Integrating Continuous Assessment into Undergraduate Computer Architecture using Automated Grading

Lucas Wanner
Institute of Computing – UNICAMP
Campinas, Brazil, 13083-852
Email: wanner@unicamp.br

***Abstract*—Continuous assessment (CA) improves student engagement and understanding through regular evaluations and rapid feedback. This approach was integrated into a foundational Computer Architecture course using frequent quizzes. To manage large class sizes and limited teaching assistance, automated grading software was used. This paper discusses the implementation of automated grading for quizzes, detailing the process, and presenting course results and student feedback. Observations based on student feedback and outcomes suggest that integrating CA through quizzes is beneficial for student engagement and learning.**

I. Introduction

Continuous assessment (CA) is a teaching strategy that incorporates regular evaluations throughout a course, rather than relying solely on final exams. CA includes frequent assignments that enable students to engage consistently with the material, promoting a deeper understanding of the subject [1]. A key component of CA is quick feedback, which allows students to identify and address their weaknesses promptly, leading to improved learning outcomes. Furthermore, CA allows for comprehensive coverage of the curriculum and allows instructors to adjust their teaching strategies based on student performance, creating a more adaptive and effective learning environment.

The Design of Computing Systems course at Unicamp (MC732) provided an opportunity to apply the benefits of continuous assessment in a foundational computer architecture class for students in computer science and engineering. Based on the "Computer Organization and Design" book by Patterson and Hennessy [9], this 60-hour course covers performance evaluation, Instruction Set Architecture (ISA) design, single-cycle and pipelined microarchitecture, caches and memory hierarchy, multicore and a brief introduction to advanced microarchitectures. As the last mandatory course in the computer organization sequence, following prerequisites in computer organization and digital systems, it serves as a critical component for student formation, particularly for those pursuing advanced study in related fields.

In an effort to introduce continuous assessment in recent years, MC732 instructors have implemented frequent quizzes that cover specific topics throughout the course. A typical quiz consists of a few multiple choice questions or quantitative problems that require numeric answers. These quizzes are planned for 20-30 minutes and are administered at the end of a class, focusing on topics discussed in the previous 2-3 classes. If students perform well on a quiz, receiving a grade of 70% or higher, they are allowed to skip the corresponding questions on the final exam, with their quiz grade applied to those exam questions. In some editions of the course, quizzes also account for a small fraction of the final grade.

Despite these efforts, with an average of approximately 100 students per class each semester and no teaching assistance, the difficulty of grading tests and providing rapid feedback in MC732 is substantial. Without quick assessment, quizzes do not effectively contribute to CA, as a lack of timely feedback prevents students and instructors from identifying and addressing learning gaps. This hinders the ability to adapt the course to meet students' needs, diminishing the potential benefits of CA.

This paper relates an experience of using automated grading for in-person and online quizzes in a large undergraduate computer architecture course. It presents the use of Auto Multiple Choice (AMC) [4], a software package for creating and grading multiple choice tests, in the Design of Computing Systems course at Unicamp over the last few years. AMC generates individual exams from a bank of questions written in LaTeX , varying both the questions and the numeric variables for quantitative problems. Students fill in the answers on paper forms that are then scanned and graded by AMC. For online quizzes, students download a form and submit their answers through a web system. Students typically receive their graded quizzes via email the day after the exam. The instructor can use statistics from the exams, including the distribution of grades and the difficulty of each question, to improve the course and revise topics when necessary. The use of objective questions and automated grading allows the scale of the evaluation to expand, allowing students to typically solve 70–100 individual questions over the semester.

The paper describes the process of creating questions and generating and grading quizzes and exams, introducing examples of multiple choice and quantitative questions for different topics in the course. Additionally, it discusses the results of quizzes and exams, and the impact of quizzes on final grades. Finally, it presents a summary of student feedback on the use of quizzes in the course.

II. Computer Architecture Teaching at Unicamp

Computer Systems Design (MC732) is the final mandatory undergraduate course in hardware systems for Computer Science (CS) and Computer Engineering (CE) students at the University of Campinas (Unicamp). The course follows the Computer Organization and Design textbook by Patterson and Hennessy [9], and provides an overview of key topics in modern computer architecture.

Before taking MC732, students are required to complete courses in digital systems and computer organization. Introduction to Digital Systems (MC602) focuses on digital logic design, covering combinational and sequential circuits, as well as basic circuit design using Hardware Description Languages. Introduction to Computer Organization (MC404) covers computer organization, instruction sets, assembly programming, memory addressing, input/output operations, and interrupt handling.

