Assembly Academy: Using Video Games and Virtual Robots to Teach Assembly Programming

Kaden Gryphon
Department of Computer Science
University of Alabama in Huntsville
Huntsville, Alabama, United States
kb0125@uah.edu

Haeyong Chung
Department of Computer Science
University of Alabama in Huntsville
Huntsville, Alabama, United States
hc0021@uah.edu

2023 IEEE International Conference on Advanced Learning Technologies (ICALT) | 979-8-3503-0054-3/23/$31.00 ©2023 IEEE | DOI: 10.1109/ICALT58122.2023.00037

***Abstract*—Assembly is underutilized as a beginner programming language. Its simple and repetitive syntax allows students to focus on logic and problem-solving. While the low level of abstraction causes Assembly to be challenging to work with, using Assembly gives students better insights into how computers operate and more fundamental skills for procedural programming. This paper introduces Assembly Academy, a programming puzzle game utilizing a virtual robot. We include a use case demonstrating how the game provides feedback to students to keep them engaged and motivated while learning Assembly.**

***Index Terms*—assembly programming, learning programming, programming game, virtual robot**

I. Introduction

Computer programming has increasingly become a core part of education, and as such, there have been many efforts to improve methods of teaching programming [1], [2]. While a multitude of tools and games are available for visual and scripting languages, there are few for assembly programming. Assembly language is usually considered more challenging to learn as it is harder to visualize its lower abstraction level than high-level programming languages like Python or JavaScript. Assembly programming is important for students to learn despite these challenges. Assembly's simple syntax can make for a good beginner language as the students can focus on solving logic problems instead of fixing syntax errors. It gives greater insight on how a PC operates and executes commands and gives students the problem-solving skills to write high-performance code for embedded systems [3].

In this paper, we introduce Assembly Academy, a game that walks students through learning assembly programming with a virtual robot and progressively more challenging puzzles. Robots can be a powerful tool for learning programming. Using a robot makes programming more tangible, as the student's code is executed in a real-world environment. This makes learning more engaging as the student can directly see the effects of their code [4]. We believe a virtual robot has these same advantages but is more accessible. Code can be edited and executed more quickly in a virtual environment where the student does not have to reset the robot by hand. This creates less downtime between testing each code iteration, helping keep students in a flow state. Assembly Academy is designed to provide a fun and engaging way to learn programming and develop fundamental skills using a robot. We aim to keep students focused and motivated and negate the challenges of visualizing assembly code. Our use case shows how our game accomplishes this by providing feedback to the student to help walk them through using their newly learned skills to complete programming puzzles.

This work was partially supported by National Science Foundation Grant HCC-2146523.

II. Related Work

Assembly Academy's main design goal is to give students a virtual robot to help visualize their code, with game-like puzzles to keep them focused and motivated.

Professional programming environments and high-level languages can be intimidating for new students. Mobile robots were tested against traditional methods as a means to address this [4]. Students felt that using robots made the course more enjoyable and less difficult, as a result, they were absent less often and had higher programming self-concept. We designed Assembly Academy's user interfaces to clearly show the program state and provide constructive error messages to reinforce this idea. Thus, students can focus on problem-solving without getting overwhelmed by a complex environment.

Multiple studies have linked using robots or games in teaching to increased student motivation and programming self-concept [1], [2], [5]. This includes SPIMbot, a simulator that lets students write MIPS assembly to control virtual robots [6]. However, SPIMbot lacks the constructive feedback and guidance that Assembly Academy aims to provide.

Combéfis et al. [1] suggest that games need feedback to help motivate students to improve, and that guidance helps keep the students from feeling confused and demotivated. They also say that using scoring or contests motivates self-improvement through competition. Assembly Academy also gives students scores on their code. Using assembly means that the scoring and efficiency can be more precisely measured, encouraging students to be more competitive.

The general consensus on beginner programming education is that the focus should be on teaching and strengthening problem-solving and logical thinking over the syntax of any one programming language [2], [4], [5]. This is the goal of using assembly as the target language. It has a simple and

2161-377X/23/$31.00 ©2023 IEEE
DOI 10.1109/ICALT58122.2023.00037

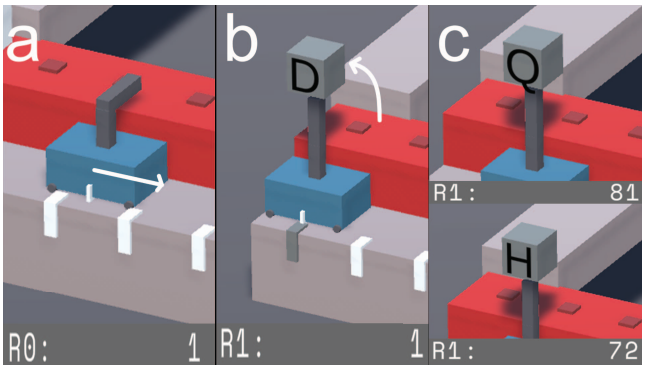

Fig. 1. The three robot commands, (a)botmove: if R0 is positive the robot moves right, else it moves left. (b)botgrab: the robot picks up a block and sets R1 to 1, or puts down a block and set R1 to 2. (c)botlook: the robots sets R1 to the ASCII value of the held letter block.

