THE IMPACT OF CHANGING HOMEWORK FREQUENCY IN A COMPUTER ARCHITECTURE COURSE [*]

Micah Taylor
Department of Computer Science
Rose-Hulman Institute of Technology
Terre Haute, IN 47803
812-877-8396
taylormt@rose-hulman.edu

Sid Stamm
Department of Computer Science
Rose-Hulman Institute of Technology
Terre Haute, IN 47803
812-877-8364
stammsl@rose-hulman.edu

Christine Taylor
Department of Mathematics and Computer Science
Indiana State University
Terre Haute, IN 47809
812-237-9050
christine.taylor@indstate.edu

ABSTRACT

Formative assessment in a computer architecture course is often modeled as lab exercises and written homework. In this paper, we discuss the impact of splitting large homework assignments into daily assignments for a second–year computer architecture course. We describe our motivation for the adjustment, the design goals for the change, and preliminary results showing the effect on student learning. In evaluating the impact, we analyze student homework and test scores, as well as informal and formal course evaluation surveys.

[*] Copyright © 2018 by the Consortium for Computing Sciences in Colleges. Permission to copy without fee all or part of this material is granted provided that the copies are not made or distributed for direct commercial advantage, the CCSC copyright notice and the title of the publication and its date appear, and notice is given that copying is by permission of the Consortium for Computing Sciences in Colleges. To copy otherwise, or to republish, requires a fee and/or specific permission.

INTRODUCTION & BACKGROUND

Computer architecture courses provide the foundation for understanding the hardware on which almost all computer programs run. The fundamental concepts of the hardware/software interface inform many design and performance decisions at higher levels of abstraction.

For many years, tools have been used to aid in teaching computer architecture to students [11]. The MARS simulator[10], SPIM[5], SPIMBot[12], HASE [1], and other tools have provided excellent opportunities for computer architecture students to practice hands-on development. Despite these tools, student learning may not be at the expected level upon completing a computer architecture course [6]. In this paper, we focus on improving student learning through changes in the homework assessment tool used by our computer architecture course.

Studies have shown that homework can be an effective tool if used correctly [2]. Homework is a common formative assessment tool [7], primarily designed to help the student understand material instead of assessing the student's capabilities. Formative assessment can vary in many ways: intended difficulty, frequency, designed length, type of feedback, and more [3]. Reoccurring low-risk formative assessment tools offer opportunities to improve student performance on exams [4] and there appears to be a positive correlation between homework frequency and improved student performance [9]. Though homework strategies are often studied, there does not appear to be research in the use of homework in computer architecture courses.