Building on this foundation, MC732 covers fundamental aspects of computer architecture and microarchitecture. The course offers an in-depth exploration of Instruction Set Architecture (ISA), with an emphasis on RISC-V, which students are introduced to in MC404. There is a significant focus on microarchitecture, including single-cycle and pipelined datapath designs, with additional discussions on deep pipelines, superscalar, and out-of-order execution. Memory hierarchy is covered with a focus on caches and virtual memory. Finally, the course briefly introduces multicore architectures and GPUs.

By the end of the course, students are expected to have a solid understanding of modern computer architectures, including the ability to analyze the design of basic processors, especially pipelined architectures. In addition, students should be able to assess the potential performance impact of optimizations in various parts of a system, such as accelerating specific instructions, including more processing cores, or adding additional levels of cache. For further study, students are encouraged to take elective courses such as Parallel Processing and graduate-level Computer Architecture I and II, which are also available to undergraduates.

The evaluation of MC732 is based primarily on exams, as there are no practical assignments in the course. Although this method allows for comprehensive testing of theoretical knowledge, it can make it difficult to ensure ongoing student engagement and immediate feedback on their understanding of the material. To address these challenges, instructors have introduced frequent quizzes as part of a continuous assessment (CA) approach. These quizzes, which include but are not limited to the ones listed below, assess students' grasp of recent topics and contribute to their overall grade.

1) Performance and CPI Calculation: Focuses on measuring and calculating computer system performance, using Cycles Per Instruction (CPI) as a metric. Requires students to differentiate metrics related to performance such as time, frequency, cycles, MIPS, etc.
2) ISA and Code Generation: Requires students to understand and write simple code snippets in the RISC-V assembly language, demonstrating an understanding of the ISA and code generation for basic templates such as if and for statements.
3) Application binary interface: Tests knowledge of the Application Binary Interface (ABI), including calling conventions, register allocation, and stack management.
4) Arithmetic for Computers: Covers numeric representation and the design of arithmetic units within computers, including ALU design, adders, parallel and multi-cycle multiplication, and division.
5) Datapath: Control and data signals: Explore how control signals direct operations within the datapath, including the values of control and data signals for various instructions.
6) Pipeline: Hazards and Forwarding: assesses understanding of data and control hazards in pipelined processors to performance, and how techniques like forwarding and branch prediction can be used to mitigate these hazards.
7) Pipeline: Control and data signals: Requires students to apply pipelined execution concepts by identifying the state of the pipeline and determining which instructions are active in each stage during a given cycle.
8) Cache: Direct-mapped Organization: Focuses on the principles and operation of direct-mapped cache organization, including address address mapping and aliasing for and varying capacity and block sizes.
9) Cache: Misses and Performance: Examines the impact of cache misses on system performance, with an emphasis on understanding how cache design affects efficiency.
10) Virtual Memory: covers virtual memory concepts, including address translation, page tables, and the management of memory in modern computer systems.

These quizzes are designed to reinforce key concepts covered in lectures, providing students with regular opportunities to apply their knowledge and prepare for more comprehensive exams. However, practical implementation of quizzes faces the significant challenge of managing and grading a large number of quizzes for approximately 100 students, especially without teaching assistance. This workload can lead to delays in providing feedback, reducing the effectiveness of continuous assessment (CA) by limiting the ability to quickly identify learning gaps and adjust course content accordingly.

Automated grading offers a solution to this challenge by expanding the scope of evaluation and allowing for rapid feedback. Using tools such as Auto Multiple Choice (AMC) [4], instructors can efficiently generate both paper and digital quizzes, streamline the grading process, and ensure that students receive timely feedback. This approach not only supports continuous assessment, but also enhances the overall learning experience by allowing for quick course adjustments based on student performance. The next section discusses how AMC can be used to implement a continuous assessment strategy supported by frequent quizzes.

III. AMC Workflow

Auto Multiple Choice (AMC) is a LaTeX package and toolset designed to streamline the generation and grading of exams, particularly for multiple choice and numeric questions. The exams are then assembled by selecting questions that match the desired difficulty and topic coverage. These exams can be distributed in either paper or digital format. In paper format, the quizzes are printed, completed by the students, and scanned for grading. Each exam includes a section for student identification and a unique quiz identifier. In the digital format, students receive a unique, pre-identified quiz, which they complete using a PDF viewer and submit through an online Learning Management System.

AMC processes scanned or digital submissions, recognizing filled checkboxes and numeric answers. Errors in scanning or recognition can be corrected manually by rescanning or adjusting the recognized answers within AMC. The system automatically grades multiple choice and numeric questions, with provisions for correcting grading errors or adjusting point allocations. Open-ended questions are manually graded.