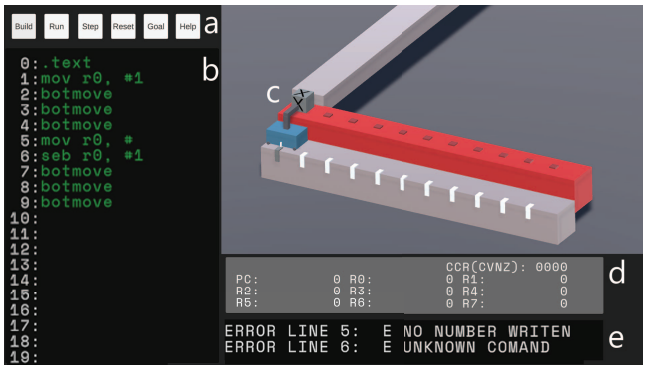

Fig. 2. Assembly Academy's user interface during one of the levels.

repetitive syntax that can let students focus on logic and problem-solving.

III. Design of Assembly Academy

Assembly Academy is a game that walks the student through the basics of assembly programming by having the student program a virtual robot. The game uses commands and syntax based on the ARM assembly language. The robot is based on a final programming assignment of a beginner programming class where the student has to write code for a virtual robot that can move left and right, carry blocks, and compare the values of blocks. The game is implemented in the Unity game engine using the C# programming language.

A. General Structure and Mechanics

Assembly Academy consists of short tutorials that explain a new ARM command, followed by puzzles that have the student use the newly learned commands to program the robot. The game contains a compiler that converts the student's code into ARM machine code, reporting any syntax errors it finds and their line numbers. An emulator then runs the machine code and sends commands to the virtual robot.

Each level gives the player an increasingly difficult puzzle to solve. Most puzzles have a dynamic element that is randomized each run to encourage the student to make a general solution to the puzzle. Part of the puzzle is managing the computer's eight registers, R0-R7. Registers act as variables and hold a small amount of data (2 bytes). The first two registers are often reserved for function input and output. This creates a challenge for the programmer who has to juggle this limited resource. An example puzzle is "Move the block to the position given in R2". Every time the student runs their program, the value in register two will be randomized.

Assembly Academy uses a limited instruction set of the ARM language found in the ARM 7TDMI data sheet [7]. There are a few commands that are added in order to control the robot: `botmove`, `botgrab`, `botlook` (Fig. 1).

When the execution of the student's program is complete, the game will evaluate if the program completed the puzzle. If the program fails, the game will ask the student to try again and give a hint. If the program succeeds, the game will show the student's memory and clock cycle usage.

B. User Interface

Fig. 2 shows the user interface for the puzzles. The top left has a row of buttons to control the game (Fig. 2a).

- **Build**: Complies student's program and checks for errors.
- **Run**: Executes the student's program start to finish.
- **Step**: Advance the execution of program by one step.
- **Reset**: Clears the registers memory, resets the robot's position, and randomizes the dynamic parts of the puzzle.
- **Goal**: Toggles a pop up showing the puzzle's goal.
- **Help**: shows list of ARM commands and their syntax.

Below the buttons is the text area where the student writes their code solution to the level's puzzle (Fig. 2b). On The right is the robot view, where the student can watch the virtual robot execute their program (Fig. 2c). Below this is the current contents of the emulator's registers. This includes an interpretation of the currently executed command, registers R0-R7, the program counter (PC), and the configuration and control register (CCR) (Fig. 2d). These get updated in real-time as the emulator executes the student's program. Below this is where error messages are displayed (Fig. 2e).

IV. Use Case

To illustrate the learning process supported by Assembly Academy, it was used by a beginner programmer. The student has already completed the early tutorials and puzzles and has made it to the tutorial on "labels, loops, and branches." First, the student reads through the tutorial. It introduces the idea of labels and their use. Labels mark locations in the code that loops and functions can jump to. The tutorial explains the syntax for labels and gives a simple example of an infinite loop (Fig. 3a). The tutorial then explains the CCR flags and conditional branch commands with similar syntax explanations and examples. After the tutorial, the student is given a puzzle based on the newly learned commands. "Use a loop to move the robot off the right side of the screen." The student writes their first attempt and hits build (Fig. 4a). The compiler finds

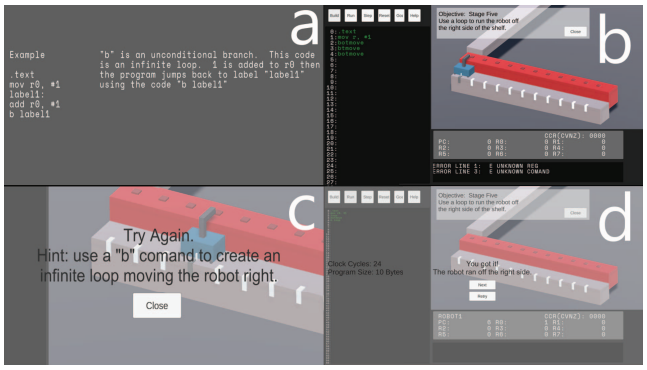

Fig. 3. (a) The tutorial's example code for an infinite loop. (b) The student's first compile attempt with error messages. (c) The student's first run attempt, the robot did not travel far enough, and the game gives a hint. (d) The student's program succeeded, and the game scores the memory and clock cycle use.

