improving the Educational CPU Visual Simulator: a sustainable Open Pedagogy approach

Renato Cortinovis
Freelance researcher
Bergamo, Italy
rmcortinovis@gmail.com

Ranjidha Rajan
Department of Computer Science
MSU Denver
rranjidh@msudenver.edu

Abstract

This paper describes the user-evaluation of a recently significantly redesigned old but effective educational CPU visual simulator, and its subsequent further improvement in a second main iteration of an Open Pedagogy / OER-enabled Pedagogy project. The goal of the simulator is to support novices in understanding the key components of a CPU by means of detailed animations of the execution of its instructions, in understanding the mapping from high-level control structures to low-level (assembly and machine) code, and in coding meaningful programs with a simple but representative assembly language. The simulator developed in the previous iteration, published with an open licence, was evaluated in three different educational settings: technical high school and adult-education specialised computer science courses in Italy, and a university in the U.S.A. This evaluation, mainly based on the thematic content analysis of the feedback from about 50 students, has been very positive, and provided constructive feedback for further improvements and extensions. Grounded on the results of the previous evaluation, the simulator is being re-developed as a brand-new Web application, further extended with features such as the addition of synchronised audio description of what is happening while the simulator animates the inner working of the CPU, the introduction of support for array processing, and spin-off projects such as a supporting open e-book, and a natural language conversational agent to answer students' queries.

The above extensions are being carried out by students as an OER-enabled pedagogy project, integrated in their more conventional educational activities. This strategy aims to reduce the deplorable waste of resources associated with "disposable" traditional assignments, at the same time challenging students to address real-world professional problems. The open material resulting from this latest iteration will be adopted, further evaluated, and hopefully enriched once again next year on a wider scale, demonstrating the feasibility of a self-sustainable process where students fully engage in iteratively improving and extending open resources, developing their professionality while benefiting the commons.

1. Introduction

Cortinovis (2021) describes in detail the recent redesign and extension of an old but effective educational CPU visual simulator (Decker and Hirshfield, 1998), which was quite popular in USA universities, but which used technologies that became obsolete. The development was carried out by students attending computer science continuing-education evening-classes of a technical high school in Bergamo, Italy (ITIS P. Paleocapa Serale). The extensions were substantial, including, among many others, the addition of CPU flags and related conditional jump instructions to better illustrate the control flow in low-level languages; the possibility to define and use labels for numerical addresses, in order to clarify the concept of a variable as well as the mapping of high-level programming constructs to assembly language; enhanced colour-coded animations to better understand the sequential nature of the control unit and the role of the control bus in addition to the address and data buses. Figure 1 shows a screenshot of the redesigned Educational CPU Visual Simulator, with a sample program computing the sum of all numbers from 1 to MAX (which has value 5 in this case), loaded in RAM.