Finally, AMC generates a spreadsheet for the instructor detailing the results for each student and question. This allows for the identification of inconsistencies in form completion and the assessment of question difficulty. If an email client is connected to AMC, students can receive a marked PDF of their exam, including scores and any corrections made during grading. The original submission, with corrections, is also provided for review. Students can review their mistakes and appeal their grades if necessary. Instructors may also include explanations of correct answers in the marked PDFs. The following section details the creation of questions and exams using AMC.

IV. Questions and Quizes

Auto Multiple Choice (AMC) supports different types of questions, including simple multiple choice, multiple choice with multiple answers, numerical questions, and open questions graded manually by the instructor. Each question is assigned a unique name and can be associated with a specific topic (or element in AMC). By grouping questions under the same topic, AMC can randomly select one or more questions when generating a test. In this section, we introduce examples for each type of question. The typeset output of the questions is presented in a sample quiz in Section IV-E.

A. Multiple Choice Questions

Figure1 illustrates a multiple choice question designed to assess students' understanding of Amdahl's law. The question asks students to determine the required speedup for multiplication instructions in a program where 60% of the execution time is spent on these operations, in order to achieve a doubling of the program's overall speed. To correctly answer, the student must calculate the potential performance improvement when optimizing only a part of the system.

Multiple versions of this question can be generated by varying the numerical values provided, such as the percentage of time spent on multiplication or the desired overall speedup. This can be done by creating multiple versions of the question with different hardcoded values or by using variables and formulas to dynamically generate the numbers, as described in Section IV-B. Additionally, Python can be used to programmatically generate these values, providing even greater flexibility, as outlined in Section IV-C. This diversifies the questions and also plays a role in preventing cheating by reducing the likelihood that students will encounter identical questions.

```
\element{basic-amdahl}{
 \begin{question}{amdahl1}
 If a program spends 60\% of its time executing
    multiplication instructions, how much faster do
     the multiplication instructions need to be for
     the program to run 2 times faster?
  \begin{choices}
   \correctchoice{6x faster.} \scoring{1}
   \wrongchoice{3x faster.}
   \wrongchoice{12x faster.}
   \wrongchoice{It is not possible to achieve the
    desired performance by optimizing only the
    multiplication instructions.}
   \wrongchoice{It is not possible to determine the
     answer with the given information.}
  \end{choices}
 \end{question}
}
```

Fig. 1. Multiple choice question. A single correct choice is defined with a score of one. The alternatives are shuffled for each exam. Questions with the same element identifier can be randomly selected for different exams.

```
\element{basic-short}{
  \begin{question}{reg-bits}
     How many bits are needed to address a register
     in the RISC-V register file?
    \begin{choiceshoriz}
      \correctchoice[5]{}
      \wrongchoice[16]{}
      \wrongchoice[32]{}
      \wrongchoice[6]{}
      \wrongchoice[8]{}
    \end{choiceshoriz}
  \end{question}
}
```

Fig. 2. Simple multiple choice question. The correct answer is shown inside the answer box.

Simpler multiple choice questions can present the alternatives in the answer box itself. Figure 2 presents a straightforward question of information encoding in the RISC-V Instruction Set Architecture (ISA), focusing on how many bits are needed to address a register in the register file. Variations of the question can assess related concepts, such as determining the number of registers given a specific bit length or calculating how many instructions can be encoded with a given opcode length.

It is also possible to define questions where multiple answers may be correct, or none at all. Finally, for all multiple

```
\element{num-fp}{
    \begin{questionmultx}{fpcv1}
      Convert the number 0xc0980000 representing a
    single-precision floating point value to
    decimal.
    \begin{center}
        \AMCnumericChoices{-4.75}{sign=true,
    vertical=true,base=10,digits=4,decimals=2,
    scoreexact=1}
    \end{center}
    \end{questionmultx}
}
```

Fig. 3. Quantitative question with a signed decimal answer. Only exactly correct responses are considered.

```
\FPeval{\blp}{round(random*4+1,0)}
\FPeval{\blw}{round(2^\blp,0)}
\FPeval{\ccp}{trunc(\blp+random*4+4,0)}
\FPeval{\ccb}{round(2^\ccp,0)}
\FPeval{\idx}{round(\ccp-\blp-2,0)}
\FPeval{\off}{round(\blp+2,0)}
\FPeval{\tag}{round(32-\off-\idx,0)}

\begin{questionmultx}{indexsize}
    Consider an architecture with 32-bit words and
    a direct-mapped L1 data cache with \ccb{} bytes
     capacity. If the cache has blocks of \blw{}
    words, how many bits are required for the index
     field in the address division?
    \begin{center}
        \AMCnumericChoices{\idx}{sign=false,
    vertical=false,nozero=true,base=10,digits=1,
    scoreexact=1}
    \end{center}
\end{questionmultx}
```

Fig. 4. Quantitative question with random variables and response determined by a formula.

choice questions, points can be awarded or deducted for each individual answer, allowing for a more precise assessment. This approach helps to reward partial knowledge or penalize significant errors, providing a more accurate assessment of student comprehension.

