Automatic Test-Based Assessment of Assembly Programs

Luís Tavares[1], Bruno Lima[1,2] and António J. Araújo[1,2]
[1]*Faculty of Engineering of the University of Porto, Porto, Portugal*
[2]*INESC TEC, Porto, Portugal*

Keywords: Arm, Assessment, Programming, Assembly.

Abstract: As computer science and engineering programs continue to grow in enrollment, automatic assessment tools have become prevalent. Manual assessment of programming exercises can be time-consuming and resource-intensive, creating a need for such tools. In response, this paper proposes a tool to assess assembly exercises, specifically ARM64 programs, and provide real-time feedback to students. The tool includes features for evaluating, analyzing, and detecting plagiarism in student submissions. After two years of intensive usage in a higher education environment, the results and analysis show a positive impact and potential benefits for teachers and students. Furthermore, the tool's source code is publicly available, making it a valuable contribution to building more effective and efficient automatic assessment tools for computer science and engineering schools.

1 INTRODUCTION

In the last half-century, there has been a growing interest in developing and deploying automatic assessment tools, ranging from assembly language programs on punched cards to web-based platforms. Programming assignments have become essential for understanding programming languages (Douce et al., 2005). Today, programming courses are offered in nearly all engineering schools in addition to computer science programs (Caiza and Álamo Ramiro, 2013).

As the number of students enrolling in computer science and engineering programs continues to increase, teachers face the challenge of providing accurate and timely feedback in a time-efficient manner (Marchiori, 2022). This paper proposes the AEAS (ARM [Extensible] Assessment System) tool for automatically evaluating ARM64 programs to address the time-consuming manual assessment of assembly exercises. The AEAS tool is a web-oriented platform designed to assess assembly exercises automatically and provide students with immediate feedback on their performance. The tool is built on top of the AOCO command-line system (Damas et al., 2021), containing features for plagiarism detection, analysis, and evaluation of student submissions.

The AEAS tool goes beyond traditional automatic assessment systems that offer only teacher-specific features. It provides a web-oriented platform that allows students to test their assembly programs' flow on the browser without installing any compiler or Integrated Development Environment. Assembly development environments, such as DS-5 (ARM, 2010), can be complex to configure, which may discourage novice users. The AEAS tool addresses this challenge by providing an intuitive platform for testing assembly exercises.

The main contribution of this paper is to present the AEAS assessment tool and its features.

Section 2 provides an overview of automatic assessment systems' background, while section 3 describes the system's functionalities from the perspectives of both students and teachers. The system's architecture is presented in section 4, and the grading process is discussed in section 5. Section 6 reports on the AEAS tool's experiences and analyses its results in an educational setting. Finally, section 7 concludes the paper and points out some future work.

2 BACKGROUND

Douce et al. (Douce et al., 2005) defined three generations of assessment systems. Firstly, the early assessment systems are grader programs that test the students' programs with either correct or wrong results. Secondly, tool-oriented systems are more complex grader programs, usually encapsulated in a ter-

Tavares, L., Lima, B. and AraÃžjo, A.
Automatic Test-Based Assessment of Assembly Programs.
DOI: 10.5220/0012129100003538
In *Proceedings of the 18th International Conference on Software Technologies (ICSOFT 2023)*, pages 572-579
ISBN: 978-989-758-665-1; ISSN: 2184-2833

minal interface or GUI. The second-generation tools also introduce features to students themselves to test their assignments. Lastly, web-oriented systems are the latter approach to automatic assessment systems.

Typically, these last-generation tools focus on more than just testing assignments. These tools propose utilities for managing and maintaining the assessments to permit teachers to understand how students are progressing, thus entitling teachers to make corrective efforts against students who may be encountering difficulties. The AEAS tool is a third-generation tool and has evolved from a second-generation one.