two errors in the student's code and reports them to the debug console (Fig. 3b). The first error: "Error line 1: unknown reg" was caused because the student did not write a number for the register, and the second one: "Error line 3: unknown command" was from misspelling 'botmove'. Using these messages the student can quickly find their mistakes and correct them. The student runs their code but it only moves the robot over by two spaces. The game asks the student to try again and gives them a hint, "use a `b` command to create an infinite loop" (Fig. 3c). The student remembers the example code given in the tutorial for an infinite loop and recreates that structure but with the `botmove` command (Fig. 4b line 3). Their new program complies without error and the robot successfully completes the puzzle. Finally, the end screen congratulates the player and gives the program's memory usage and clock cycles used (Fig. 3d). The next puzzle reinforces the skills the student just learned. Giving them a similar puzzle to keeps them from getting overwhelmed, but it adds some extra complexity to keep the student from getting bored, hopefully keeping them in a flow state.

V. Discussion and Future Work

Assembly programming is a good candidate for teaching programming and computer hardware due to its simple structure. Learning assembly programming helps students understand how the computer works at a low level and write code that is optimized for efficiency. Because of its limited instruction set and consistent syntax, less time needs to be spent fixing syntax errors when using this language. This allows students to spend more time developing their problem-solving and logical thinking abilities.

An advantage of using a complete emulator for running the student's code is that the quirks of writing and debugging real ARM assembly carry over. The ARM instruction set does not have a `mov` command. `mov` gets replaced with `add #0` by the compiler. Functionally this does the same thing, but when debugging the code the `mov` will show up as `add` in

(a) First

```
0: .text
1: mov r, #1
2: botmove
3: btmove
4: botmove
```

(b) Final

```
0: .text
1: mov r0, #1
2: loop:
3: botmove
4: b loop
```

Fig. 4. (a) the student's first attempt that produced compiler errors. (b) the student's code that completed the puzzle.

the debugger. By using a complete emulator, students can get used to seeing quirks like these.

We found that using a virtual robot has advantages over using a physical robot. One advantage is that setting up and calibrating a physical robot can take a lot of time, while a virtual robot can eliminate this issue. Additionally, a virtual robot is available to students at any time, unlike a physical robot that may only be accessible during lab hours. These limitations and delays can lead to student frustration, as highlighted in Fernández's study [8].

One area we would like to expand upon is visualization of the hardware. Showing how the commands are being decoded and moved around the registers and ALU during the fetch-execute cycle. This could further improve the students' understanding of low-level programming and could open the game up to supporting micro-programming puzzles.

The assembly language is a powerful tool for students to learn. It gives them the freedom to focus on problem-solving rather than fixing syntax errors. We believe that Assembly Academy can serve as the foundation for an educational game that can introduce beginner programmers to the assembly language by visualizing their code through the use of virtual robots.

References

[1] S. Combéfis, G. Beresnevičius, and V. Dagienė, "Learning programming through games and contests: overview, characterisation and discussion," *Olympiads in Informatics*, vol. 10, no. 1, pp.39-60, 2016.

[2] F. Kalelioğlu, "A new way of teaching programming skills to K-12 students: Code.org," *Computers in Human Behavoir*, 52, Nov., pp.200-210, 2015.

[3] R. Logozar, M Horvatic, I Sumiga, M Mikac, "Challenges in teaching assembly language programming - desired prerequisites vs. students' initial knowledge," *IEEE Global Engineering Education Conference,* pp.1689-1698, 2022.

[4] A. Pásztor, R. Pap-Szigeti, E. Torok, "Mobile robots in teaching programming for IT engineers and its effects," *International Journal of Advanced Computer Science and Applications*, vol. 4, no. 11, pp.162-168, 2013.

[5] R. Rajaravivarma, "A games-based approach for teaching the introductory programming course," *ACM SIGCSE Bulletin,* vol. 37, no. 4, Dec., pp.98-102, 2005.

[6] C. Zilles, "SPIMbot: an engaging, problem-based approach to teaching assembly language programming," *Proceedings of the 2005 Workshop on Computer Architecture Education: Held in Conjunction with the 32nd International Symposium on Computer Architecture*, June, pp.4-es, 2005

[7] *ARM 7TDMI Data Sheet*, Advanced RISC Machines Ltd, Cambridge, United Kingdom, pp.5-1 - 5-46, 1995.

[8] B.G. Fernández, et al. "Robotics vs. game-console-based platforms to learn computer architecture," *IEEE Access,* vol. 8, May, pp.95153-95169, 2020.