Project Title: CS 200 Project 2 Card Dealer

Kaleb Coleman

Due Date: April 30, 2025

Northern Arizona University

- **Overview of the project**:

  The goal of this project was to create an assembly program that simulates shuffling drawing cards from a deck of 52. The main components are the shuffle routine that copies and randomizes the cards, and a draw card routine that allows drawing a card from the deck one at a time. User interaction is handled through a loop that processes S for shuffle, D for draw, and Q for quit.

  The main challenge was managing array manipulation, integrating a working random number generator, and ensuring safe memory access in MIPS assembly but overall I had a good experience working on this project.

- **Design/Algorithm explanation**:

  Fisher-Yates shuffle algorithm

  The shuffle uses the Fisher-Yates algorithm which iterates through the array and swaps each element with a randomly chosen index. This makes sure that it is a uniform and fair shuffle without duplicating or missing any cards. This algorithm works because it systematically assigns each element to a position with equal probability, which makes sure that all changes are equally likely and this simple mathematical workings of this help for a great shuffle algorithm to use.

  Tracking drawn and discarded cards

  To track the cards that were drawn I used a separate discard array. Everytime a card is drawn, it is moved from the deck and placed at the top of the discard array. The drawn slot in the deck is cleared using sw $zero and the draw index is decremented. This allowed me to track the history of drawn cards in order.

  Random Number Generation

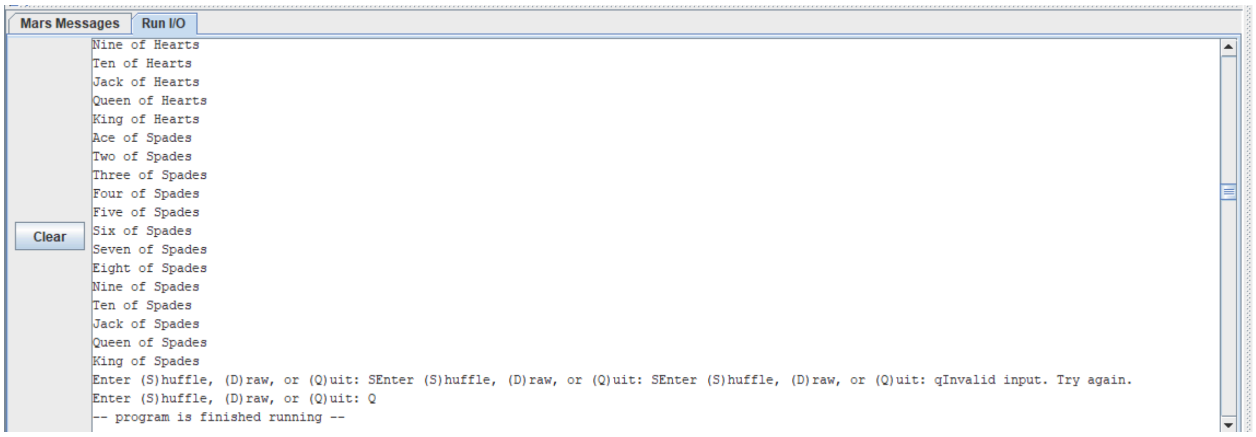
  I implemented the RNG by using the linear congruence method. I used a large prime multiplier which was 1073807359 and syscall 30 to get the system time as the seed. The RNG is used during the shuffle to pick the index between 0 and 51.

  Design tradeoffs

  Although I'm sure I could have avoided using a discard array by just tracking the draw index I felt that using the discard array helped me get it working and also helped with debugging and testing the behavior of the program. It also allowed me to implement a good way to store the drawn card history.

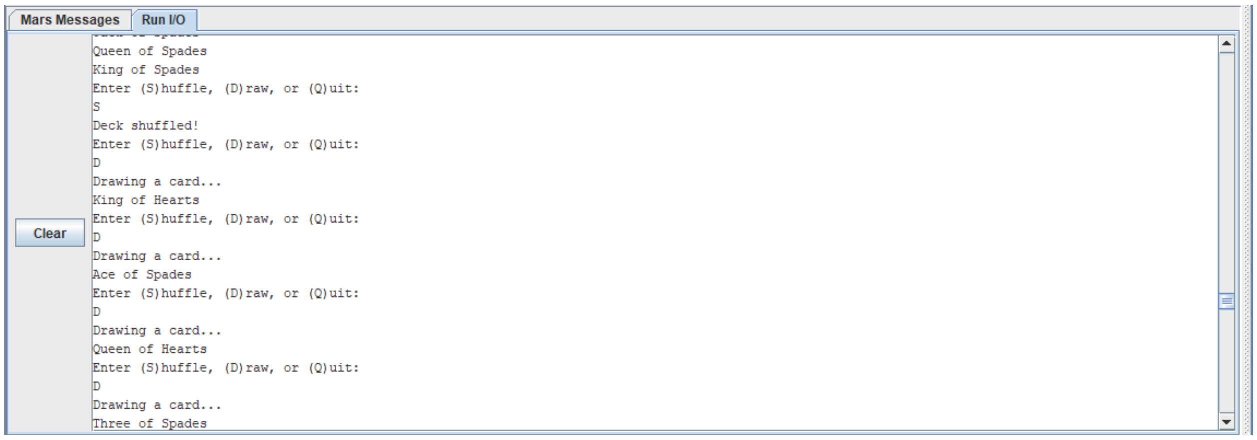
- **Testing and Sample Output**:

Image 1



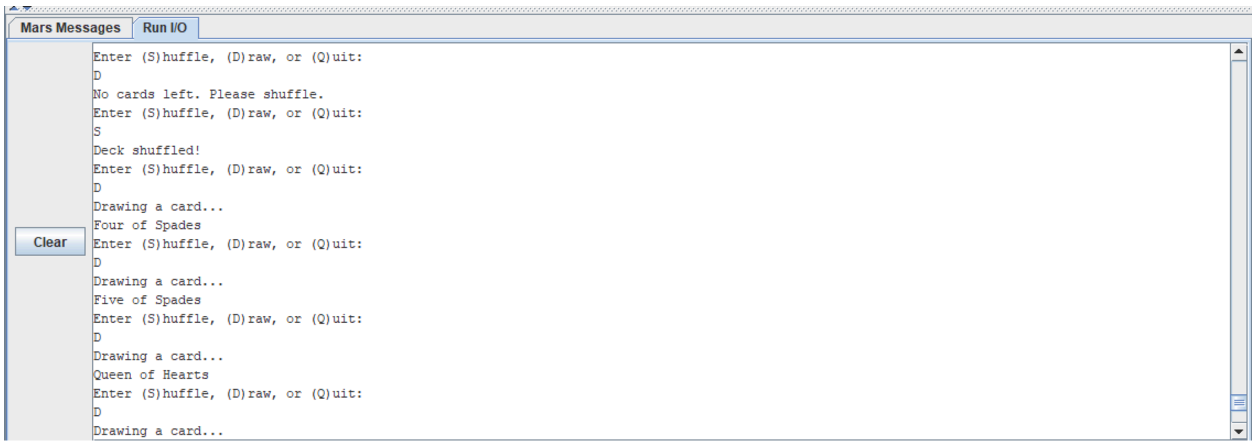
```
Nine of Hearts
Ten of Hearts
Jack of Hearts
Queen of Hearts
King of Hearts
Ace of Spades
Two of Spades
Three of Spades
Four of Spades
Five of Spades
Six of Spades
Seven of Spades
Eight of Spades
Nine of Spades
Ten of Spades
Jack of Spades
Queen of Spades
King of Spades
Enter (S)huffle, (D)raw, or (Q)uit: SEnter (S)huffle, (D)raw, or (Q)uit: SEnter (S)huffle, (D)raw, or (Q)uit: qInvalid input. Try again.
Enter (S)huffle, (D)raw, or (Q)uit: Q
-- program is finished running --
```

Image 2



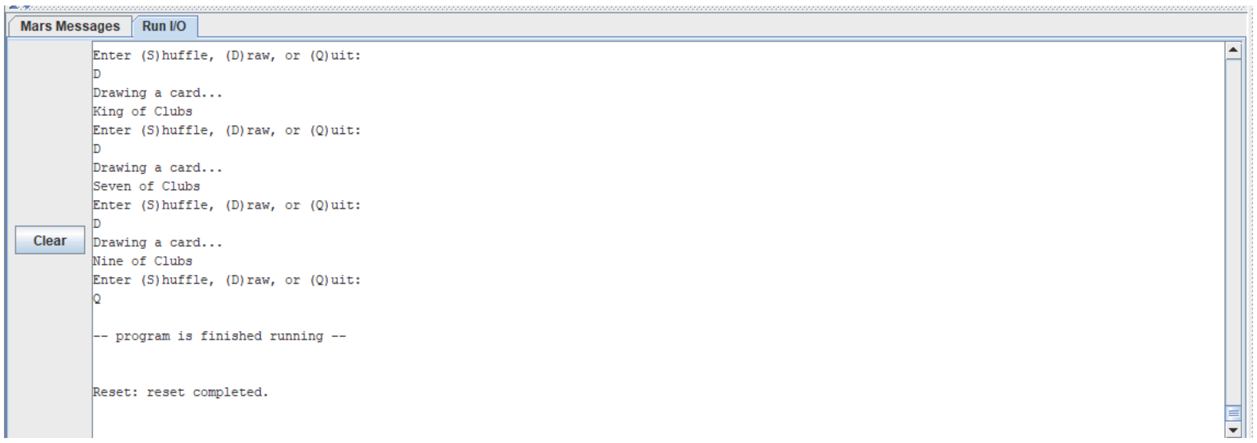
```
Queen of Spades
King of Spades
Enter (S)huffle, (D)raw, or (Q)uit:
S
Deck shuffled!
Enter (S)huffle, (D)raw, or (Q)uit:
D
Drawing a card...
King of Hearts
Enter (S)huffle, (D)raw, or (Q)uit:
D
Drawing a card...
Ace of Spades
Enter (S)huffle, (D)raw, or (Q)uit:
D
Drawing a card...
Queen of Hearts
Enter (S)huffle, (D)raw, or (Q)uit:
D
Drawing a card...
Three of Spades
```