Looking at the state of the art, there are several automatic assessment evaluation tools with different characteristics and focused on different programming languages, such as:

- CodePost (codepost, 2023): CodePost is a web-based commercial tool that provides automatic grading and feedback for programming assignments. It supports multiple programming languages (but not assembly) and integrates with popular learning management systems(Mpungose and Khoza, 2022) like Canvas and Blackboard.
- Gradescope (Turnitin, 2023): Gradescope is a popular commercial platform for online grading and assessment, including programming assignments for multiple programming languages (but not assembly). It allows for various question types, including coding questions, and provides automatic grading and feedback tools.
- CodeRunner (CodeRunner, 2022): CodeRunner is a free open-source question-type plug-in for Moodle that can run program code submitted by students in answer to a wide range of programming questions in many different languages (assembly not supported). It is intended primarily for computer programming courses, although it can be used to grade any question for which the answer is text.
- Web-CAT: (Edwards and Perez-Quinones, 2008): Web-CAT is a web-based tool for automatically assessing programming assignments. It supports multiple programming languages (but not assembly) and provides various testing options, including input/output and black-box testing.
- BOSS (Joy et al., 2005): BOSS is a platform-independent system for submitting and grading programming assignments in C and Java. It includes a set of customizable program metrics, allows for online marking and feedback delivery, and has built-in plagiarism detection software.
- CourseMarker (Higgins et al., 2005): CourseMarker is a Java-based client-server evaluation tool that automatically evaluates Java and C/C++ programming assignments and diagrams.

Regarding these types of tools, (Tarek et al., 2022) recently carried out an extensive review where they identified a few more tools such as Mooshak (Leal and Silva, 2003) and EvalSeer (Nabil et al., 2021). However, none of the presented tools support the evaluation of assembly assignments.

However, in constructing this tool, we want to take advantage of our previous tool, AOCO (Damas et al., 2021), and what is best in current tools. The objective is to construct a free and open-source tool, thus allowing the community to adapt it to its needs.

3 FUNCTIONALITIES

The AEAS tool offers functionalities for two distinct types of users: students and teachers. Students use the tool to test the assignments. Teachers use the system to assess students' work, maintain assignments, and support administration demands. Before going into detail about the system's features, it is necessary to make a few considerations about the general operation of the system and how it is modeled.

The AEAS tool does not provide a direct channel for students to upload their assignments to be automatically graded by the system. The system exclusively provides students with a way to test their exercises. When the assignments reach the grading process, it is necessary to deliver and later store the students' solutions (typically a source file) in another infrastructure. This architectural decision is motivated to avoid requiring students to log into the assessment system. With these considerations in mind, it is possible to describe the grading progress of an assignment with the following workflow.

1. The teacher creates an assignment in the system. The assignment contains a set of public and private tests.
2. The teacher informs students that a new assignment is available in the system.
3. Students access the system and develop solutions for the assignment.
4. Students test their assignment solutions in the system against a set of public tests.
5. Students upload their assignment solutions to an external infrastructure before the deadline.
6. The teacher collects the assignment solutions from the external infrastructure.
7. The teacher grades the assignment in the system with the private tests and gives students feedback.

3.1 What Defines an Exercise?

This subsection defines an exercise inside the AEAS tool, going through its definition and structure. Exercises are the central component of grading students. Exercises must have a description of how to design the exercise, a list of parameters and returns, and a list of tests that map each group of inputs to a specific output. Teachers can add a list of private tests. These latter tests are merely used for student grading, while the system uses the default ones for student public feedback.

The exercise requires two definitions that work symbiotically, a viewing definition and a technical definition. The technical one includes the exercise parameter types, return types, and the list of tests. On the other hand, teachers must define a viewing definition that contains a name, a description of the work to be done, and, optionally, a label to categorize itself. Below are the explanation and examples of each viewing definition and the technical definition.

Name. This is the name of the exercise. It is the main field to identify an exercise.

Example: Sum of odd numbers in an array

Description. The description provides the students with a complete exercise description. It should guide students on the details of the exercise and contain all the information needed to develop it. Additionally, the system supports LaTeX, Markdown, and code snippets on this field to allow teachers to point more efficiently to particular aspects.

Example: Develop a subroutine to calculate the sum of the odd integers in an array. If there are no odd numbers, return 0.. A subroutine call can be as follows `oddSum(array, length)` where `int array = {1,4,7}` and `int length = 3` *and the expected output would be 8.*

Label. This optional field categorizes exercises. It can be helpful to differentiate distinct types of exercises and allow students to identify them better.

Example: Array problems

Technical Definition. The technical definition of the exercise defines how the exercise's solution must be modeled and how it should perform. The AEAS tool module that uses this technical definition provides an instruction manual [1] on defining an exercise.