B. Quantitative Questions, Variables, and Formulas

AMC allows the creation of quantitative questions in which the expected result is defined as a number. Figure 3 shows an example in which students are asked to convert a hexadecimal number from single-precision floating point representation to its decimal equivalent. The specified parameters indicate that the answer must be provided as a signed number in decimal representation (base 10), using 4 digits in total, with two decimal digits. Only exactly correct responses are considered for full credit. It is possible to configure numerical tolerances for correct responses as well as partial credit for approximately correct responses.

As with multiple choice questions, multiple variations of numeric questions can be created for the same topic, such as using different values or asking students to convert from decimal to floating-point representation instead. For questions in which the answer can be derived from a formula involving variables, the Fixed Point (FP) package [8] can be used. This package allows for the random selection of variable values and the automatic calculation of the correct answer, enabling the generation of diverse question sets that test the same concept without requiring the creation of multiple versions of each question.

The question in Figure 4 assesses the student's understanding of how memory addresses are divided in a direct-mapped cache architecture. It presents an architecture with 32-bit words and an L1 data cache with a capacity of ccs bytes, organized into blocks containing blw words. The student must calculate the number of bits required for the index field in the address division. The values for cache size and block size are randomly chosen using the LaTeX Fixed Point (FP) package. The sizes of the index, offset, and tag fields are calculated and could be used to generate further questions. In this example, only answers between 1 and 9 are accepted, so care must be taken to ensure that the correct answers fall within these bounds.

C. Integration with Python

Integration with Python code is possible using the pythontex package [10], which allows Python code to be executed and its output to be included directly in the document. This is useful for generating dynamic content, such as questions and answers based on complex calculations.

Figure 5 demonstrates this integration with a question about multicycle division. The first Python function calculates the value for each register in a multicycle divider, and the second function generates a question for a specific register and cycle. Random input values are selected, and a question asking for the state of the Remainder register at a given cycle is inserted into the document.

To produce the final document, LaTeX is preprocessed with PythonTeX, which executes the Python code and inserts the output into the LaTeX file for AMC processing. Although debugging can be challenging, this approach greatly simplifies the creation of complex questions.

D. Open Questions

AMC allows for the combination of multiple choice and numeric questions with open-ended questions that require manual grading by the instructor. In these cases, after students complete the exam, the instructor manually checks and grades open-ended responses before processing the files for automated grading. Figure 6 illustrates an example of an open question in which students are asked to write a function using RISC-V assembly. This approach ensures that, while objective questions are automatically graded, the instructor systematically evaluates more complex and essay answers.

E. Quiz and Exam Organization

Figures 7 and 8 show examples of paper and digital quizzes, respectively. The process of question selection and exam composition is the same for both formats.

```
\begin{pycode}
def mc_div(ddend, dsor, b, cycle):
    dsor <<= b
    quotient = 0
    remainder = ddend
    if cycle == 0 :
        return  {'Quotient':quotient,
                 'Remainder':remainder,
                 'dsor':dsor}
    for i in range(1,b+2):
        remainder = remainder - dsor
        if remainder < 0:
            remainder = remainder + dsor
            quotient <<= 1
        else :
            quotient <<= 1
            quotient += 1
        dsor >>= 1
        if i == cycle:
            return  {'Quotient':quotient,
                     'Remainder':remainder,
                     'dsor':dsor}

def div_q(reg, ddend, dsor, b, cycle, pts):
    ans = mc_div(ddend, dsor, b, cycle)[reg]
    bsize = b * 2 if (reg == 'dsor') else b
    print("""
    \\begin{questionmultx}{Division%s%d}
    \\\\%s, Cycle %d
    \\begin{center}
    \\AMCnumericChoices{%d}{sign=false,vertical=
    true,base=2,digits=%d,scoreexact=%f}
    \\end{center}
    \\end{questionmultx}
    """ %(reg, cycle, reg, cycle, ans, bsize, pts))
\end{pycode}

\element{divider}{
\pyc{import random; ddend=random.randint(11, 14);
    dsor = random.randint(3, 5);}

For the next question, considering a multi-cycle
    divider, divide \py{ddend} by \py{dsor},
    indicating the value of the specified register
    at the end of the clock cycle. Assume that the
    inputs and results are represented with 4 bits,
     and give your answer in binary.

\pyc{div_question('Remainder',ddend,dsor,4,3,1)}
}
```