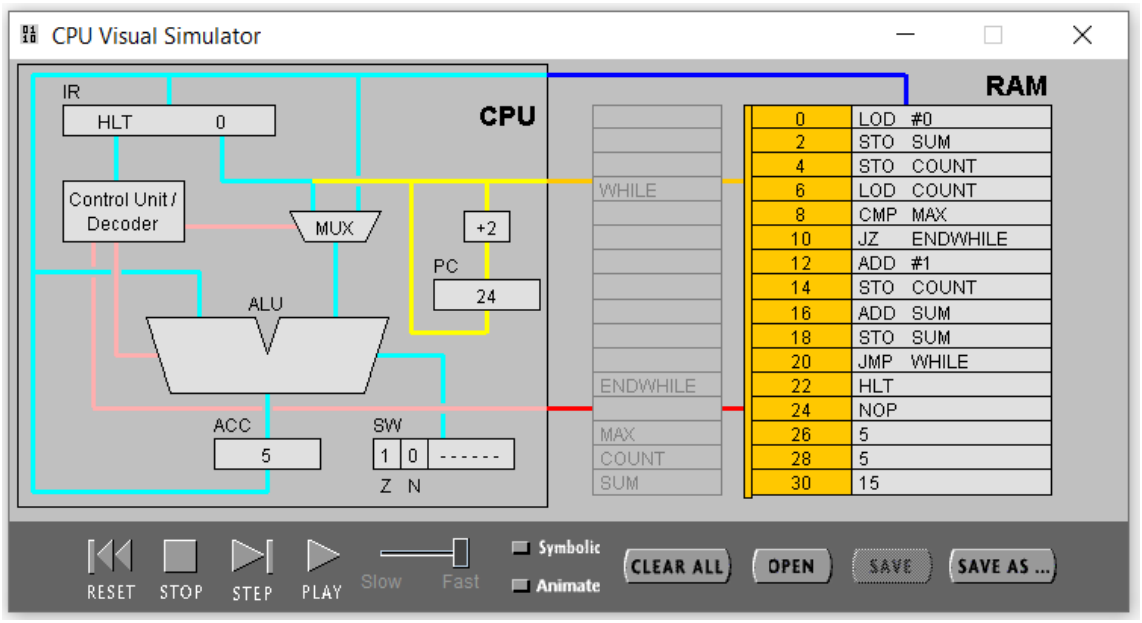

Figure 1 – A screenshot of the new CPU Visual Simulator

The literature reports that often students, despite studying both programming with high-level languages as well as the basics of computer architecture, do not fully grasp how code written in high-level languages is actually executed on the hardware of a computer (Evangelidis et al., 2001; Miura et al., 2004). Hence, the main goal of the simulator is to support novices in understanding the key components of a CPU by means of detailed animations of the execution of its instructions, in understanding the mapping from high-level control structures to low-level (assembly and machine) code, and in coding meaningful high-level programs with a simple but representative assembly language. As an example, Figure 2 reports the high-level pseudocode from where the assembler code in Figure 1 was derived.

```
// sum all the integer numbers from 0 to MAX
// assertion: MAX >= 0

SUM = 0
COUNT = 0
WHILE COUNT != MAX DO
      COUNT = COUNT +1
      SUM = SUM + COUNT
ENDWHILE
```

Figure 2 – Pseudocode corresponding to the assembler code in Figure 1

The redesigned simulator was developed following an Open Pedagogy / OER-enabled pedagogy approach (Wiley and Hilton, 2018), where students modified the old simulator, carrying out all the necessary design, refactoring, and coding, as part of their educational activities in an otherwise traditional programming course. The result of their efforts was made openly available for use and for further improvements by other students and teachers. This approach challenged students to address a real-world professional problem, an experience which was deeply appreciated. One of them, for example, reported:

*"I was impressed to experience how solving real problems such as introducing a new feature in a broader complex application, had such a positive impact on my motivation and my **passion** for software development".*

This strategy also aimed to reduce the deplorable waste of resources associated with "disposable" (write and forget) traditional assignments, while attempting to provide a contribution to the sustainable development goals (Lane, 2017). All the students involved in this open pedagogy project, indeed, were thrilled by the opportunity to personally contribute to the common good. One of them, for example, commented:

"Working for a purpose and contributing to a wider goal is definitely more rewarding than getting a good score on a typical classroom assignment for its own sake"

Besides motivating and benefiting the students who worked on enhancing it, the simulator is also benefiting the students who use it to learn computer architecture concepts; in this paper, indeed, we describe the international evaluation of the redesigned simulator from the perspective of these last students (Section 2). We also describe the resulting identification of further requirements and opportunities for extensions of the simulator (Section 3), and its development as a brand-new Web application (Section 4). The resulting product will be published again with an open licence, hence enacting a sustainable OER-enabled pedagogy process.

2. Evaluation of the simulator

The open simulator described above, was evaluated in three different educational settings: technical high school and continuing-education computer science courses in Italy, and Metropolitan State University Denver (MSU Denver) in the U.S.A.

2.1 Experimentation in a technical high school and in continuing-education evening courses

The simulator was first used by one of the authors in a continuing-education introductory class of 23 students, for a total of 30 hours. First, many students experienced difficulties installing the simulator, which proved to be a real hassle. Indeed, we discovered, different operating systems, Java runtimes, and computer settings, required minor but annoying ad-hoc interventions. Second, there was a fairly comprehensive help already embedded in the simulator, but many students expressed the desire for additional supporting material. This request led to the fast prototyping of a supporting e-book with comprehensive explanations of the hardware structure, the instruction set, patterns for the translation of high-level programming constructs, exercises with step-by-step solutions, and additional proposed exercises. The e-book was quickly prototyped by the students of an advanced class of the same school, and extensively used by both the students and the instructor in the previous foundational class. This e-book was developed in a cloud-based Content Management System in order to allow the developers to quickly make the necessary modifications, reacting in real time to the emerging needs.