[1]Repository of the AEAS grading module with the instruction manual on defining exercises. github.com/luist18/areas

Listing 1: Example of an exercise's technical definition.

```
# exercise definition
sumOdd:
    params: [array int, int]
    return: [int]

# tests definition
sumOdd:
  - inputs: [[], 0]
    outputs: [0]
    weight: 0.5
  - inputs: [[2,5,7], 3]
    outputs: [12]
    weight: 0.25
  - inputs: [[5,7], 2]
    outputs: [12]
    weight: 0.25
```

3.2 Student Functionalities

Students interact through the web interface. In the system, students have access to a list of the exercises, together with a succinct description and a pointer to their page. The name, description, label, author, and publication date are some properties that define an exercise. The students see these properties and an integrated code editor that allows them to develop their solutions inside the browser. Then, students can submit their approaches and acquire immediate feedback from the system on how their solution performed on the public tests. When students submit their exercise solutions, a set of unit tests verifies the program's validity. These unit tests are public tests that teachers define and do not cover all possible program flows. While students may use this as a means to test their exercises more efficiently, it is important to note that passing all public tests does not guarantee the complete validity of their solutions.

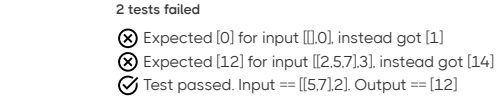

Figure 1: Output for the exercise and solution mentioned above.

Considering the previously defined exercise in the examples of section 3.1 that is assumed to sum odd integers, and a program which sums all integers regardless of their parity. Also, assume that the program does not check if the array is empty before reading from it. Thus, it will fail on the edge case of the empty array since it will try to read from an invalid memory address. With these considerations, the output for three test cases can be observed in figure 1.

3.3 Teacher Functionalities

Teachers have access to a large set of functionalities. This happens because teachers are the main actors maintaining and managing the system. Hence, this leads to a few distinct categories of functionalities that a teacher can perform. This section explains teachers' functionalities in each area.

Teachers are privileged users in the system. Therefore, to access their features, teachers must be previously registered in the system and then log in to the system's dashboard to access those features.

3.3.1 Administrative Functionalities

To give teachers an overall summary of the system's performance and operations, the AEAS tool offers teachers a dashboard to monitor information about the system's status. These details include CPU usage, memory usage, server uptime, and request traffic. In addition, the system also presents a functionality to set up other teachers' accounts.

3.3.2 Managing Functionalities

As teachers are the only administrative user of the system, they are responsible for managing almost every data stored in the system's database. This data includes the exercises as well as other information entries related to them. Exercises are the central information element of the system. Each database entry, except for teachers' authentication entries, is connected to an exercise. The system's information elements are the following:

- **Exercise** - represents the central information element of the system. Holds the exercise definition;
- **Label** - represents a designation to label exercises;
- **Submission** - represents an exercise submission. This element is mostly used for statistical purposes and later explained in section 3.3.4;
- **Batch Result** - represents the information of an assignment batch evaluation. The following section details this element. Holds the information about each student's unit-test results, plagiarism, and instruction.

The assessment system allows teachers to perform CRUD operations (Create, Read, Update, Delete) on the above items. Filling out forms on the system's dashboard enables teachers to create new instances of these information items. To read the information from the system, teachers can access a set of listings in the dashboard. Listings allow them to view the data stored and have a menu of quick actions to delete the element or navigate to the element's update/edit page.

3.3.3 Grading Functionalities

The grading functionalities are an important set of the system's functionalities. These are assessment systems' characteristic functionalities. This group comprises functionalities to test student assignments in three ways: unit-test-wise, plagiarism-wise, and instruction presence-wise. In addition, it also features an option to grade student assignments in a batch mode, allowing teachers to test unit tests, plagiarism, and instruction present at once. Details about the internal functioning of these functionalities are given in section 4.

The AEAS tool evaluates student assignments using unit tests. As previously stated, students have a collection of public tests to work with when designing exercises' solutions. Most of the time, public tests only cover some potential program flow of an exercise solution, leaving edge situations to private testing. The teachers then use a different set of tests, known as private tests, to evaluate the student's responses. With this in mind, the system enables teachers to design exercises that include the two kinds of tests. While students can only test their solutions in a public setting, teachers can evaluate the assignments in a more confidential configuration.

