COURSE ASSESSMENT REDESIGN IN ASSEMBLY LANGUAGES

K. Jelemenská, P. Čičák, J. Parízková, M. Juríkovíč

Slovak University of Technology Bratislava, Faculty of Informatics and Information Technologies, Bratislava (SLOVAKIA)
jelemenska@fiit.stuba.sk, cicak@fiit.stuba.sk, parizkova@fiit.stuba.sk, jurikovic@fiit.stuba.sk

Abstract
Knowledge and skills assessment has always been a problem in the courses devoted to programming languages, whether it was assembly or higher level programming language. To master the programming language, the students should solve various problems using the language. Similarly, thorough knowledge and skills assessment during the exam should also be based on students’ programming activities. However, to correct and evaluate the programs manually by teachers or examiners is not an easy task. Unfortunately the choice of automated assignments assessment systems devoted to assembly languages is very limited. Therefore we have years of experience with manual assignments assessment. Due to the increased number of students, the manual evaluation of students’ programs written during the exam was not possible any more so we switched to the traditional on-line tests. Several years results showed this was not the ideal solution. This was the reason why we started the course assessment redesign.

The paper presents our first experience with course assessment redesign which in present situation covers the midterm tests and exam tests. In these tests we used the new types of test questions oriented more towards the skills-based assessment as well as unconventional Drag&Drop tests prepared in Moodle LCMS (Learning Content Management Systems) using the special Moodle extension recently developed at the Faculty of Informatics and Information Technologies, Slovak University of Technology in Bratislava. Although we did not return to actual programming during the tests, the presented results show clear improvement towards the more realistic values.

Keywords: Knowledge assessment, assembly language, on-line testing, Academic Information System, Moodle, Drag&Drop technique.

1 INTRODUCTION

The programming courses assessment process is often difficult to design in a way it provides objective and realistic image of students’ knowledge and skills. In programming course, no matter what programming language it covers, the students should write a lot of programs solving various kinds of assignments. The quality assessment process should not only evaluate the knowledge and skills the students gained but also give them a useful feedback that can improve them. That is why the programming courses assessment process often consist of several parts.

In lab exercises students are solving more or less individually allocated assignments. They can work on them at school or at home. The solutions are then evaluated by lab teachers and the evaluations usually form part of the overall course assessment. The second part is typically formed by an exam test, and sometimes the midterm tests can form the third part of the course assessment.

Several issues have to be taken into account concerning the mentioned assessment schema:

• It is not possible to prevent students trying to cheat in lab assignments (e.g. copy the solutions from Internet or classmate).
• Lab teachers can review and give the students some feedback concerning the solution quality. They also can check up students’ understanding asking them some checkout questions or asking for some minor changes in their solutions. However difficult and time consuming work it is, it can not completely prevent students from gaining points for non-authentic work.
• Different teachers might use various scales for assignments evaluation so the assessment objectivity can hardly be guarantied when there are several teachers teaching in the same course. Obviously, this can not be prevented in highly populated courses.
• Midterm tests and exam test should checkout not only the knowledge but also understanding and practical skills the students gained.
• The tests should be objective and suitable for large number of students.
Systems that automatically assess student programming assignments (requirements satisfaction, programming style, originality etc.) have been designed and used for over forty years [1][2][3][4]. However, they are typically designed for one or several related programming languages. As long as C++ or Java languages are covered massively, the choice of automated assignments assessment systems devoted to assembly languages is very limited. Therefore it is not an easy task to replace teachers, at least partially, in lab assessment evaluation process.

There is another possible viewpoint: we give the students the possibility to master the language by means of solving the assignments and receiving the feedback on their quality and consequently we will check the level they reached by means of high quality midterm tests and final exam test eventually. However, to ensure tests objectivity they should be evaluated automatically. Applications that objectively test and mark students’ programming skills were developed simultaneously with programming assignments evaluating systems [1][5][6][7]. Similarly to those systems only the most widespread programming languages are supported and we were not able to find a high quality application devoted to assembly language or even the HDLs (Hardware Description Languages).

Nowadays there is a common practice to use LCMS-type (Learning Content Management Systems) e-learning tools not only for course management, but also for on-line tests management. Most of the LCMS support only traditional types of test questions like multi-choice, True/False, gap filling, matching etc. These types of test questions are suitable when assessing students' knowledge and understanding, but are difficult to use for programming skills evaluation. What is more, nowadays LCMS do not typically support test and/or test questions design consistent with educational objectives taxonomy commonly adopted in pedagogical theory and practice [8][9]. Neither do they support knowledge assessment test development methodology or test life-cycle. So to design a high quality test in current LCMS is a difficult task.