The visual simulator plus its associated e-book were then used in another technical high school class with about 20 students. The feedback from the experienced teacher was definitely positive, and he contributed a few additional exercises that were further elaborated and integrated into the existing e-book. Yet the teacher observed that the simulator could be even more useful if it supported functionalities to handle arrays too.

2.2 Experimentation in CS organisation courses at university level

The simulator and related e-book were then experimentally adopted at MSU Denver by one of the authors, in a "2000 level architecture course" class of 30 students, and subsequently in a second "1000 level architecture" class with 20 students.

2.2.1 First MSU Class: "2000 level architecture course"

Despite the simulator being intended as a resource for an introductory course, it could only be used, in the first university class, after the students had already completed their ARM (Arm Limited, 2021) programming class, because the decision to use the material was taken after the policies and the syllabi for the university courses were already established. The material was introduced by the lecturer in a two-hour session, and was distributed to the students via the internal LMS. Students were required to carry out the exercises proposed in the documentation with the simulator, and to complete, within three weeks, a short survey to collect their feedback. The following questions were included:

- What was most interesting?

- What should be improved so you can get a better learning experience?
- What is your experience when learning a new instruction-set with the basic knowledge of ARM instructions and its datapath?

The 30 students of this first class provided more than 2000 words of meaningful feedback, that was analysed following a thematic content analysis approach (Cho and Lee, 2014; Stemler, 2001). Most higher-level codes ("themes") were identified top-down from the questions, while the lower-level codes were identified inductively bottom-up. The most relevant codes were:

- Positive aspects
    - Detailed animations
    - Flow-control instructions
    - Simulator execution control
    - Switching between symbolic and binary
    - Embedded help
- Suggested improvements
    - Additional explanations
        - Real-time explanation of animations
        - Explanation of CPU sub-components
    - Usability
        - Installation
        - More detailed user control on the animations
        - Full screen
        - Use of colours
- Acquisition of previously missed concepts

Here a few excerpts are reported, to illustrate some of the codes.

Positive aspects / Detailed animations

The detailed animations of the simulator were, by far, the most appreciated feature. For example:

> *The most interesting part to me was visually being able to indicate what was happening when each instruction was executed. [Student 15]*

> *I found it really helpful to see what was happening as the program moved and did the steps in the program. [Student 3]*

> *The most interesting part of the program was the CPU's circuit and how we can see how it's getting looped through. [Student 11]*

Suggested improvements / Additional explanations / Real-time explanation of animations

Various students expressed the wish to have additional real-time explanations of the animations. For example:

> *I think written details of what is occurring, an explanation of the animation would make a better learning experience. [Student 4]*

> *It would be cool to have the option to have a guided tutorial/dynamic text instruction of what is happening while each instruction executes. [Student 17]*

Acquisition of previously missed concepts

As previously discussed, the simulator was intended as introductory material, while students of this first MSU class were exposed to an ARM processor before the visual simulator. This unplanned situation proved very useful to demonstrate that the simulator helped the students to grasp aspects that they missed by working first in the ARM environment. This is suggested, for example, by the following excerpts:

*The most interesting part was watching how the PC directs the CU to the register [memory location] where the command is found and then storing the instructions in the IR. I always imagined the process as one; **instead**, the PC points to the register [memory location], the CU takes the command and stores it in the IR, decodes the instructions, and then carries out the instructions. [Student 5]*

*The flow from the program counter and the control unit, the way things were executed was **different** than I had really expected. [Student 14]*

*The most interesting part about the data path simulation to me was to see the instructions getting passed around. I was **surprised** to see that the control unit is pretty much always in use whether it is grabbing instructions or passing them along. [Student 19]*

This was likely induced by the visualization and animation of the CPU behaviour, which helped students to enrich their understanding from a partially confused abstract level to a more firm and concrete grasping.