Delivering the same exercises to multiple students could lead to plagiarism. Plagiarism in coding assignments differs from other types of plagiarism (Agrawal and Sharma, 2016). It often happens that if the exercise assignment is very objective and, since there are not many workarounds in assembly solutions, students' programs are similar (Kustanto and Liem, 2009). Given this, the evaluation system allows teachers to choose a similarity level between pairs of assignments for each plagiarism analysis. The preceding allows teachers to raise the barrier when exercise solutions are likely identical and lower the threshold when solutions are expected to be highly diverse. The results of a plagiarism analysis advise the teacher on which pair of students exceed the defined barrier. These examples, however, are not free from manual analysis by the teacher because they may turn out to be false positives.

Another grading functionality is to check whether students have used a specific assembly instruction in their code submissions. Teachers may intend to evaluate the use of particular instructions in an assignment. For example, a teacher may want students to implement an activity with SIMD (Single Instruction Multiple Data) instructions. However, it could be possible to implement the same program without SIMD instructions. Despite completing the work, doing so would be immoral. Thus, to prevent these circumstances, the system checks for specific instructions in

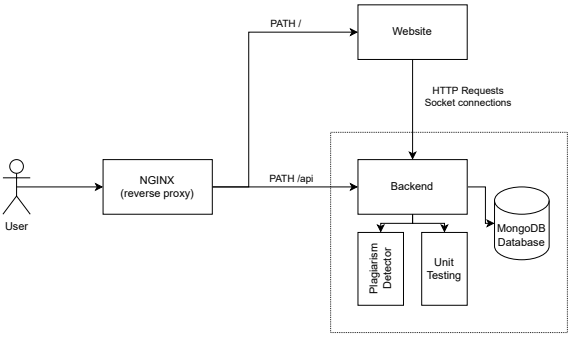

Figure 2: Architecture diagram of the AEAS tool.

assignment solutions and creates a report for each student.

Despite allowing to run the actions aforesaid individually, the system can run these actions in batch mode. The batch mode is advantageous when grading a whole assignment for a group of students, allowing the teacher to obtain a combined report for each student with all the task information.

3.3.4 Progress Functionalities

Tracking students' progress and performance are functionalities that third-generation systems should offer (Douce et al., 2005). The AEAS tool is no exception. The system offers a method to supervise students' progress.

Through a statistics section in the dashboard, the teacher can track how students progress in an exercise. For each exercise submission, the system stores the score for that exercise in the database. Later, the teacher can observe which tests students fail most and some score metrics for that exercise. The previous helped teachers shift efforts and tackle problems that students might encounter.

4 INTERNAL ARCHITECTURE OF THE AEAS TOOL

The AEAS tool is complex and substantially large. This section uses four made-up modules to help clarify the assessment system: web interface, backend, unit testing, and plagiarism detector. These modules do not exist in practice since they grow in more additional microservices. However, for explaining purposes, they are convenient.

As it is possible to observe in figure 2, apart from the modules mentioned above, two additional architectural modules represented in the diagram appear – reverse-proxy (NGINX) and the database (MongoDB). Firstly, the system uses a reverse proxy to proxy requests from the outside environment to the inner system backend environment and the website. Secondly, the system's database is a MongoDB cluster running alongside the system. The system's open-source architecture is available at github.com/luist18/computers-architecture-platform.

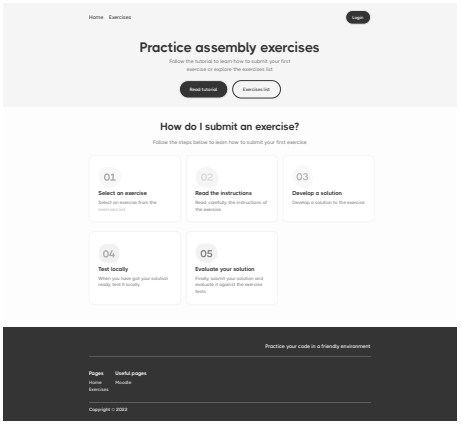

Figure 3: The AEAS tool Web-interface main page.

4.1 Web-Interface

The web-interface module communicates between end-users (students and teachers) and the system's backend. It is accountable for most of the system's functionalities, while some administration and maintenance features are exclusively accessible via backend API.