Fig. 5. Integration with Python Code. The first Python function calculates the value for each register in a multicycle divider, and the second function generates a question for a specific register and cycle. Finally, random input values are selected and a question is inserted.

```
\element{asm}{
\begin{question}{open-asm}
 Implement the \texttt{int strlen(const char* str)}
     function in RISC-V assembly. %...
\AMCOpen{lineheight=0.7cm,lines=5}{
\wrongchoice[0]{0}\scoring{0}
% ...
\correctchoice[5]{5}\scoring{5}}
\end{question}
}
```

Fig. 6. Open question. Prior to processing, the instructor must manually grade the question using the reserved field.

Questions can be categorized into different topics, called elements. Within each topic, questions may be shuffled and randomly selected, allowing for variability in the quizzes. In addition, multiple topics can be combined to form new topics, offering flexibility in exam design. The final exam is composed by inserting a set number of questions, or all of them, from one or more of these topics.

For paper quizzes (Figure 7), the header includes a section for student identification, used to associate the completed exam with the correct student. Answers to the questions can be presented along with the questions themselves or on separate answer sheets. Exams can span multiple pages, provided that each exam contains a unique identifier for the student. It is recommended that students fill in their answers completely, rather than using checkmarks, to ensure better accuracy during the scanning process.

For digital quizzes (Figure 8), each student receives a unique exam. To manage the distribution of online exams generated by AMC, a Python script is used to separate individual student forms from the single PDF file provided by AMC. The system generates a single PDF containing all student exams in sequence, but to deliver each student their specific form, the document must be split into individual files while preserving interactive fillable fields. For each student, a new PDF is created with only their respective pages, named according to their identification data, and then uploaded for distribution.

The system can also be used to create comprehensive exams that combine new objective questions, new open-ended and essay questions, and a selection of questions reused from previous quizzes. Reusing quiz questions may be appropriate when quiz results indicate that most students struggled with certain material. In such cases, the instructor may choose to revise the content and then include those same questions in the exam to retest student understanding.

In MC732, quiz grades are used to allow students to skip certain exam questions. For instance, some questions on the exam may include an instruction stating: “If you received a grade of 7 or higher on quiz (number), you may skip this question (or the questions on this section or page) and copy the quiz grade into the question’s grade.” This approach rewards students for strong performance on quizzes, reducing their workload on the final exam.

The results of the quiz can also inform the selection of questions for the exam. If a quiz question was answered correctly by most students, it might be excluded from the exam to focus on areas where students demonstrated weaker understanding. This allows the exam to better target knowledge gaps and ensure that the assessment is meaningful.

Typically, an exam in MC732 consists of a mix of question types. It includes general objective questions that were not covered in the quizzes, which all students must answer. There is also a set of objective or open questions that students answer based on their quiz performance, as well as a set of open-ended questions that all students are required to complete.

Open-ended questions in the exams are designed to assess higher levels of understanding. These questions might involve

+1/1/60+

ID

0 0 0 0 0 0
1 1 1 1 1 1
2 2 2 2 2 2
3 3 3 3 3 3
4 4 4 4 4 4
5 5 5 5 5 5
6 6 6 6 6 6
7 7 7 7 7 7
8 8 8 8 8 8
9 9 9 9 9 9

Undergraduate Computer Architecture

Quiz Example

Instructions: Fill in your student ID and sign below. All answers must be provided on this page; answers outside the form will be ignored. Questions with ♣ may have zero, one, or more correct answers. The use of calculators and consultation materials is not allowed. This is an individual quiz.

Signature :

.....

Question 1 If a program spends 60% of its time executing multiplication instructions, how much faster do the multiplication instructions need to be for the program to run 2 times faster?

- [A] It is not possible to determine the answer with the given information.
- [B] It is not possible to achieve the desired performance by optimizing only the multiplication instructions.
- [C] 3x faster.
- [D] 12x faster.
- [E] 6x faster.

Question 2 How many bits are needed to address a register in the RISC-V register file?

[16] [8] [6] [5] [32]

Question 3 Convert the number 0xc0980000 from single-precision floating-point representation to decimal.

	0 0	0 0		
	1 1	1 1		
	2 2	2 2		
	3 3	3 3		
	4 4	4 4		
	5 5	5 5		
	6 6	6 6		
	7 7	7 7		
+	8 8	8 8		
−	9 9	9 9	·	

Question 4 Consider an architecture with 32-bit words and a direct-mapped L1 data cache with 512 bytes capacity. If the cache has blocks of 4 words, how many bits are required for the index field in the address division?