OVERVIEW

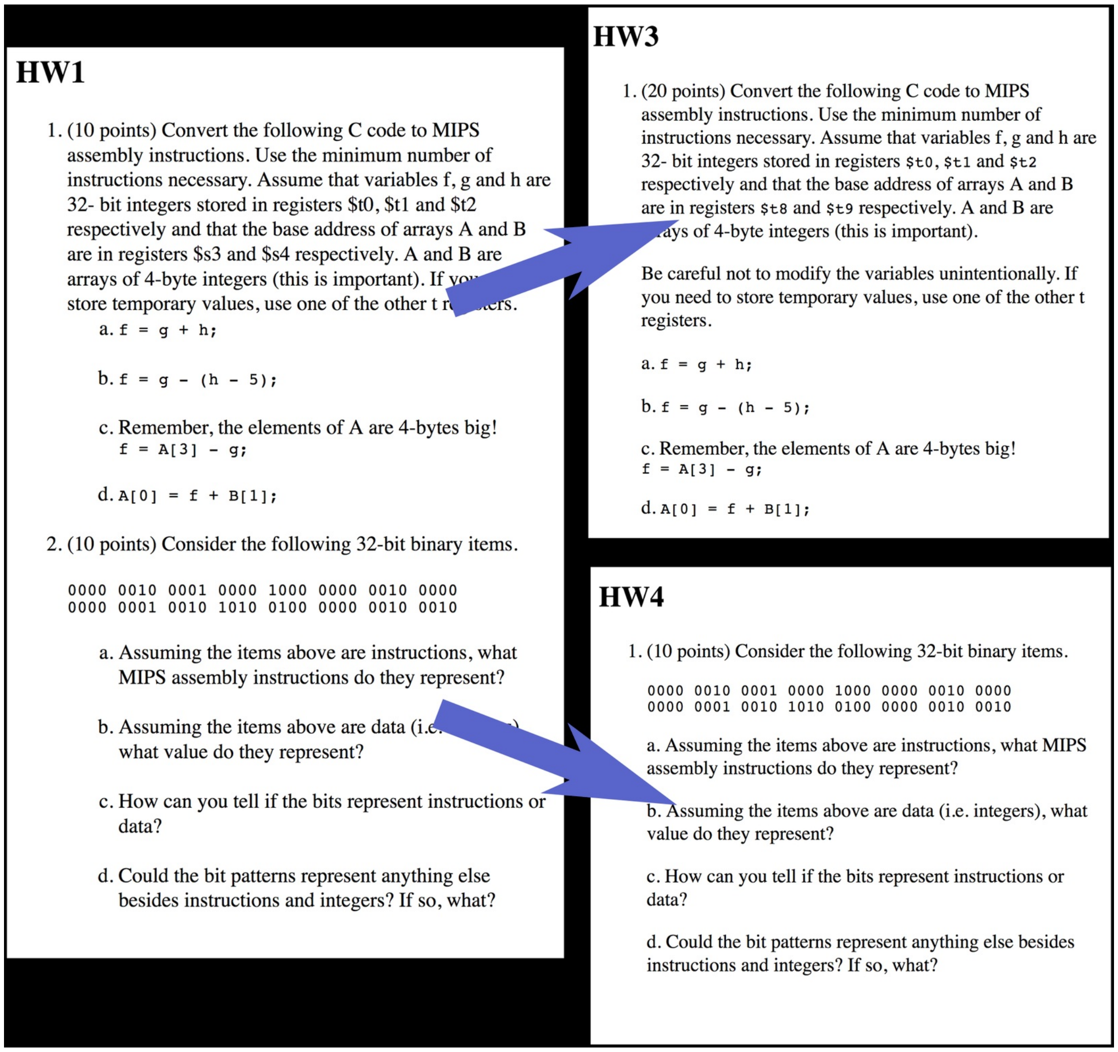
Our computer architecture course is taught to second–year Computer Science and Software Engineering students and to third year Computer Engineering students. The course uses four main assessment tools: daily quizzes/worksheets, homework assignments, coding and design laboratory sessions, and written exams. The course meets five times a week for a total of six in–class hours. At the beginning of the term, two of the sessions are used for coding labs. The other three sessions are used as lecture and active learning exercises to cover new material. Near midterm, the students begin a large processor design project. At this point, the frequency of labs and homework decreases and session time is spent on project review.

Homework assignments are an integral part of education at our institute and students are expected to spend a fair amount of time outside of class working practice exercises. In this course, homework assignments were initially distributed as large weekly or semi-weekly assignments. The assignments were targeted at about 2-5 hours of work for an average student. The material in each assignment was designed to follow the material in class over the week, allowing students to incrementally complete the homework.

Motivation

Our primary motivation for making change to the course was student course evaluation comments. Student feedback often mentioned delays in getting homework returned, homework difficulty, and frequency or size of the homework. Example comments from course evaluations:

- *"The homework really needed to get back sooner."*
- *"I think having more, shorter homework assignments would help it provided faster turnaround on homeworks"*
- *"The homework assignments are too large and don't seem to always be related"*
- *"The homework is far too hard."*
- *"I would suggest homeworks for this class be given daily ... (this would significantly increase the workload, but having those daily assignments would've likely given me the practice I needed to completely learn the material)."*



