CP2: Playing Blackjack

*Note: Sub-titles are not captured in Xplore and should not be used

Eugene Thompson
Mathematics
Ursinus College
Collegeville, Pennsylvania
euthompson@ursinus.edu

Susannah Cheezum
Mathematics
Ursinus College
Collegeville, Pennsylvania
sucheezum@ursinus.edu

***Abstract*—This project serves as a final project in CS 274 – Computer Architecture and Organization at Ursinus College, taught by Dr. Santiago Núñez-Corrales. The project sought to test our ability to implement NASM assembly language code.**

Keywords—AI, memory, iteration,

I. Introduction

The latter half of Computer Architecture and Organization focused on theory of architecture as well as NASM assembly code. Throughout the past months, we were tasked with various assembly code assignments to prepare us for this culminating project. The specifications require that students work in teams of two to develop an interface in which a player can play a game of Blackjack against the computer. The implementation includes multiple levels of risk, betting amounts, as well as computer behavior accounted for in game mechanics.

Through leading assignments from our professor, we isolated larger components of the project and worked on them individually. We selected the appropriate person for each task based on aptitude and interest, that person then tackled their assigned portion. Next, after receiving feedback from our professor, we were able to connect each individual piece. This process eased our minds going into the final product, a result that we are proud of.

II. Software Design

The turn and win dynamics within our project were rather straightforward to implement. Through intricate iterative loops, we manage the flow of each turn with precision, ensuring seamless transitions between player decisions and AI actions. These loops dynamically adjust to the ever-changing state of the game, facilitating smooth gameplay and maintaining suspense until the final outcome. Additionally, our algorithm for determining wins and losses is robust and comprehensive, taking into account various factors such as hand values, Blackjack combinations, and betting amounts.

The user interface for this project is simple and readable, clearly representative of each corresponding card. Players can easily make betting decisions using straightforward commands, while clear visual cues communicate game state and outcomes effectively. Moreover, our project supports flexible customization options, allowing users to adjust risk levels and skill settings effortlessly to tailor the experience to their preferences.

Through meticulous data management and efficient memory allocation, we maintain accurate records of card distributions, player hands, and dealer actions throughout the game. This enables seamless gameplay progression and ensures that each turn unfolds authentically, reflecting the unpredictability of real-world Blackjack scenarios. Additionally, our project implements algorithms for randomness and preventing unfair advantages for either the player or the AI opponent.

III. Implementation

One of the primary advantages of using assembly language is its low-level control over hardware resources, allowing for optimized execution speed and minimal memory footprint. By leveraging this capability, we ensure that our game runs smoothly even on resource-constrained systems, delivering a responsive and enjoyable experience to players.

However, working with assembly language also presents certain limitations and challenges. The lack of high-level abstractions means that code readability and maintainability can suffer, making it more challenging to debug and update the project over time. To address this, we adopt a structured and modular programming approach, breaking down complex tasks into smaller, more manageable components, as designated by our professor. This not only improves code organization but also facilitates collaboration among team members and simplifies future modifications.

Another challenge in implementing our project lies in managing program complexity while maintaining a balance between performance and functionality. As the game grows in scope and features, the risk of introducing bugs and inefficiencies increases, potentially compromising the overall quality of the gaming experience. To mitigate this risk, we employ rigorous testing methodologies, including unit testing, integration testing, and stress testing, to identify and resolve issues early in the development process. Additionally, we continuously optimize our

codebase through profiling and performance analysis, identifying bottlenecks and inefficiencies to ensure optimal runtime performance.

Conclusion

In terms of achievements, Thompson was able to carefully craft the user interface and front end visuals and organization. His attention to detail and mathematical prowess were incredibly useful for the implementation of this intricate project. On the other hand, Cheezum was useful in coding repetitive logic algorithms, such as the dynamics of turns and game mechanics. Cheezum was also active in terms of team management.

One of the most significant limitations was the inherent complexity of working with low-level programming languages like assembly. The lack of built-in abstractions and the need for manual memory management required us to invest additional time and effort in designing and implementing robust solutions. This complexity also made it more challenging to debug and maintain the codebase, highlighting the importance of careful planning and documentation from the outset.

Despite this, we emerged triumphant. As mathematicians, we are regularly exposed to the concept of abstraction, our familiarity with the concept proved useful in this project. Our patience and care when it came to coding this project ensured the best possible product we could produce as a team.

Acknowledgments

We would like to thank our professor, Dr. Núñez-Corrales for graciously guiding us through this process. His assistance in office hours as well as shared resources helped us to make headway on this cumbersome project.