1 2 3 4 5 6 7 8 9

For the next question, considering a multi-cycle divider, divide 13 by 5, indicating the value of the specified register at the end of the clock cycle. Assume that the inputs and results are represented with 4 bits, and give your answer in binary.

Question 5

Remainder, Cycle 3

0 0 0 0
1 1 1 1

Question 6 Implement the `int strlen(const char* str)` function in RISC-V assembly, following all necessary ABI conventions and without using pseudo-instructions. The function should return the length of the string.

0 1 2 3 4 5

.....

.....

.....

.....

.....

Fig. 7. Sample quiz. Questions are typeset from the examples in Section IV. The source code for each question is presented in the respective Figure number. Each quiz has a unique identifier, so they can span multiple pages and students only need to identify themselves on the first page. ID code format may be adapted as necessary.

+2/1/59+

Undergraduate Computer Architecture – Online Quiz Example

Instructions: Locate the file with your name and student ID. Upload the PDF with the filled-out answers in the form. Answers outside the form will be ignored. This is an individual quiz.

Name:	ID:
Firstname Lastname	**123456**

Consider a half-precision floating-point representation using 16 bits with the following characteristics:

- Sign Bit (Bit 15)
- 5-bit Exponent (Bits 14–10)
- 10-bit Fractional Part (Bits 9–0)
- Exponent bias equal to 15

This representation follows all the conventions of the IEEE 754 floating-point representations, including normalization, value conversion, representation of infinity and NaN, denormal numbers, etc.

Question 1 How is the number 26.25 represented in half-precision floating-point notation?

0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1

Fig. 8. Digital quiz for online submission. Each student receives a pre-identified form, completes the quiz by selecting checkboxes in the PDF, and submits the finished file through an online platform such as Google Classroom.

TABLE I
AVERAGE SCORES AND SUCCESSFUL RESULT PERCENTAGES FOR TOPICS IN THE COURSE OVER DIFFERENT SEMESTERS.

Topic / Semester	Average Grade					Successful Result				
	22.1	22.2	23.2	24.2	Mean	22.1	22.2	23.2	24.2	Mean
Performance and CPI Calculation	6.7	5.7	6.5	5.7	6.2	57%	67%	56%	62%	61%
ISA and Code Generation	5.3	6.2	4.5	-	5.3	38%	70%	21%	-	43%
Application Binary Interface	6.2	6.8	6.4	6.8	6.6	57%	73%	56%	66%	63%
Arithmetic for Computers	5.3	-	8.4	7.8	7.2	40%	-	84%	95%	73%
Datapath: Control and Data Signals	4.1	4.2	7.4	6	5.4	36%	47%	71%	66%	55%
Mean for Exam 1 Quizzes	5.5	5.7	6.6	6.6	6.1	46%	64%	58%	72%	60%
Exam 1	5.6	6.3	6.5	6	6.1	54%	75%	82%	84%	74%
Pipeline: Hazards and Forwarding	5.7	4.8	4.5	3.6	4.7	47%	52%	40%	33%	43%
Pipeline: Control and Data Signals	5.9	3.3	5.6	5.2	5.0	50%	36%	44%	70%	50%
Cache: Direct Mapped Organization	6	6.4	7.3	6.2	6.5	63%	69%	74%	83%	72%
Cache: Misses and Performance	6.8	3.1	7.6	5	5.6	65%	36%	73%	73%	62%
Virtual Memory	-	-	-	5.3	5.3	-	-	-	71%	71%
Mean for Exam 2 Quizzes	6.1	4.4	6.3	5.1	5.5	54%	48%	58%	66%	57%
Exam 2	5.6	5.1	6.2	6.5	5.9	68%	67%	85%	86%	77%

evaluating or creating new solutions or modifications to an architecture implementation. For example, students might be asked to modify a datapath to support a new type of instruction. These questions are difficult to automatically grade and require careful analysis by the instructor. However, they are crucial for evaluating a deeper understanding of the course material and the ability to apply concepts in novel situations.

V. Learning Assessment

AMC has been used in varying degrees in MC732 since 2019. This section briefly summarizes the learning assessment results for the most recent editions of the course. Table I presents an overview of the average scores and successful results for various topics covered in the computer architecture course over different semesters. The data spans the four semesters since the return to in-person classes after the COVID-19 pandemic, during which quizzes with automated grading were applied.