HW1

1. (10 points) Convert the following C code to MIPS assembly instructions. Use the minimum number of instructions necessary. Assume that variables f, g and h are 32- bit integers stored in registers $t0, $t1 and $t2 respectively and that the base address of arrays A and B are in registers $s3 and $s4 respectively. A and B are arrays of 4-byte integers (this is important). If yo store temporary values, use one of the other t r ters.

    a. `f = g + h;`

    b. `f = g - (h - 5);`

    c. Remember, the elements of A are 4-bytes big!
    `f = A[3] - g;`

    d. `A[0] = f + B[1];`

2. (10 points) Consider the following 32-bit binary items.

```
0000 0010 0001 0000 1000 0000 0010 0000
0000 0001 0010 1010 0100 0000 0010 0010
```

    a. Assuming the items above are instructions, what MIPS assembly instructions do they represent?

    b. Assuming the items above are data (i.e)
    what value do they represent?

    c. How can you tell if the bits represent instructions or data?

    d. Could the bit patterns represent anything else besides instructions and integers? If so, what?

HW3

1. (20 points) Convert the following C code to MIPS assembly instructions. Use the minimum number of instructions necessary. Assume that variables f, g and h are 32- bit integers stored in registers `$t0`, `$t1` and `$t2` respectively and that the base address of arrays A and B are in registers `$t8` and `$t9` respectively. A and B are ays of 4-byte integers (this is important).

    Be careful not to modify the variables unintentionally. If you need to store temporary values, use one of the other t registers.

    a. `f = g + h;`

    b. `f = g - (h - 5);`

    c. Remember, the elements of A are 4-bytes big!
    `f = A[3] - g;`

    d. `A[0] = f + B[1];`

HW4

1. (10 points) Consider the following 32-bit binary items.

```
0000 0010 0001 0000 1000 0000 0010 0000
0000 0001 0010 1010 0100 0000 0010 0010
```

    a. Assuming the items above are instructions, what MIPS assembly instructions do they represent?

    b. Assuming the items above are data (i.e. integers), what value do they represent?

    c. How can you tell if the bits represent instructions or data?

    d. Could the bit patterns represent anything else besides instructions and integers? If so, what?

Figure 1: An assignment before and after a split.

Goals

The original homework assignments were quite large and were likely difficult for the students to complete in one sitting. Since this class is taken by second–year students, it is possible that some of them overestimate their abilities and delay beginning the homework until just before the due date. Since the large homework assignments required many hours to grade, the graded copies were returned quite some time after submission.

This combination of procrastination, large (possibly intimidating) assignment size, and long delays in feedback led us to hypothesize that smaller, more quickly reviewed assignments would enhance the students' learning.

To test our hypothesis, we modified the course materials to replace the larger homework assignments with smaller, more frequent ones.

Design

Each original homework assignment was split into multiple smaller assignments (Figure 1). Existing content was split into sizes that we estimated could be done in one hour or less each day. The content in the homework remained the same except for two assignments. One assignment had a long continuous problem that could not be split, so a new problem was designed. The other assignment was redesigned to better match in–class activities.

We have used this new format for two consecutive offerings of the class. In the first offering, we had a very aggressive schedule that included one assignment each day, due the following day. Students quickly offered feedback that there was not enough time to seek help for the assignments, and we adjusted the schedule to give an extra day between giving the assignment and the due date.

DATA & METHODOLOGY

We analyzed both quantitative and qualitative data. The quantitative data consisted of exam scores from twelve sections across four terms: this range included two terms before and two terms after the change. We computed and compared mean exam scores by section.

We also analyzed two types of qualitative data: informal, mid–term, plus/delta surveys and end-of-term course evaluations. We used a set of plus/delta data and evaluations from the first time the new homework was offered, in which we specifically solicited homework feedback. We did not explicitly solicit this feedback in any other survey. We also had another set of plus/delta data from a second time the new homework was offered.

To analyze qualitative student comments, we rely on a coding based approach [8]. Our a priori codes, codebook, and definitions are shown in Figure 2.

