Universidade Estadual de Maringá
Departamento de Informática
Disciplina: Programação em Linguagem de Montagem
Professor: Ronaldo Augusto de Lara Goncalves

Relatório Técnico
Jogo de Truco em Assembly

Equipe:

RA 99660 - Henrique Misael Machado
RA 99514 - William Rodrigues da Silva

Maringá, 2018

Sumário

Introdução

O presente trabalho tem como objetivo a aplicação dos conceitos e conteúdos aprendidos em sala de aula na matéria de Programação em Linguagem de Montagem, ministrada pelo professor Ronaldo Augusto de Lara Goncalves no ano de 2018, para o desenvolvimento de um Jogo de Truco Paulista implementado em Assembly.

Montando e executando o código fonte

Para poder jogar o Jogo de Truco implementado deve-se seguir os passos descritos em um dos métodos a seguir.
Como pré-requisito para ambos os métodos deve-se ter instalada na máquina a biblioteca gcc-multilib. Para instalação da biblioteca, digite o seguinte comando no terminal:

sudo apt-get install gcc-multilib

Montagem e execução manuais

Digite os seguintes comandos no terminal:

as -32 truco.s -o truco.o
ld -m elf_i386 truco.o -l c -dynamic-linker /lib/ld-linux.so.2 -o truco
./truco

Montagem e execução com script

Digite o seguinte comando no terminal:

./ce truco

Este método executa os mesmos comandos do método manual mas de forma automatizada, com o uso de um script (ce) que deve estar na mesma pasta do código fonte, sendo necessário passar como argumento apenas o nome do arquivo que contém o código fonte (sem extensão).

Definições e acrônimos

Vira: É a carta que o embaralhador vira quando distribui as cartas. É a carta que definirá as 4 manilhas.

Manilhas: São as cartas mais fortes do jogo, mais fortes do que o 3.

Rodada: Fração da mão. Em cada rodada, os jogadores mostram uma carta. A carta mais forte ganha a rodada.

Mão: Fração da partida, vale 1 ponto e poderá ter seu valor aumentado para 3, 6, 9 e 12 pontos. É disputada em melhor de 3 rodadas.

Partida: Jogo valendo 12 pontos, conseguidos através das “mãos”.

Truco : É um pedido de “aumento de aposta” que só pode ser feito na vez de cada jogador. Se nenhum jogador fizer o pedido, a partida valerá 1 ponto. Se o Truco for pedido e o adversário aceitar, a partida passa a valer 3 pontos. Se o adversário não aceitar, o jogador que fez o pedido ganha um ponto.

Tentos: Termo usado como adjetivo de pontos.

Seis: É um pedido de “aumento da aposta” que pode ser feito quando um jogador é desafiado com o pedido de Truco. Neste caso, a partida que inicialmente valia um ponto e foi trucada, passando a valer 3, passará a valer 6, caso o adversário aceite. Se o adversário não aceitar o pedido, o desafiador ganha 3 pontos.

Nove: É um pedido de “aumento de aposta” que pode ser feito quando um jogador é desafiado com o pedido de “Seis”. Neste caso, a partida que estava valendo 6, passará a valer 9 com esse pedido, caso o adversário aceite. Se o adversário não aceitar o pedido, o desafiador ganha 6 pontos.

Doze: É um pedido de "aumento de aposta" que pode ser feito quando um jogador é desafiado com o pedido de "Seis". Neste caso, a partida que estava valendo 6, passará a valer 9 com esse pedido, caso o adversário aceitei. Se o adversário não aceitar o pedido, o desafiador ganha 9 pontos.

Mão de Onze: O jogador que consegue chegar a 11 pontos na partida, tem o direito de olhar as suas cartas e analisar se irá ou não aceitar a partida. Caso o jogador aceite, a Mão de Onze já começa valendo 3 pontos. Se achar que não será possível vencer a mão com tais cartas, podem optar por “Correr”, dando apenas 1 ponto ao adversário.

Carta fechada: É quando um jogador joga a carta virada para a mesa, passando assim a não valer nada. Também chamado de “carta coberta” ou “carta encoberta”. Não é permitido esconder a carta na primeira rodada de cada mão.