Overall, the average scores show variability between topics and semesters, with some notable trends. For example, the topic Performance and CPI Calculation" saw a slight decline from 6.7 to 5.7, while "Application Binary Interface" maintained relatively stable performance around 6.8. The successful results, which indicate the percentage of students achieving a satisfactory grade (7/10 for tests and 5/10 for exams), generally

TABLE II
PERCENTAGE OF STUDENTS USING QUIZ GRADES ON EXAMS (S), AND MEAN PORTION OF THE EXAM GRADE ATTRIBUTED TO QUIZZES (G).

Semester	Exam 1		Exam 2	
	S	G	S	G
22.1	71%	31%	90%	58%
22.2	92%	53%	86%	32%
23.1	91%	33%	80%	37%
24.1	73%	41%	73%	33%
Average	82%	40%	82%	40%

align with the average scores. A significant improvement is observed in "Arithmetic for Computers" with successful results increasing from 40% to 95%. Furthermore, the average scores and successful results for Exam 1 and Exam 2 quizzes are indicative of the effectiveness of continuous assessment methods, with a general upward trend in student performance and understanding over time.

Table II shows the percentage of students who used their quiz grades on exams (S) and the mean portion of the exam grade attributed to the quizzes (G) in different semesters. For Exam 1, the highest percentage of students using quiz grades was observed in semester 22.2 with 92%, where quizzes accounted for 53% of the exam grade. Similarly, for Exam 2, semester 22.1 saw the highest percentage at 90%, with quizzes contributing 58% to the exam grade. The average percentage of students using quiz grades on the exams was consistent for both exams at 82%, with an average contribution of 40% of the exam grade attributed to quizzes. This data indicates a significant reliance on quiz grades for overall exam performance.

VI. STUDENT FEEDBACK

Unicamp applies course evaluation forms every semester, in which students numerically assess a series of quantitative criteria and fill out a text field for qualitative evaluation. The average result of quantitative evaluations of the course in semesters when the continuous evaluation system was applied was approximately 95%, compared to an average of 85% for other courses at the Institute of Computing. Qualitative evaluations mentioning tests and continuous assessment can be categorized into positive comments and suggestions (approximately 80%) and negative comments (approximately 20%). The most significant positive comments are summarized below.

Keeping up with the course. Many students appreciated the quizzes, feeling that they helped them stay up-to-date with the material. The continuous assessment model allowed students to follow the course topics gradually and provided a clear view of their assimilation of the content.

Assessment Format and Motivation The quizzes were seen as a helpful way to maintain engagement with the material and ensure class attendance. Students found the quiz model to be motivating, even those who typically do not enjoy attending classes. The course was praised for requiring continuous attendance and maintaining consistency between quiz content and class material, with suggestions that this method could be applied to other courses.

Quick Grading. The quick correction of quizzes was seen as a valuable aspect of the course structure.

The most frequent suggestions included:

Conducting quizzes during the first period. Students suggested holding quizzes on days when the course is taught in the first period, particularly for evening courses, as they felt more rested and prepared at that time.

More quizzes and greater valuation. There were requests for more quizzes and exercises for home practice, formatted similarly to the quizzes. Additionally, some students suggested increasing the weight of quizzes in the final grade calculation.

Finally, the vast majority of negative comments mentioned the rigidity of the correction system. Some students felt that the quiz correction was too binary and did not consider partial knowledge or reasoning. They expressed concerns that the evaluation method focused solely on the final result rather than the student's thought process. Suggestions were made to include partial credit for partially correct answers, as the binary approach was seen as too rigid.

Note that while multiple choice questions are naturally binary in evaluation, quantitative questions can include tolerances in results and partial credit for answers close to the correct one. However, re-visiting student responses to give partial credit for correct reasoning is challenging for a large-scale class.

VII. RELATED WORKS

Continuous assessment (CA) requires significant effort from the instructor to implement effectively [11], making the use of computer-assisted assessment tools essential to manage this workload, particularly in large-scale settings [2].

The use of AMC, particularly for quizzes to support continuous assessment, has been described in various higher learning settings. Sauerwein et al. [12] investigated the use of AMC software to facilitate continuous assessment in university-level physics courses. By using AMC, the authors were able to streamline the correction process, reduce the time required for feedback, and ultimately make continuous assessment more feasible in large classroom settings.

Clancy and Quinn [3] highlight the effectiveness of AMC in reducing the response time for providing feedback in Physics courses. The authors also explore the impact of automated feedback on student motivation and the ability to predict exam performance. The study found that the students often performed better on the final exam than the quizzes, suggesting that the quizzes may have motivated some students to improve their understanding and performance.

Santos et al. [5] describe how Matlab scripts can be used to generate multiple versions of exams, similarly to the strategy using Python presented in this paper. The exams are generated and processed by AMC, allowing for fast and detailed feedback to students, and allowed for statistical analyses of student performance with respect to specific topics. Similarly, Hamada

et al. [6] describe the integration of AMC with Computer Algebra Systems using the Lua language.