The web interface communicates between the users and the assessment system. This way, the interface supplies the essential features for users to communicate with the system. The functionalities available to students and teachers significantly differ in finality and complexity. Consequently, their visual interface implementation follows the same rationale. The student interface is designed to be simple and user-friendly, allowing them to select and complete exercises quickly. In contrast, the teacher interface is more complex, requiring a manual for initial interaction. The teacher interface mainly consists of forms and data listings for managing system data, accompanied by helpful text hints or instructions when necessary. Figure 3 shows the web platform implementation main page.

4.2 Backend

The backend of the system is essentially the core of the system, it does most of the work, and all system requests pass by it through the API. In this subsection, the responsibilities of the backend are discussed in the following order: providing endpoints through an API,

communicating with the internal modules, and managing files in the system's storage.

4.2.1 Rest API

The API is the external facade of the system and is accountable for all the communication between an end-user and itself. All interactions from the outside environment that affect the assessment system must go through the API. The API endpoints, which follow the RESTful architecture, must be used to communicate with the assessment system. Essentially, the API offers endpoints to create, read, update, and delete data either in the database or stored in the system, together with various endpoints to perform grading operations that do not necessarily impact the data already in the system.

4.2.2 Communication with Internal Modules

Aside from communicating with the end-users, the backend has to communicate with the modules encapsulated internally in the system's environment: plagiarism detector and unit testing module. These small modules perform extensive operations such as checking plagiarism between assignments or testing them against a set of use cases. Internally, a RESTful API encapsulates both modules. This way, the communication between the backend and these modules is performed via HTTP requests.

4.2.3 Managing System Local Storage

At last, another backend's responsibility is managing and storing information. For most data, the system stores it in the database. Although, student assignment files are not stored in the database but directly in the system's file system. Whenever a teacher evaluates a set of students, the system generates a directory with the students' files. Later, the database record for that evaluation holds a pointer to the files stored.

4.3 Unit Testing

As said in the previous subsection, the unit testing module is a RESTful API. Internally, the module uses an extended Python library built on the AOCO tool (Damas et al., 2021). Apart from performance changes, the modifications made to the tool add new data types to the exercise definition and JSON output to facilitate communication via HTTP. The API in this module only offers one endpoint employed to test an assignment against a set of test cases.

The library typically receives an exercise definition (*i.e.*, name, parameters' types, and return type), a set of tests, and an exercise solution. After receiving that data, the library runs the unit tests against the solution received in the parameter. It returns a JSON object containing information about the result of the unit tests.

4.4 Plagiarism Detector and Instruction Presence

In the same way as the unit testing module, the plagiarism detector module is also a RESTful API. This module offers two operations available through two endpoints: verifying occurrences of instructions in assignment files and checking plagiarism between assignment files.

The first operation verifies the occurrences of instructions, a simple operation that reads an assignment file and searches for any instruction dynamically specified as a parameter. The operation searches instructions in the code and determines if an instruction is in a student's code file.

The second operation verifies plagiarism between students' submission files. The AEAS tool implements a plagiarism detection system natively. The detection system uses a token comparison approach, one of the simplest and most effective source code plagiarism detection methods (Agrawal and Sharma, 2016). The system converts each file into tokens in a tokenization process to detect plagiarism. Tokenization removes comments and transforms equivalent code into the same token (*e.g.*, registers `x0` and `x1` are converted to `REG`). After tokenization, the tokens present in all submissions, such as the return token `RET`, are removed. The previous is made to avoid similarity smoothing. Finally, the approach uses the Sørensen–Dice coefficient to gauge the similarity between each pair of token sets.

5 GRADING

The AEAS tool grading system is semi-automatic when considering plagiarism detection. The previous means that, in some cases, it is inconceivable to grade students based on the system's output automatically. As plagiarism detections often come as false positives, it is impossible to directly map the system's results to a student's grade. Thus, this section covers an assignment's grading process without considering plagiarism detection. When using the tool, teachers should handle detected plagiarism manually and separately, and the final decision is up to each teacher's belief about each case.

The grading process of an assignment in the AEAS tool takes advantage of the batch mode explained in section 3.3.3. Teachers collect students' assignments and upload them to the system's batch mode. After the batch mode is complete, the system generates a report containing the information represented in table 1. There is also an option to export the report to a spreadsheet format.