Two coders evaluated student comments. One coder developed an initial codebook and then we compared coders on a selection of 57 student responses. The two coders found 17 possible items to code out of the 57 responses. The coders disagreed about one code and missed one code each. We made minor adjustments to the codebook after comparing coder results and then coded the remaining data.

	Fall 1617 Original HW	Winter 1617 Original HW	Fall 1718 Daily HW	Winter 1718 Daily HW
Section 1	82.2	82.0	83.3	86.3
Section 2	*	79.0	84.2	82.1
Section 3	82.6	78.4	80.9	80.0
All Sections	82.4	79.8	82.9	83.1

Table 1: Mean exam scores and homework frequency.

**The gradebook for sections 1 and 2 in Fall 1617 was combined, so there is no entry for section 2 in the table*

| *Definition:* For all codes, homework should be considered an out-of-class assignment that is not one of the other assessment tools (quiz, test, project). Additionally, for an assignment to be considered homework, it must have be assigned to the student on one date and then later due for submission on another date.

- Schedule of homework
  - Schedule of homework is related to homework scheduling or frequency. The homework schedule is defined by assigned dates and due dates. How often homework is assigned and due is the frequency of the homework.
  - *Inclusion:* This code should be used if the data indicates a concern with how often the homework is assigned or due. It should also be used if there is a concern with the length between the assigned date and the due date.
  - *Exclusion*: The code should not be used if there are concerns with about specific calendar scheduling events which are beyond the control of the instructor. The code should not be used if the data indicates issues with individual time management or scheduling. Specific comments about difficulty should not be marked with this code.
- Homework feedback schedule
  - *Homework feedback schedule* is related to the graded homework that is returned to the students. The total return time is the the time from when the homework is due and when it is returned.
  - *Inclusion:* This code should be used if the data indicates there was a concern with the length of time before the homework was returned. It should be used if the data indicates a the student was affected by the return time of the homework or the homework not being returned in time for some event (i.e. an exam).

- *Exclusion:* This should not be used for data concerning quality of the feedback on homework. It should not be used for comments that do not involve return time (i.e. frequency or scheduling).

- Homework difficulty
  - *Homework difficulty* is how hard the homework was to complete. The problems themselves may be too easy or too hard, or there may be too much or not enough time to complete the assignment.
  - *Inclusion*: This code should be used if the data indicates any concerns with difficulty intrinsic to the homework. This includes ease or hardship from scheduling, size, length, or content.
  - *Exclusion:* Difficulties specifically related to scheduling should not be indicated by this code (i.e. specific dates on which due dates fall). Difficulties related to return dates should also not be included.

Figure 2: Our codebook

RESULTS

As noted in Section 3.3, during the first offering, we adjusted the schedule mid-term to allow students more time to seek help for difficult problems. The difficulty with the initial aggressive schedule and the result of the change is evident in the data.

We first present our quantitative data based on exam grades, then qualitative results and themes based on student comments.

Quantitative results

The content of the homework exercises is intended to expose students to concepts and topics they will later be tested on through exams. If the exercises and exam assess the same learning outcomes, it follows that students who practice (and receive effective feedback) are likely to perform better on the exams in this course.

To test whether or not the modified homework schedule improved student performance on exams, we examined the mean exam grades before and after the homework was adjusted to small-size, high-frequency exercises. Table 1 shows a small increase in average exam scores with the smaller and more frequent homework. This suggests that on average students saw a marginal (but positive) average benefit from the more frequent assignments.

Code counts

Figure 3 shows the volume of coded responses increases with the change, but drop as we adjust to student concerns about making assignments available earlier and stabilizing the due dates. Figure 3 also shows the sentiment of the responses becomes more positive after the change is adopted, suggesting they are more likely to embrace the assignments as a benefit and not a barrier.