Descrição dos principais módulos

A seguir uma breve descrição dos principais módulos que compõe a implementação do Jogo de Truco.

iniciaVariaveis

Este módulo é responsável por inicializar todos os rótulos necessários declarados em .section .data com o objetivo de manter o bom funcionamento durante a execução do jogo, suas mãos e rodadas. Tal módulo é chamado sempre que uma nova mão se inicia.

geraSementeRandom

Configura uma semente aleatória com base no tempo que será utilizada como valor inicial para geração de números aleatórios.

geraRandom

Gera um número aleatório entre 0 e intervalo-1. Sendo intervalo um rótulo que deve ser configurado antes da chamada de tal módulo.

geraCartaValida

Gera um número aleatório entre 0 e 39, através de chamada ao módulo *geraRandom* e verifica se o número gerado não é igual ao número contido nos rótulos que armazenam as cartas dos jogadores e a carta vira. Caso seja um número repetido gera um novo aleatório. Desta forma evita-se cartas repetidas na execução do jogo.

distribuiCartas

Gera 7 números através de chamadas ao módulo *geraCartaValida* e armazena nos rótulos referentes às 3 cartas de cada jogador e a carta vira.

imprimeAbertura

Imprime as mensagens iniciais sobre o programa

imprimeCarta

Recebe um número entre 0 e 39 como parâmetro pelo registrador *eax* e imprime a carta correspondente usando o vetor de mapeamento.

imprimeCartasJ1

Imprime todas as cartas disponíveis para Jogador 1 (Humano) através de chamadas ao módulo *imprimeCarta*. O controle de cartas disponíveis é feito através de rótulos usados como flags, por exemplo *flagUsouCarta1J1*.

imprimeTodas

Imprime todas as cartas disponíveis para ambos jogadores. Utilizado para fins de testes (quando é interessante que o operador veja as cartas do computador.

imprimeVira

Imprime a carta vira a fim de informar ao usuário

imprimePlacar

Imprime na tela o placar do jogo, informando os tentos de cada jogador.

defineManilha

Define as manilhas do jogo com base na carta vira e armazena seu valor no rótulo *manilha*.

compara2Cartas

Usa como parâmetro os número armazenados nos rótulos *cartaEscolhidaJ1* e *cartaEscolhidaJ2* e faz a comparação entre as cartas para ver qual dos jogadores ganha a rodada atual setando o rótulo *flagGanhouAtual*. Existe também a possibilidade de empate. Tal comparação leva em conta a carta vira da mão e portanto suas manilhas.

sorteiaPrimeiro

Na primeira mão do jogo escolhe aleatoriamente o jogador que inicia a partida, nas seguintes mãos a ordem de quem inicia a mão vai se alternando entre os jogadores.

escolheCartaJ1

Imprime as cartas disponíveis para o Jogador 1 (Humano) e solicita que ele escolha uma delas para jogar. Pergunta também se ele deseja jogar a carta aberta ou fechada.
Caso ainda esteja na primeira rodada da mão o jogo não oferece a opção de carta aberta ou fechada.

escolheCartaJ2

Verifica qual a rodada atual, quem iniciou a rodada, qual carta jogada pelo adversário, e define então, o algoritmo que usará para escolher a carta, considerando também o resultado de rodadas anteriores.

verificaGanhadorMao

Verifica se algum jogador ganhou a mão atual com base nos rótulos que representam as flags de quem ganhou as rodadas, por exemplo, *flagGanhouRodada1*.
O módulo leva em consideração as seguintes regras do jogo:
Se empatar na primeira rodada, quem ganhar a segunda vence a mão;
Se empatar na segunda rodada, quem ganhou a primeira vence a mão;
Se empatar na primeira e segunda rodada, quem fizer a terceira vence a mão;
Se empatar na terceira rodada, quem ganhou a primeira vence a mão;
Se todas as três rodadas empataram, ninguém ganha ponto.

verificaGanhadorJogo

Verifica se um dos jogadores já atingiu a pontuação de 12 ou mais tentos. Caso afirmativo o jogador que atingiu tal pontuação ganhou a partida.

executaMao

Inicializa as variáveis para evitar informações de outras mãos influenciarem a mão atual, distribui três cartas para cada jogador, define a manilha virando a próxima carta e mostra a vira na tela. A mão é composta por até 3 rodadas, em cada rodada um jogador é definido para jogar (aleatoriamente no início, e após a primeira rodada, o vencedor da rodada anterior deve iniciar), e pode escolher pedir truco ou jogar cartas. Ao jogar carta, pode-se jogar aberta ou fechada (e aí a rodada é considerada vencida pelo adversário); ao pedir truco, o adversário pode aceitar, fugir (e nesse caso ele perde com o valor da aposta anterior), ou aumentar a aposta (para 6); quando a aposta é aumentada, volta ao jogador que pediu truco aceitar (6), correr (perder 3) ou pedir aumento da aposta novamente (9), e assim até um limite de valor de aposta 12.

