Course Project 2: Playing Blackjack

Michael Cummins
Computer Science and Philosophy
Ursinus College
micummins@ursinus.edu

Isabelle Son
Computer Science
Ursinus College
isson@ursinus.edu

***Abstract*— —This blackjack game specification is an evaluation of the course project for Ursinus College's Computer Science Architecture and Organization course. The course was taken by the above students in the Spring of 2024, taught by Dr. Santiago Núñez-Corrales.**

I. Introduction (*Heading 1*)

We were tasked with creating a blackjack game using assembly code, specifically the 8086 emulator found at https://yjdoc2.github.io/8086-emulator-web/. The game involves a singular user playing against the computer in a modified version of blackjack. There are various modes that the computer can operate in, such as betting and difficulty modes, as well as decision making which needed to be implemented. On the user end, text prompts and the translation of answers needed to be generated in order to record user decisions.

We split the problems fairly evenly between the partners as one focused on the user and front end of the game and the other focused on the computer opponent and the logic needed to implement the decision making. Once the divided problems were completed, we worked together to complete the skeleton structure of the code and fill in all of the blanks. It was important to use similar syntax and terminology to ensure that all memory locations were accessed correctly.

II. Software Design Decisions

A. Front End/User Programming

For the front end display, textual representation of cards was used. An algorithm to determine how the cards were represented based on their index in a 52 card deck was implemented rather than storing each representation in order to save memory space. With this saved memory space, the textual cues to the user were able to be stored, allowing for more ease of use for the user as the instructions could be longer and more detailed. Since the 8086 emulator only has one line of input, the user is required to input small, specific numbers or characters in order to indicate an action; these are clearly outlined in the user instructions which are displayed using an interrupt at the appropriate times.

B. Computer Decision/Logic Programming

The computer logic decisions were complicated to implement, so procedures were utilized in order to simplify the complexity. The computer had three possible betting modes, being aggressive, normal, and conservative, where each results in the computer betting a certain percentage of the user's bet. Therefore, three separate procedures were created in order to be able to calculate the computers bet based upon a numerical representation of the betting mode. Procedures were also implemented to determine the starting amount of money the computer has, as it is related to the difficulty setting for the user. Numerical representations were used to represent an easy, medium, and hard difficulty, and directed the program to procedures which determined the computers starting sum. The actual gameplay decisions, being hit, stand, or fold, was determined using real blackjack logic where based upon the likelihood of reaching 21 as well as the player's cards, the computer will either hit or stand. The computer folds if there is no numerical possibility of beating the player.

C. Combining the elements

Before completely combining the user and computer components of the game, a code skeleton was constructed in order to determine where each components should be placed in the code. This allowed for a much quicker and smoother combination of the two elements of the code, as they seamlessly fit into the slots of the skeleton, with only a few tweaks necessary. The main issue that arose when combining the code was different syntax for memory elements and labels, but this was quickly fixed as a standard naming system was determined.

III. Implementation

The advantages of our design is the speed of the computer logic and the tracking of the accomplished portions of the game. The way memory was allocated allowed for easy tracking of cards used by utilizing fifty two empty slots of memory to record whether or not a card was used. We also allocated memory to make it easier for the user to understand what was necessary with clear and concise messages.

One clear disadvantage of the design is the runtime, as the emulator performs one singular line execution at a time. This allowed for easy debugging and code tracing, but since the game contains around 1200 lines of code it takes a long time to complete. This limited testing and overall enjoyment in playing the game. Another limitation is that the emulator only provides one line for input and one line for output. Therefore we had to simplify and shorten the representations of the cards as well as the user inputs and instructions.

We believe that this game would be much improved if it was translated into TASM and ran on Visual Studio Code. Then, there would be a much larger space for inputs and outputs, and the code would run much faster. If further work is to be done for this game, translating to TASM would be the first step, and then other possible improvements could include making the games rules more similar to the real life

card game and making the card representations more aesthetically pleasing.

IV. Discussion and Conclusions

Son performed excellently in the implementation of the user interface and the communication between the game and the user. Specifically, her algorithm for the translation of the numerical representation of the card into the textual representation worked perfectly within this game. Cummins provided the code to determine the logic and behavior behind the computer opponent and its various states.

We believe that this game was overall a success, but has much to be improved before being a fully fledged blackjack game. With more time and experimentation with TASM the game could become exponentially better and more playable. However, the present state of the game accomplishes what we set out to do and we are proud of our work.

Through the completion of this project, our skills in assembly language programming greatly increased. While similar to high-level-language programming, the mental processes that need to be accomplished when programming in assembly are extremely different. We believe that this project was an excellent way to hone and display our skills in assembly language that we have been learning throughout this half of the semester.

V. Acknowledgements

We would like to offer acknowledgement to our professor, Dr. Santiago Nuñez-Corrales, for introducing us to assembly language, the 8086 emulator, and this project in general. His assistance greatly helped us form our game.

VI. References

We utilized the 8086 emulator and instruction set found at https://yjdoc2.github.io/8086-emulator-web/