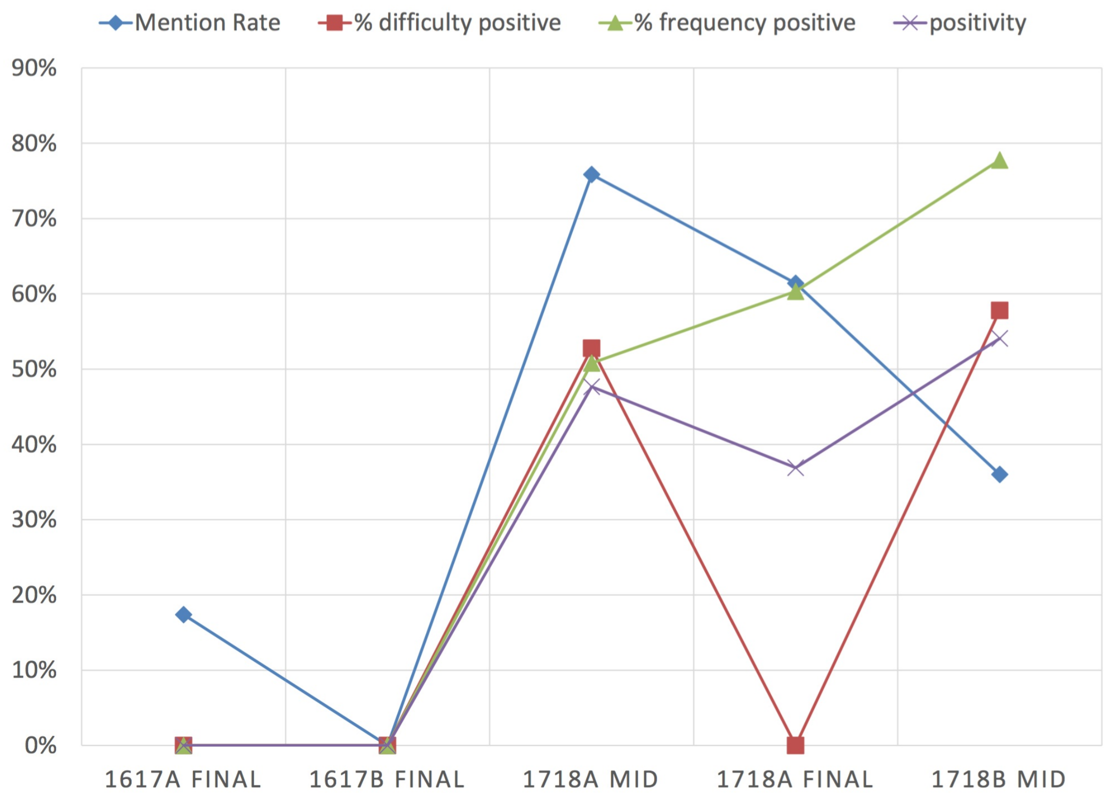


Figure 3: Graph of positive responses. Mention rate is the percent of responses with a coded element. Positivity is the percent of responses that are favorable. The other items show a category's percent of positive responses.

Theming

After coding, we analyzed the comments to identify themes. These themes give us a sense of the strengths and weaknesses of this approach.

Daily, manageable learning: We observed mostly positive comments about the schedule of homework. Students noted that the daily schedule forced them to review recent material and not procrastinate the work.

- *"I felt having daily homework was essential to my learning process in the course, as weekly assignments are generally pushed off until the last minute minimizing learning."*
- *"I personally liked the daily homeworks, I thought it spread out the work to make it more managable"*

Slow it down: Our initial schedule of 'homework assigned, then due the next day' was highly criticized. After the change to 'assigned, then due in two days', these complaints disappeared.

- *"I really liked the homework as it reinforced the concepts from class. However, the homwork would be a lot better if it was given every two days, instead of every day."*
- *"The homework was not great everyday. I liked it when it was changed."*

Return it sooner: Even before the change, students wanted faster feedback on homework. We hoped that the smaller assignments would improve turn around time, but we still received many negative comments about return time after the change.