That is why we concentrated our work on new applications development that would enable testing of knowledge, understanding, and especially the gained skills, all in an automated manner. Several applications have been designed, implemented and tested in the educational process at the Faculty of Informatics and Information Technologies, Slovak University of Technology in Bratislava, most of them oriented toward the HDL design and other behavioural specification means [10][11][12]. Some of them can be adopted to other areas as well, as it was done as part of the course assessment redesign in Assembly Languages course. The first results and experience with this process are presented in the paper.

2 COURSE ASSESSMENT HISTORY

Teaching of Assembly Languages course at Slovak University of Technology (STU) has a long tradition. The beginnings go back to the period of 40 years ago, when the courses dealing with Computer Science and Engineering have begun to be taught at STU. The biggest Assemblers education techniques development took place in the first half of the eighties and was associated with the development of microprocessor technology. The microprocessors and microcontrollers were exactly those things that have made available machine-level programming to other professional disciplines in addition to Computer Science and Engineering. As a result, assembly language teaching started also in other study fields and programs, which resulted in a large number of student on the courses. In comparison to the present days the education process was influenced by low status of available computational resources (computers, later Personal Computers) for exercises, practice and also for verification of knowledge and practical skills.

Since the beginning of teaching in Assembly Languages courses we put emphasis on the following areas:

• Understanding the general principles of machine-level programming independent on the specific processor type (codes i.e. internal data, numbers and instructions representation)
• Understanding the machine level of the specific processors, depending on the available computing resources for practical exercises. (Usually Intel x86 family processors)
• Understanding the principles of assembly language, its possibilities and way of usage
• Demonstration of practical skills in programming on machine level.

The course assessment process was influenced by a number of students and the availability of computing resources. Typically it consisted of three parts with the following significance:
• Individual programming of assigned tasks on practical exercises - 30%
• Midterm test in written form - 15%
• Final test in written form - 55%

The individual programming work was supported by teachers’ consultation and subsequent assessment. The tests (midterm and final) in written form were focused on theoretical knowledge demonstration in the areas of processor machine-level from the programmer's point of view (20%), internal data representation (10%) and assembly language compiler characteristics (10%) as well as on practical skills demonstration writing a program in assembly language (60%).

Each of the abovementioned parts was evaluated individually by teachers. In global course assessment we had to take into account a large number of students (around 300) which meant larger number of teachers who were engaged in the course (around 8). It is obvious that the evaluation was often highly subjective. Due to the lack of computers in such large number, the students were supposed to write a program on a piece of paper which was difficult to read, correct and evaluate. What is more, the students had no possibility to verify their programs. On the other hand the average course attainment evaluation almost followed the ideal Gaussian curve as illustrates Fig.1.

![Fig. 1 - Distribution of students’ results in Assembly languages course](image)

<table>
<thead>
<tr>
<th>Grade</th>
<th>Percentage</th>
</tr>
</thead>
<tbody>
<tr>
<td>A</td>
<td>7.83%</td>
</tr>
<tr>
<td>B</td>
<td>21.46%</td>
</tr>
<tr>
<td>C</td>
<td>28.76%</td>
</tr>
<tr>
<td>D</td>
<td>17.60%</td>
</tr>
<tr>
<td>E</td>
<td>11.97%</td>
</tr>
<tr>
<td>FX</td>
<td>12.28%</td>
</tr>
</tbody>
</table>

3 COURSE ASSESSMENT REDESIGN

With the expansion of LCMS the paper tests were replaces by on-line tests mostly with multi-choice questions. However, these on-line tests turned out to be much easier compared to paper tests. That is why the course assessment redesign was needed to get the results back to the realistic values. We used various types of test questions to prepare skills-based oriented tests, including the special Moodle activity, developed at the Faculty of Informatics and Information Technologies.

3.1 Multi-choice on-line tests

During the last 10 years the conditions that promote not only teaching, but also the possibility of testing and validation of knowledge and practical skills has been significantly changed. In order to eliminate the weaknesses and particularly the demanding work connected with students assessment, we have completely changed the system of testing and assessment implementation. In this second generation of assessment process the elimination of subjectivity caused by large number of teachers involved in assessment process was partially achieved by introducing the online tests. In order to cover all the previously set objectives that should be included in tests we have implemented the following classes of test questions:

• Binary Representation
• Hexadecimal Representation
• BCD Representation
• Direct and Two Complement Code Representation
• Floating Point Representation
• Little and BigEndian
• Flags & Jumps Instructions
• Assembly Language

It is obvious that these classes of test questions can verify the knowledge of machine level and assembler, but not the ability and skill in programming. Unfortunately, this part of the students’ assessment remained unchanged, that means based on an individual approach and programming assignments supported by teachers’ consultation and subsequent assessment. Obviously, the on-line tests were much easier compared to the previous paper tests as illustrates the average course attainment evaluation given in Fig. 2.

