Work in Progress–Improving Feedback Using an Automatic Assessment Tool

Computer Architecture Department, Technical University of Catalonia (UPC), Spain. {djmenez, 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 1, 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

<table>
<thead>
<tr>
<th>Statements</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
</tr>
<tr>
<td>2</td>
</tr>
<tr>
<td>3</td>
</tr>
<tr>
<td>4</td>
</tr>
</tbody>
</table>

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].

October 22 – 25, 2008, Saratoga Springs, NY
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>
<thead>
<tr>
<th>Type of command</th>
<th>Example</th>
</tr>
</thead>
<tbody>
<tr>
<td>Execution</td>
<td>start, mm, stop, reset, reload</td>
</tr>
<tr>
<td>Check Processor Values</td>
<td>mem_read, reg_read, sym_value, pc, disasm</td>
</tr>
<tr>
<td>Check System Values</td>
<td>clock_time, keyb_data, video_data, ptr_data</td>
</tr>
<tr>
<td>Check Events</td>
<td>io_access_subscribe, clk_add_timer</td>
</tr>
<tr>
<td>Modify Processor Values</td>
<td>mem_write, reg_write</td>
</tr>
<tr>
<td>Execution of Events</td>
<td>keyb_send, clk_pause, clk_resume</td>
</tr>
<tr>
<td>Breakpoint handling</td>
<td>breakpoint_add, retumpoint_add</td>
</tr>
</tbody>
</table>

| SIMULATOR COMMANDS |

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

```python
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 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”.

October 22 – 25, 2008, Saratoga Springs, NY

38th ASEE/IEEE Frontiers in Education Conference  
TIA-0