2.2.2 Second MSU Class: "1000 level architecture course"

The second university class of 20 students fit precisely the target population for which the simulator was intended. Following the same process as in the previous class, with a change in the questions to align with the foundational course, the following questions were used:

- Which examples did you try for understanding the simulator (submit the screenshot of those)?
- Reflect on your understanding of how high-level code runs on hardware using Von Neumann architecture.
- What improvements can be made in the simulator for a better understanding?

More than 1000 words of feedback were collected and analysed. Despite the different background of the students, the feedback was consistent among the two classes, even if with a slightly different emphasis, and the codes obtained from the analysis of the previous class could be used in this second case too. For example:

Positive aspects / Detailed animations

The visualizer helps a lot in showing how assembly operates. [Student 17]

Suggested improvements / Additional explanations / Explanation of CPU sub-components

I think there are different parts that I wish I could hover my mouse over to get a detailed description of what it actually is. [Student 5]

Acquisition of previously missed concepts

Unlike the students of the previous class, the students of this class had not completed a previous course on computer architectures. Yet, they did have some prior knowledge coming from previous lessons, and the feedback from the following student shows again that the simulator was instrumental in integrating previously acquired knowledge:

I think the Von Neumann architecture model provides a great visual for putting together everything we have learned so far. [Student 2]

Finally, the following feedback provides some indication that one of the main objectives of the simulator was achieved:

I previously didn't know how higher-level languages got translated into something the computer can understand and seeing this is quite fascinating. [Student 17]

The simulator helps us to understand that high level code runs on Von Neumann architecture by first being run through a compiler, which converts it into assembly, which is then assembled into machine code. [Student 3]

3. New requirements identified

The analysis of the feedback as previously discussed, allowed us to identify requirements for the improvement and extension of the visual simulator in a new OER-enabled pedagogy iteration. The main requirements could be summarised as follows:

- Re-develop the simulator as a new Web application;
- Provide support for array processing;
- Provide synchronised audio (and textual) description of the animations in the simulation (with a multilanguage architecture);
- Improve user control over the animation speed and level of detail; in particular, provide the possibility to pause the animation and resume it without restarting the instruction;
- Port the existing supporting e-book to HTML/JavaScript and enrich it with additional interactive exercises;
- Provide appropriate mechanisms (possibly including a natural language conversational agent) to answer students' queries.

4. Development of the new enhanced version

Following the Open Pedagogy / OER-enabled Pedagogy approach, and according to the requirements identified, students of the continuing-education evening courses are now developing a new version of the educational material. That is, they are producing improved/extended open educational material, based on the evaluation of previously existing open educational material.

First, a brand-new Web-based version of the simulator is being finalized, to let users enjoy the material directly online through a familiar browser, avoiding the hassle of installing an application and the Java runtime. The new simulator provides improved functionalities, notably a synchronised audio or textual description of what is happening, such as: "The CPU is now going to fetch an instruction: the content of the PC is placed on the address bus... a read operation is signalled on the control bus... the memory retrieves the content stored at the address indicated and transmits it on the data bus...". The audio description has been selected because the visual channel of the user is already fully engaged in following the animations of the simulator. Yet the possibility to use a textual description is made available to avoid excessive noise when multiple students are using the simulator in places like a lab without headsets.

To address the issue raised by the teachers of the technical high school class concerning the desirable support for arrays, the possibility of self-modifying code (code that alters its own instructions at runtime) has been introduced. This solution has been selected, instead of alternatives such as adding a new register with indirect addressing, for the following reasons: (1) it does not increase the complexity of the visualization by requiring another implicit or explicit register; (2) it does not increase the complexity of the existing instruction set with a new addressing mode and possible register addressing instructions; (3) it naturally exploits the Von Neumann architecture of the simulator; (4) it opens the door to address interesting and contemporary issues such as security aspects and genetic programming.

Other features include the possibility to pause and resume the visual simulation without restarting the whole instruction, and a better control of the execution speed.

The associated e-book is being further extended, based on the requests for additional documentation and further examples. The current version is being ported from a proprietary CMS to pure HTML/JavaScript technology, further enriched with additional interactive exercises, and improved providing additional support that proved to be frequently needed by students and instructors. As an example, users got frequently stuck trying to use a label in an instruction as placeholder of a physical address, when the label was not previously defined.