**Fig. 2 Distribution of students’ results in Assembly languages course - an average of the second generation of assessment process**

**3.2 Advanced on-line tests**

Based on the last years results in Assembly languages course attainment there was an urgent need to make the testing process more demanding to get the results that would realistically reflect the students’ knowledge and skill. With the introduction of the Academic Information System (AIS) at the Faculty of Informatics and Information Technologies in the academic year 2006/2007, the new possibilities for creating on-line tests opened up. AIS is an electronic system designed to support the educational process at Czech and Slovak universities. One of its parts “Test and examination” was designed for test creation and student examination. Before the test composition a test question base has to be prepared. There are fourteen test question types supported by AIS (see Fig. 3).

**Fig. 3 Test question types available in AIS**

Although we did not use the complete range of the available questions types we managed to prepare a relatively large test question base with a wide variety of test questions types. The questions are organized in six thematic groups. The most of the questions are of “filling in” type which turned to be
one of the most suitable manners of question presentation. The prepared test questions are summarised in Tab.1.

Tab.1. The summary of the created test question base divided according to the question types

<table>
<thead>
<tr>
<th>Thematic group</th>
<th>Question type</th>
<th>Macro</th>
<th>Data</th>
<th>Operands addressing</th>
<th>Procedure</th>
<th>Running of the code</th>
<th>Basic elements</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>M+1</td>
<td>2</td>
<td>0</td>
<td>2</td>
<td>1</td>
<td>3</td>
<td>4</td>
</tr>
<tr>
<td></td>
<td>M/0</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>7</td>
<td>0</td>
</tr>
<tr>
<td></td>
<td>AN</td>
<td>4</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>4</td>
<td>1</td>
</tr>
<tr>
<td></td>
<td>M</td>
<td>1</td>
<td>5</td>
<td>1</td>
<td>0</td>
<td>3</td>
<td>1</td>
</tr>
<tr>
<td></td>
<td>M+2</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>3</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td></td>
<td>M+1</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td></td>
<td>M+1</td>
<td>17</td>
<td>11</td>
<td>1</td>
<td>5</td>
<td>9</td>
<td>26</td>
</tr>
<tr>
<td></td>
<td>M+2</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td></td>
<td>M+2</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td></td>
<td>Sum</td>
<td>26</td>
<td>22</td>
<td>7</td>
<td>7</td>
<td>24</td>
<td>38</td>
</tr>
</tbody>
</table>

The new test question base was used to prepare the final exam test in the academic year 2009/2010 and formed about 50% of the test. Another 50% was covered by the previously used multi-choice questions. Fig. 4 and Fig. 5 illustrate the average test questions success rates based on the question types and thematic groups respectively.

![Fig. 4 Average test questions success rate based on the question types](image)

Comparing the old and new questions success rate we can see the distinctive drop (see Fig.6). However, to prepare a quality skills-based test question using the question types provided in AIS is quite a time consuming task. For example, when using the “filling in” question type sometime a lot of answers has to be added. An example is given in Fig. 7.
3.3 Alternative skills-based tests

Since we were not able to find any system supporting skills based assessment in the area of HDL modelling, in the last five years we devoted our work to several skills assessment systems development supporting languages like VHDL and SystemC. However, the Drag&Drop Moodle module [11] originally designed for skills-based tests in VHDL does have much wider utilization. It has been used as part of the student’s knowledge and skills assessment process in Specification and Description Languages course for last two academic years. The results were promising so we decided to incorporate it to Assembly languages course assessment process as well.

3.3.1 Drag&Drop Testing Module

The Drag&Drop testing module was designed as an extension to an existing Moodle LCMS [12], improving thus its assessment capabilities. The module is based on drag & drop technique, therefore very intuitive and easy to use with traditional mouse as well as stylus and touch-screen.

The main idea was to require the students to put together the correct program, based on the provided set of statements. Typically the program frame is already given and the students just complete the missing parts. Based on the verbal specification of the assignment the students have to study the program frame very carefully to understand the programmers intentions and the main ideas of the program. In this way they are learning the best programming practice while being tested at the same time. Of course, this can hold only in case the original program was written carefully, following all the
programming rules. That is why this kind of test is especially suitable for midterm tests oriented towards the specific programming skills the students should have already gained.

