Work in Progress–Improving Feedback Using an Automatic Assessment Tool

Daniel Jiménez-González, Carlos Álvarez, David López, Joan-M. Parcerisa, Javier Alonso, Christian Pérez, Ruben Tous, Pere Barlet, Montse Fernández, Jordi Tubella.
Computer Architecture Department. Technical University of Catalonia (UPC), Spain. {djimenez, calvarez, david, jmanel, alonso , christip, rtous, pbarlet, montsef, jordit}@ac.upc.edu

Abstract - **Students of Computer Science freshman year usually develop assembler programs to learn processor architecture. Homework exercises are done on paper, while those in lab sessions are solved with the aid of programming tools. Students perceive theory and lab as different subjects, so they don't use lab tools to test their theory solved problems. Moreover, during lab sessions, students often tend to ask for the teacher's guide and advice instead of using the debugging tools because these are new and unfriendly for them, and do not offer a quick and clear feedback. In this paper we present an automatic and friendly assessment tool, SISA-EMU, with a novel feature: exercise driven feedback with teacher's expertise. It provides correctness information and clues to help the students solve their most common mistakes for each individual problem (and not typical generic debug information) without the physical support of a teacher. SISA-EMU is currently in pre-deploy phase via a Moodle learning platform and we will have first evaluation results by the end of the current term.**

Index Terms – Feedback, automatic assessment tool, learning process, compartmental, assembler programming.

INTRODUCTION

The Computer Architecture courses' objectives of Computer Science freshman year in our university aim for students to learn basic concepts about the processor and memory hierarchy architecture, to program with an assembly language and to be able to use I/O synchronization mechanisms. Usually, simulators are used during these courses rather than real processors since they illustrate the basic assembly-level architectural concepts without the overhead or the cost of programming a real processor and its operating system.

The work we present in this paper is carried out during a course run by the Technical University of Catalonia's Barcelona School of Informatics during freshman year second term. In this course students learn the basic concepts of a computer architecture called SISA-F (Simple Instruction Set Architecture--Float), a 16-bit RISC multi-cycle processor designed in our school for learning purposes. We have a set of specific tools to assemble, link and debug SISA-F assembler programs, developed to resemble gnu programs *as*, *ld* and *ddd* [2].

During lab sessions, students don't like using the "debugging tools" because they are not familiar with them and seem to find them unfriendly; consequently, they often ask the professor for help to get a clear feedback on their mistakes and a possible solution. As a result, we end up having students depending on the teacher's feedback, and a teacher unable to properly or realistically attend all students simultaneously. Besides, the teacher is not physically available when the students are doing their homework either. All in all, the lack of a quick feedback in these cases discourages students from using lab tools on their own.

As a result, students appear to believe they master a subject simply by understanding the exercises solved by the professor in theory classes. Hence, when facing a homework exercise on paper, they simply assume that their solution is correct, without resorting to debugging tools to test it. Unfortunately, they do not perceive the lab and its tools as a means to back up theory, but as a different subject.

In order to evaluate this, students were surveyed to rank from 1 to 4 the statements shown in Table I, (where 1 stands for completely disagree, 4 for completely agree) regarding the simulator. As shown in Figure I, students considered the simulator as useful (statements 1 and 2). However, most of them recognized that they do not use the simulator out of the lab (statement 3), and that the simulator would be more useful with some kind of feedback (statement 4).

TABLE I
STUDENTS' SURVEY STATEMENTS

#	Statements
1	I think the simulator helps me to understand the subject
2	I think the simulator is useful to find an error in my program
3	I use the simulator to solve theory class' problems
4	I think the simulator will be more useful with feedback

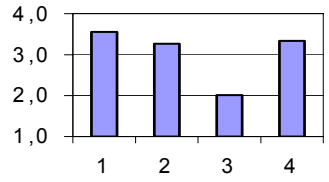

FIGURE 1
AVERAGE RANK OF THE STATEMENTS OF THE SURVEY

THE ASSESSMENT TOOL

SISA-EMU is an automatic assessment tool that offers exercise driven feedback to help students find and solve program mistakes when debugging a specific exercise [3].

978-1-4244-1970-8/08/$25.00 ©2008 IEEE

It works as a non-interactive debugger that executes and tests the assembler program following the commands written by the teacher in an exercise specific script. Apart from checking program correctness, its main goal is to identify the students' most frequently made mistakes and to output the appropriate feedback messages. Actually, since this feedback is similar to the one given by the teacher during lab sessions, knowing the most commonly made mistakes on each individual exercise is crucial for the teacher, because it allows the teacher to write useful command scripts to help students improve their learning process. This quality exercise driven feedback feature, with added teacher's expertise, is something that differentiates our tool from other simulator environments.