4.1 Tentative spin-off projects

Finally, we are also working on the development of an experimental natural language conversational agent to answer students' queries. This might well sound over-ambitious: it would require a considerable effort in itself, while not even being essential to the project. Yet, this is just one of a number of tentative spin-off projects, aiming at providing opportunities to experiment with technologies some students are curious about.

The overall project provides context and meaning: a grand vision, a broad shared goal, a sense of community both at local and global levels. In such a context, every achievement, even if modest, will be a success, because it will provide anyway some useful contribution to the wider goal. At the very least, thanks to the "open" philosophy, it will advance the starting point for other students who will forge ahead an extra step. For example, the students working on this spin-off project have already collected a number of questions and prepared suitable answers (by actually providing support to other students with lower competences): at the bare minimum, these questions could be easily integrated as useful FAQ in the documentation of the simulator. Yet, the vision is to avoid setting upper limits on what students can accomplish, but rather to carefully shape and suggest opportunities that fit the students' natural interests, boost enthusiasm, sustain and occasionally steer efforts, and finally just make the most of whatever achievement.

5. Conclusions and future activities

This paper has described the qualitative user-evaluation – in the USA and in Italy – of a recently significantly redesigned old but effective educational CPU visual simulator, as well as the resulting plans for its development as a brand-new Web application. The development of the new simulator and associated complementary material, is being carried out by students attending continuing-education evening-classes of a technical high school in Italy. These developments are organized as OER-enabled pedagogy projects integrated in their conventional educational activities, exploiting open material that was previously developed with the same approach, in the same school, mostly by students of previous generations. This strategy aims to reduce the resources wasted in "disposable" traditional assignments, challenging students to address real-world professional problems, to produce open material useful to a wider community. The resulting material will be adopted the next academic year on a wider scale – in particular, it will be integrated in the "1000 level architecture" courses at MSU Denver to explain how high-level languages end up being executed in the CPU – which is one of the main objectives of the simulator. This experience will permit a further evaluation of the simulator and associated documentation, to be hopefully further enriched in a new future OER-enabled pedagogy iteration. These activities suggest that the OER-enabled pedagogy, where students engage in iteratively improving and extending open resources, developing their professionality while benefiting the commons, is inherently self-sustainable.

6. Acknowledgements

We would like to thank the students who engaged in these OER-enabled pedagogy activities, in particular Jonathan Cancelli for his generous contribution to the development of the simulator, as well as the students who keenly provided their constructive feedback. We would also like to thank Cengage for granting the permission to modify, reuse, and republish the obsolete original applet (non-commercial, educational purposes only).

7. References

Arm Limited. (2021) *Armv7-M. Architecture Reference Manual* (E.e ed.).

Cho, J. Y. and Lee, E. H. (2014) Reducing confusion about grounded theory and qualitative content analysis: Similarities and differences. *The Qualitative Report*, 19(32), pp. 1-20.

Cortinovis, R. (2021) *An Educational CPU Visual Simulator*. Proceedings of the 32nd Annual Workshop of the Psychology of Programming Interest Group (PPIG), York, United Kingdom.

Decker, R., & Hirshfield, S. (1998) The Analytical Engine: An Introduction to Computer Science Using the Internet. PWS Publishing, Boston.

Evangelidis, G., Dagdilelis, V., Satratzemi, M., & Efopoulos, V. (2001) *X-compiler: Yet another integrated novice programming environment* [Paper presentation]. Proceedings IEEE International Conference on Advanced Learning Technologies, pp. 166-169. IEEE.

Lane, A. (2017). Open Education and the Sustainable Development Goals: Making Change Happen, *Journal of Learning for Development - JL4D*, 4(3), pp. 275-286.

Miura, Y., Kaneko, K., & Nakagawa, M. (2004) *Development of an educational computer system simulator equipped with a compilation browser* [Paper presentation]. Proc. Int'l Conf. Computers in Education, pp. 1067-1071.

Stemler, S. (2001) An overview of content analysis. *Practical assessment, research & evaluation*, 7(17), pp.137-146.

Wiley, D., & Hilton, J. (2018) Defining OER-enabled Pedagogy. *International Review of Research in Open and Distance Learning*, 19(4).