Table 1: Assignment report example.

ID	Compiled	$Test_1$	$Test_2$	$Test_3$	$Test_4$	Instruction	Score
$student_1$	true	1	1	0	0	true	0.5
$student_2$	true	1	1	1	1	true	1
$student_3$	false	0	0	0	0	true	0
$student_4$	true	0	0	1	0	true	0.25
$student_5$	true	1	1	1	1	false	0

The score value for each student depends on unit tests and instruction presence. Considering an assignment a and an arbitrary student i, the score for a student in a specific assignment is the following:

$$score(s_i,a) = [\sum_{j=1}^{T} test_a(j,s_i) \times w_a(j)] \times instruction(l,s_i) \quad (1)$$

$$test_a(j,s_i) = \begin{cases} 0, & \text{if } s_i \text{ does not pass test } j \\ 1, & \text{if } s_i \text{ passes test } j \end{cases} \quad (2)$$

$$instruction(l,s_i) = \begin{cases} 0, & \text{if } s_i \text{ does not contain } l \\ 1, & \text{if } s_i \text{ contains } l \end{cases} \quad (3)$$

$$\sum_{j=1}^{T} w_a(j) = 1 \quad (4)$$

Where s_i is the source code solution for the student i, $test_a(j,s_i)$ is the result of the unit test j for the assignment a, T is the number of unit tests, $w_a(j)$ is the weight of test j for the assignment a, and $instruction(l,s_i)$ is the result of the instruction lookup for the instruction l and the source code s_i. The score has a domain between 0 and 1. Sometimes, verifying if an instruction is in the students' assignments is unnecessary. In those cases, the score function does not consider the instruction function.

6 VALIDATION

Since the 2020 academic year, the Faculty of Engineering of the University of Porto has used the AEAS tool in the Computers Architecture course of the Bachelor in Informatics and Computing Engineering to help students progress and automatically grade the assembly assignments. On both occasions, the students had to complete weekly tasks to ease learning the assembly programming language.

To evaluate the AEAS tool's effectiveness, students answered a questionnaire regarding some aspects of the AEAS tool and the evaluation methodology. This section firstly analyzes the common questions for the 2020/2021 (Y1) and 2021/2022 (Y2) academic years, and then each year separately. On a 563 students universe (257 from Y1; 306 from Y2), 93 students (46 from Y1; 47 from Y2) answered the questionnaire – making an overall coverage of 16.52% (17.90% in Y1; 15.35% in Y2).

Table 2: Results for questions 3, 4, and 5.

Question	1	2	3	4	5
3 (Y1)	8.7%	8.7%	65.2%	8.7%	8.7%
4 (Y2)	8.5%	6.4%	17.0%	23.4%	44.7%
5 (Y2)	4.3%	10.6%	27.7%	31.9%	25.5%

The questionnaire questions are the following:

Q1 (Year 1 & 2) Do you consider mandatory weekly assignments would be beneficial to the process of learning ARM64 programming?

Q2 (Year 1) Did the web-oriented platform positively impact the development of your solutions to the weekly assignments?

Q3 (Year 1) On a scale of 1 (only used DS-5) to 5 (only used the web-oriented platform), how do you use these tools to test the assignments? (Level 3 corresponds to a balanced use.)

Q4 (Year 2) On a scale of 1 (very difficult) to 5 (very easy), how would you evaluate the ease of programming in the web-oriented test platform compared to the DS-5 IDE?

Q5 (Year 2) On a scale of 1 (only used DS-5) to 5 (only used the web-oriented platform), how do you consider your environment to develop and test the assignments? (Level 3 corresponds to a balanced use.)

Table 3: Results for questions 1 and 2.

Question	Yes	No	No opinion
1 (Y1)	93.5%	4.3%	2.2%
1 (Y2)	82.9%	12.8%	4.3%
2 (Y1)	89.1%	8.7%	2.2%

As shown in table 3, most students believed that the weekly assignments positively impacted learning the ARM64 programming language. Although, only some of them considered that it had no impact on understanding the ARM64 programming language.

Table 3 also shows that almost nine in every ten students feel that the AEAS system positively impacted the solutions for the weekly assignments. The previous confirms the usefulness of the AEAS system in helping students. Finally, regarding the first year, it is observed in table 2 that students took a balanced use

between DS-5 and the assessment system to develop their solutions.