In the context of a Computer Architecture Laboratory class, Llamas-Nistal et al. [7] introduced short exercises at the beginning of lab sessions to encourage continuous engagement and self-assessment, with these exercises contributing to a small component of the final grade. The authors found that short exercises positively impact student learning outcomes by encouraging continuous participation.

VIII. Conclusion

The implementation of continuous assessment (CA) through frequent quizzes in the Design of Computing Systems course has been effective in enhancing student engagement and understanding. Automated grading allows students to answer a very large number of questions throughout the semester, a task that is unmanageable without such tools. This approach rewards continuous engagement, reduces the exam burden for consistently performing students, and allows targeted adjustments in teaching based on student performance. Automated grading also ensures objective, fair, and error-free assessments.

However, this approach shifts the workload from grading to question creation, which requires significant effort, especially to create effective multiple choice questions and technically challenging numerical questions. Although beneficial for large classes, the approach may not be worthwhile for smaller groups. Another issue is that the distribution of graded solutions eases the creation of previous quiz and exam banks by the students, which requires regular updates to the question pool. Finally, students have expressed concerns about rigid correction methods that only recognize exactly correct responses, potentially overlooking partial understanding. Therefore, the best results are achieved when the quizzes are supplemented with open questions that can test higher-level competencies.

In general, CA supported by automated grading has created a more efficient and adaptive learning environment in the course. The code for the examples featured in this paper is available at https://github.com/lfwanner/caca.

Acknowledgments

This work was supported by the National Council for Scientific and Technological Development (CNPq) grants 402467/2021-3 and 405940/2022-0, Coordination for the Improvement of Higher Education Personnel (CAPES) grant 88887.954253/2024-00, and Unicamp FAEPEX grant 2477/23.

References

[1] Thomas A. Angelo and K. Patricia Cross. *Classroom Assessment Techniques: A Handbook for College Teachers*. Jossey-Bass, San Francisco, CA, 2nd edition, 1993.

[2] John Bull and Colleen McKenna. *A Blueprint for Computer-Assisted Assessment*. RoutledgeFalmer, London, UK, 2001.

[3] Ian Clancy and Ann Marcus-Quinn. Exploring the possibilities of automated feedback for third level students. *Form@re*, 19(3):247–256, Dec. 2019.

[4] Alexandre Courbot. Auto multiple choice. https://www.auto-multiple-choice.net, 2024.

[5] Milana Lima dos Santos, Hernán Prieto Schmidt, Giovanni Manassero, and Eduardo Lorenzetti Pellini. Automated design for engineering student examinations using matlab/octave scripts and the auto multiple choice package. In *2019 IEEE World Conference on Engineering Education (EDUNINE)*, pages 1–5, 2019.

[6] Tatsuyoshi Hamada, Yoshiyuki Nakagawa, and Makoto Tamura. Method to create multiple choice exercises for computer algebra system. In *International Conference on Mathematical Software*, volume 12097 of *Lecture Notes in Computer Science*, pages 419–425. Springer, 2020.

[7] Martín Llamas-Nistal, Martín Liz-Domínguez, Juan M. Santos-Gago, Manuel J. Fernández-Iglesias, Luis E. Anido-Rifón, and Moisés R. Pacheco-Lorenzo. Short-exercise assessment in a computer architecture laboratory. In *XVI Congreso de Tecnología, Aprendizaje y Enseñanza de la Electrónica (TAEE)*, pages 1–5, 2024.

[8] Michael Mehlich. Fixed point package for latex. https://ctan.org/pkg/fp, 2021.

[9] David A. Patterson and John L. Hennessy. *Computer Organization and Design RISC-V Edition: The Hardware/Software Interface*. Morgan Kaufmann, Cambridge, MA, 2nd edition, 2020.

[10] Geoffrey M. Poore. *PythonTeX: Run Python from within LaTeX*, 2023.

[11] Jose-Luis Poza-Lujan, Carlos T. Calafate, Juan-Luis Posadas-Yagüe, and Juan-Carlos Cano. Assessing the impact of continuous evaluation strategies: Tradeoff between student performance and instructor effort. *IEEE Transactions on Education*, 59(1):17–23, 2016.

[12] Ricardo Andreas Sauerwein, Josemar Alves, and Dioni Paulo Pastorio. Uso de um software de correção automática: uma alternativa para viabilizar o processo avaliativo contínuo no contexto do ensino universitário. *Informática na Educação: Teoria & Prática*, 21(3), dez. 2018.