SISA-EMU has been developed in JAVA, and the Python scripting interface includes a full set of debugging commands, some of which are shown in Table II. The teacher's script may be set so it runs at every step, set breakpoints, check registers and memory addresses' value, monitor memory and I/O ports' accesses, and give feedback information on encountered mistakes during execution.

TABLE II
SIMULATOR COMMANDS

Type of command	Example
Execution	stepi, run, stop, reset, reload
Check Processor Values	mem_read, reg_read, sym_value, pc, disasm
Check System Values	clock_time, keyb_data, video_data, ptr_data
Check Events	io_access_subscribe, clk_add_timer
Modify Processor Values	mem_write, reg_write
Execution of Events	keyb_send, clk_pause, clk_resume
Breakpoint handling	breakpoint_add, returnpoint_add

Typical scripts have 60 lines or more. Due to the lack of space, we only show a very basic script in Figure 2.

```
stepi(1000)
if disasm(pc())!='halt' :
  print "ERROR: Your program does not finish,\
         check you have a halt instruction"
elif   sym_read16("RES")==-1 :
  print "ERROR: do you update RES variable? "
elif   sym_read16("RES")==13 :
  print "ERROR: RES should be 7 and not 13, \
 check the computation order of the expression"
elif   sym_read16("RES")==7 :
  print "CORRECT"
else :
  print "Undefined ERROR"
```

FIGURE 2
EXAMPLE OF A VERY BASIC TEACHER'S SCRIPT

The script in Figure 2 checks the SISA-F translation of a C program that makes the assignment `RES=x-y-3`, where variables `RES`, `x`, and `y` are global variables initialized to values *-1*, *4* and *-6* respectively. As it can be seen, the novelty of our tool is that it does not return an answer like "Res value is 13" or even "Res value is 13, the program is wrong" as debugging tools do. Thanks to the teacher's knowledge, it detects the more usual errors (in our simple example, computing `x-(y-3)`) and gives a hint about how to correct them: `"ERROR: RES should be 7 and not 13, check the computation order of the expression"`.

CURRENT AND FUTURE WORK

In this paper we present an automatic assessment tool: SISA-EMU. SISA-EMU assists teachers by automatically detecting students' programs most frequent mistakes and giving accurate and immediate exercise driven feedback to help them correct each of their programs. We expect this capability to encourage self-learning [1].

At the present time, SISA-EMU is in pre-deploy phase. The JAVA simulator has been totally implemented and we have developed the Python evaluation scripts for all lab exercises. Furthermore we have integrated it as a new module on a Moodle learning platform [4]. This integration allows our students to obtain feedback on their exercises even if they are working at home because they can access the system through Internet. Current term students are already using SISA-EMU as an experiment to evaluate its utility, correctness and correction scripts adequateness.

Our evaluation plan is: (1) to count the number of exercises correctly solved during the lab sessions, as well as the number of the most frequently asked questions to the teacher; and (2) to survey the students to evaluate the new tool, especially students that failed the previous term, when it was not yet available. With the help of this data we expect to figure out whether the current term students use the new tool to solve their homework and whether SISA-EMU is helping them improve their knowledge. Since we will also have data about students' accesses to our Moodle platform when they are at home, we will evaluate how the feedback capability is encouraging students to use lab tools by themselves. We will have those results by the end of the current term (July). By then, we expect to improve SISA-EMU based on our experience and the students' evaluation results.

Finally, we will develop scripts for all the theory classes' exercises and integrate them with SISA-EMU in the Moodle learning platform. Then students will have immediate feedback available for all exercises and we'll be able to record information on their most frequent mistakes. That will let them be more aware of their own learning and allow teachers to evaluate their teaching methodology.

ACKNOWLEDGMENT

We wish to thank our students for their patience and enthusiasm while conducting our experiments, and Ceila H. Sánchez for the proof-reading work. This work is supported by the Departament d'Innovació, Universitats i Empresa de la Generalitat de Catalunya (project 2007MQD 00203) and by the Barcelona School of Informatics (FIB).

REFERENCES

[1] Black, P., Wiliam, D. "Assessment and Classroom Learning", *Assessment in Education: Principles, Policy and Practice 5,* Vol 5, No 1, January 1998, 7-74.

[2] DDD project web page: http://www.gnu.org/software/ddd/, 2008.

[3] Moddle EC1 web page: http://gsi.ac.upc.edu/moodle/EC1/, 2008.

[4] Moodle web page: http://www.moodle.org, 2008.