- *"Get the homeworks back faster so we can learn from them, expecially before tests."*
- *"It would be better if we got quizzes and homework back faster, so that we can see our mistakes and also so we can have references to look back on."*

ANALYSIS

After making the change to the homework schedule, we analyzed the results and observed some advantages and disadvantages. We also discuss future work.

Advantages

Following the change, exam grades are slightly improved, however the improvement is marginal. We will need to collect more data and correct for individual student differences in order to comprehensively analyze the result.

Students are offering much more feedback on homework, both positive and negative, after the change. Students repeatedly noted that the daily frequency was helpful in their studies. Better analysis of exam scores, as described above, will help verify this qualitative result.

Disadvantages

The daily assignment schedule also results in more administrative work for the instructor. For example, coordinating collection, grading, and return of homework becomes a daily task. This could reduce time in class for other more important activities. These costs could be mitigated by increasing load in other areas: i.e. having campus mail deliver the graded assignments to the students or hiring teaching assistants to assist in grading.

Students also noted that homework was not returned fast enough. Since there are more homework assignments to return, there is more opportunity for issues to delay the feedback. The administrative costs described above could easily delay the return time.

CONCLUSION

We observed some positive results from changing from weekly/semi-weekly to daily homework assignments. The qualitative data indicates that the frequency is a mostly positive change, but that delays in feedback must be mitigated. The quantitative data shows slight improvement, but is not robust enough to be conclusive. Longer-term studies over several years would be helpful for analyzing the trends in qualitative data, as well as providing the opportunity for more detailed statistical analysis of the data.

REFERENCES

[1] Paul S Coe, Fred W Howell, Roland N Ibbett, and Laurence M Williams. A hierarchical computer architecture design and simulation environment. *ACM Transactions on Modeling and Computer Simulation (TOMACS)*, 8(4):431–446, 1998.

[2] Swantje Dettmers, Ulrich Trautwein, Oliver Lüdtke, Mareike Kunter, and Jürgen Baumert. Homework works if homework quality is high: Using multilevel modeling to predict the development of achievement in mathematics. *Journal of Educational Psychology*, 102(2):467, 2010.

[3] Wynne Harlen and Ruth Deakin Crick. Testing and motivation for learning. *Assessment in Education: Principles, Policy & Practice*, 10(2):169–207, 2003.

[4] Beverly M Klecker. The impact of formative feedback on student learning in an online classroom. *Journal of Instructional Psychology*, 34(3):161–166, 2007.

[5] James Larus. Spim: A mips32 simulator. http://spimsimulator.sourceforge.net, 2017.

[6] Leo Porter, Saturnino Garcia, Hung-Wei Tseng, and Daniel Zingaro. Evaluating student understanding of core concepts in computer architecture. In *Proceedings of the 18th ACM Conference on Innovation and Technology in Computer Science Education*, ITiCSE '13, pages 279–284, New York, NY, USA, 2013. ACM.

[7] D Royce Sadler. Formative assessment and the design of instructional systems. *Instructional science*, 18(2):119–144, 1989.

[8] Johnny Saldan˜a. *The coding manual for qualitative researchers*. Sage, 2015.

[9] Ulrich Trautwein. The homework–achievement relation reconsidered: Differentiating homework time, homework frequency, and homework effort. *Learning and Instruction*, 17(3):372–388, 2007.

[10] Kenneth Vollmar and Pete Sanderson. Mars: An education-oriented mips assembly language simulator. In *Proceedings of the 37th SIGCSE Technical Symposium on Computer Science Education*, SIGCSE '06, pages 239–243, New York, NY, USA, 2006.

[11] Gregory S. Wolffe, William Yurcik, Hugh Osborne, and Mark A. Holliday.

Teaching computer organization/architecture with limited resources using simulators. *SIGCSE Bull.*, 34(1):176–180, February 2002.

[12] Craig Zilles. Spimbot: An engaging, problem-based approach to teaching assembly language programming. In Proceedings of the 2005 Workshop on Computer Architecture Education: Held in Conjunction with the 32nd International Symposium on Computer Architecture, WCAE '05, New York, NY, USA, 2005. ACM