Image 3



```
Enter (S)huffle, (D)raw, or (Q)uit:
D
No cards left. Please shuffle.
Enter (S)huffle, (D)raw, or (Q)uit:
S
Deck shuffled!
Enter (S)huffle, (D)raw, or (Q)uit:
D
Drawing a card...
Four of Spades
Enter (S)huffle, (D)raw, or (Q)uit:
D
Drawing a card...
Five of Spades
Enter (S)huffle, (D)raw, or (Q)uit:
D
Drawing a card...
Queen of Hearts
Enter (S)huffle, (D)raw, or (Q)uit:
D
Drawing a card...
```

Image 4

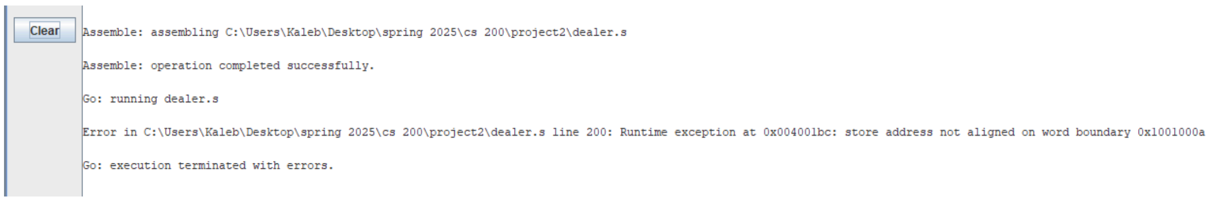


```
Enter (S)huffle, (D)raw, or (Q)uit:
D
Drawing a card...
King of Clubs
Enter (S)huffle, (D)raw, or (Q)uit:
D
Drawing a card...
Seven of Clubs
Enter (S)huffle, (D)raw, or (Q)uit:
D
Drawing a card...
Nine of Clubs
Enter (S)huffle, (D)raw, or (Q)uit:
Q

-- program is finished running --


Reset: reset completed.
```

Image 5



```
Assemble: assembling C:\Users\Kaleb\Desktop\spring 2025\cs 200\project2\dealer.s

Assemble: operation completed successfully.

Go: running dealer.s

Error in C:\Users\Kaleb\Desktop\spring 2025\cs 200\project2\dealer.s line 200: Runtime exception at 0x004001bc: store address not aligned on word boundary 0x1001000a

Go: execution terminated with errors.
```

In Image 1 you can see that my prompt was not working and it would only quit out of the program there was no new line or even a space after I typed s and also when I did lowercase letters it did not work so I quickly added code to print a newline after input and also I converted each letter into uppercase before it gets processed so that the user can use lowercase letters as well.

In Image 2 you can see after I enter S, the program prints “Deck shuffled!” on a new line. After that It shows that when entering D, it draws a card from the deck and prints “Drawing a card…” as well as prints the card name all on new lines. I did this to confirm that I did shuffle the deck and it was correctly randomized, and that the draw function correctly moves the card to the discard array and displays the card.

In image 3 I showed the output which is after I did about 50 draw commands in a row after the initial shuffle and it then shows that the deck is empty and says “No Cards left. Please shuffle” and then after I shuffle again it allows me to continue to draw cards. This means each draw correctly decrements the draw index and removes the card from the deck and functions like a stack and I am also discarding the cards as needed.

In image 4 I just show that after drawing as many cards as you would like the program will quit after the user enters Q or q

In image 5 I show the error code that says “Runtime exception at 0x004001bc: store address not aligned on word boundary 0x1001000a” which I could not figure out and I fixed this by adding .align 2 for deck and discard.

- **Reflection**:
  Implementing the core subroutines shuffle, drawcard, and getrandom52 went smoothly once I broke them down into smaller steps. The Fisher-Yates shuffle worked well as expected and printing out the cards was simple once I got the shuffle working well.

  What was tricky was some sort of alignment issues with sw and lw, which were giving me an error for unaligned memory addresses for deck and discard and this was fixed by adding .align 2 in the .data section. Another issue I faced was handling the prompt formatting and newlines in the code. Without printing a newline it makes the program look broken.

  Overall, I took my time on each piece, I debugged step by step, and I feel really confident that the code and the report show the effort I put into the project.