An undoubted advantage of this kind of test is the easiness of test assignment preparation. The first precondition is the working program, subprogram or only the program fragment together with its verbal or other non-formal specification. For testing purposes, especially for midterm tests, the short and relatively easy programs are the most suitable, covering the main and most important language constructs, the main principles and algorithms. This kinds of programs (program fragments) can even be chosen from the demonstrational examples in textbooks or lab-books. Using the tags `<droper>` and `</droper>` the teacher then marks in a text file editor all the source code lines that should be excluded from the code and replaced by the gaps. This text file is the main input of the Drag&Drop testing system. Any number of lines can be excluded from the code. This gives the teacher the possibility to prepare the test on various levels of difficulty. The weight of the excluded lines can be set to any positive integer number representing the number of points a student can gain for the correct placement of the line. The set of excluded lines, representing the correct possibilities, can be enriched by incorrect/confusing lines that will then be presented mixed together with the correct possibilities in a pool, were students will pick out the lines to replace the gaps in the code. This allows to make the test even more demanding. An example of an input source code fragment together with the resulting test assignment displayed during the test is given in Fig.8.

![Fig. 8 An example of Drag&Drop test (a) test question input (b) presented test question](image-url)
3.3.2 Assembly Languages Test Results

In Assembly languages course the Drag&Drop test was used for the first time during the midterm test in academic year 2009/2010. There were about 274 students involved. The midterm test was divided into two parts – online test with traditional test questions (5 point) and Drag&Drop test (10 points). Since the test took part during the lab exercises which means in three different days there were three different assembly language programs prepared and used for the Drag&Drop test (Task1 - Task3). Each of the programs was about 95 lines, there were 10 excluded statements and about 9 incorrect statements. The results of the test are summarised on the charts given in Fig. 9. The average test attainment of the three Drag&Drop tests is about the same (Fig. 9a). When compared to the online test (Fig. 9b) the average test attainment is also very similar. Obviously the test with only 10% of excluded lines is not a difficult one. However, the Drag&Drop activity enables to prepare the skills based tests on various levels of difficulty depending on the number and position of excluded lines as well as the number and quality of confusing statements. When compared to other Drag&Drop test that were used in Specification and description languages course (see Fig. 10), there is a distinctive dependency of average test attainment on the number of excluded lines as well as the number and quality of confusing statements.

![Fig. 9 Drag&Drop test attainment](chart_a.png) ![Fig. 9 Drag&Drop test attainment](chart_b.png)

(a) Drag&Drop test attainment in three assembly language tasks. (b) Drag&Drop test attainment compared to online test with traditional test questions.

4 RESULTS AND EXPERIENCE

If we compare the distribution of students’ results in Assembly languages course in the three generations of assessment process (Fig. 11) there is a clear shift of students results towards the better grading during the second generation assessment process when there were no programming tasks neither in midterm test nor in final exam test. In the current period the results follow more or less the typical Gaussian curve. We believe, we are back to the realistic grades reflecting the students’ actual knowledge and programming skills. However, the current question base will have to be upgraded regularly in order to keep the results in the present values.

![Fig. 10 Test attainment](chart.png)

Fig. 10 Test attainment dependability on percentage of excluded lines
5 CONCLUSIONS

In this paper our experience with Assembly languages course assessment redesign and the results compared to the previous years were presented. The results prove that the revised course assessment provides more realistic image about the students’ skills in the area of programming. Compared to the commonly used online tests the skills based tests are easier to prepare in the new unconventional application. The Drag&Drop tests bring number of advantages compared to the previous written tests. Firstly, there was the substantial reduction of time needed for tests evaluation. Secondly, the presented testing methods makes the students to study and compose a model program which contributes to learning the good programming practice. It is therefore essential for the model programs to be well written, obeying the programming rules. This makes the Drag&Drop tests especially suitable for midterm tests, when the students are in the middle of learning process. Finally, the Drag&Drop testing system enables to prepare the skills based tests on various levels of difficulty depending on the number and position of excluded lines as well as the number and quality of confusing statements.
ACKNOWLEDGEMENT

This work was partially supported by Slovak Science Grant Agency VEGA 1/0649/09 “Security and reliability in distributed computer systems and mobile computer networks” and VEGA 1/0822/08/04 “Intelligent Embedded Systems”.

REFERENCES