Concerning the second year, table 2 shows an overall opinion that the web-oriented test platform is easier to use than DS-5. Finally, the table also shows that compared with the assessment system in the previous year, there was a shift towards using more the assessment system than DS-5.

These results help confirm the system's validity and that students can shift their programming environment to mostly the assessment system with more student-directed changes and easier processes to develop exercises.

7 CONCLUSIONS

This paper proposes a novel and open-source automatic assessment tool for assembly exercises, the AEAS tool. This highly configurable tool was developed to be used in a teaching environment and to support the assessment of exercises in courses with a high number of students.

The results of the validation, carried out using the students who used the system, revealed that it significantly impacted their understanding of the ARM64 assembly language. It allowed students to acknowledge almost instantaneous feedback on a process that before was slow and complex. The AEAS assessment tool has become a compelling choice for automatic assessment. Its functionalities were tested and refined in a higher education environment over two years. The results show that it is a reliable tool for grading programming assignments written in ARM64.

Overall, this work demonstrates the value and effectiveness of the AEAS tool as a solution for automatic assessment in computer science and engineering. Having the system available and open-source opens the possibility of community improvements and usage of the same tool in other educational environments.

As future work, there should be work in improving scalability, enhancing student features with more configurable exercise listings, and improving accessibility to turn the system even more uncomplicated. Since the system's architecture is highly modular, there is likewise an open way to add compatibility with other assembly languages, such as RISC-V.

REFERENCES

Agrawal, M. and Sharma, D. K. (2016). A state of art on source code plagiarism detection. In *2016 2nd International Conference on Next Generation Computing Technologies (NGCT)*, Dehradun, India.

ARM (2010). ARM DS-5 Getting Started with DS-5 Version 1.0.

Caiza, J. C. and Álamo Ramiro, J. M. d. (2013). Programming assignments automatic grading: review of tools and implementations.

codepost (2023). *codePost*. Available at https://codepost.io/.

CodeRunner (2022). *CodeRunner*. Available at https://coderunner.org.nz/.

Damas, J., Lima, B., and Araujo, A. J. (2021). AOCO - A Tool to Improve the Teaching of the ARM Assembly Language in Higher Education. In *2021 30th Annual Conference of the European Association for Education in Electrical and Information Engineering (EAEEIE)*.

Douce, C., Livingstone, D., and Orwell, J. (2005). Automatic test-based assessment of programming: A review. *Journal on Educational Resources in Computing*.

Edwards, S. H. and Perez-Quinones, M. A. (2008). Web-cat: automatically grading programming assignments. In *Proceedings of the 13th annual conference on Innovation and technology in computer science education*.

Higgins, C. A., Gray, G., Symeonidis, P., and Tsintsifas, A. (2005). Automated assessment and experiences of teaching programming. *Journal on Educational Resources in Computing*.

Joy, M., Griffiths, N., and Boyatt, R. (2005). The boss online submission and assessment system. *Journal on Educational Resources in Computing*.

Kustanto, C. and Liem, I. (2009). Automatic Source Code Plagiarism Detection. In *2009 10th ACIS International Conference on Software Engineering, Artificial Intelligences, Networking and Parallel/Distributed Computing*, Daegu.

Leal, J. P. and Silva, F. (2003). Mooshak: A web-based multi-site programming contest system. *Software: Practice and Experience*.

Marchiori, A. (2022). Labtool: A Command-Line Interface Lab Assistant and Assessment Tool.

Mpungose, C. B. and Khoza, S. B. (2022). Postgraduate students' experiences on the use of moodle and canvas learning management system. *Technology, Knowledge and Learning*.

Nabil, R., Mohamed, N. E., Mahdy, A., Nader, K., Essam, S., and Eliwa, E. (2021). Evalseer: An intelligent gamified system for programming assignments assessment. In *2021 International Mobile, Intelligent, and Ubiquitous Computing Conference (MIUCC)*.

Tarek, M., Ashraf, A., Heidar, M., and Eliwa, E. (2022). Review of programming assignments automated assessment systems. In *2022 2nd International Mobile, Intelligent, and Ubiquitous Computing Conference (MIUCC)*.

Turnitin, L. (2023). *gradescope*. Available at https://www.gradescope.com/.