executaMao11

Faz as mesmas coisas que o executaMao, mas se divide na execução das rodadas, pois não é possível pedir truco, e o jogador que está “na mão de 11” pode prosseguir desistindo da mão, e trocando suas cartas, ao custo de o adversário ganhar 1 ponto. Caso o jogador decida jogar a mão, e perder, o adversário ganha 3 pontos.

executaMaoSemTruco

No caso de ambos os jogadores terem 11 pontos, então temos uma mão onde não é possível aumentar a aposta (truco/seis/nove/doze), mas que deve ser jogada obrigatoriamente, uma vez que a decisão por troca de cartas de um dos jogadores resultaria na vitória do outro.

verificaMao11

Verifica inicialmente se o jogador 1 tem onze pontos. Se tiver, então verifica se o jogador também o tem: se ambos tiverem, então é setada a flag *flagJogadorTem11* com valor **3**; se apenas o jogador 1 tiver, então é setado *flagJogadorTem11* com valor **1**. Se o jogador 1 não tiver 11, então verifica-se se o segundo jogador possui 11: em caso positivo, *flagJogadorTem11* recebe **2**, se não, 0.

probabilidadeJ2pedirTruco

Calcular a probabilidade com base 5, e roda um número aleatório; se o número resultante for menor que a probabilidade, então o rótulo *numDigitado* é setado como 1, para que o truco seja pedido; do contrário, setado como 2 que significa “não”.

probabilidadeJ2aceitar12

Calcular a probabilidade com base 3, e roda um número aleatório; se o número resultante for menor que a probabilidade, então o rótulo *respostaApostaJ2* é setado como 1, para que o aumento de aposta seja aceito; do contrário, setado como 2 que significa “não”.

probabilidadeJ2_foge_aceita_aumenta

Realiza um cálculo dinâmico para definir a base da probabilidade, de acordo com o valor atual da aposta, a fim de ter as seguintes bases para respectivas apostas:

Para a aposta inicial (**truco**), base 8
Para **seis**, base 6
Para **nove**, base 4
Para **doze**, base 2

Calcular a probabilidade para esta base, e caso o valor aleatório gerado seja maior que ela, *respostaApostaJ2* recebe o valor **0** que significa que ele recusou o aumento (fugiu). Caso seja menor, então ainda um novo valor aleatório é rolado e há 30% de chances de o jogador aumentar a aposta, e 70% de apenas aceitar, onde *respostaApostaJ2* é setado com **1** ou **2**, respectivamente.

opcaoJ1pedirTruco

Imprime e pergunta se o Jogador Humano deseja pedir truco, informa as opções (1 sim; 2 não), e calcular a resposta. Se a resposta for 1, seta que houve truco pedido na mão, calcular a resposta do jogador 2 (foge, aceita ou aumenta), e executa:

- Se fugiu, o jogador 1 ganha a mão;
- Se aceitou, aumenta a aposta para 3;
- Se aumentou, aumenta a aposta para 3, e pergunta se o jogador 1 quer fugir, aceitar ou aumentar
    - Se fugir, o jogador 2 ganha a mão;
    - Se aceitar, aumenta a aposta para 6;
    - Se aumentou, aumenta a aposta para 6, e calcular a decisão do jogador 2
        - Se fugir, o jogador 1 ganha a mão;
        - Se aceitar, aumenta a aposta para 9;
        - Se aumentar, aumenta a aposta para 9, e pergunta a decisão do jogador 1
            - Se fugir, o jogador 2 ganha a mão;
            - Se aceitar, aumenta a aposta para 12.

opcaoJ2pedirTruco

Calcular a probabilidade de o Jogador 2 (Máquina) pedir truco, e se aceitar, imprime as cartas do jogador 1, marca que o truco já foi pedido na mão, e pergunta ao jogador 1 se ele foge, aceita ou aumenta.

- Se fugir, o jogador 2 ganha a mão;
- Se aceitar, aumenta a aposta para 3;
- Se aumentar, aumenta a aposta para 3, calcular a probabilidade de o jogador 2 fugir, aceitar ou aumentar
    - Se fugir, o jogador 1 ganha a mão;
    - Se aceitar, aumenta a aposta para 6;
    - Se aumentar, aumenta a aposta para 6, pergunta ao usuário a decisão
        - Se fugir, o jogador 2 ganha a mão;
        - Se aceitar, aumenta a aposta para 9;
        - Se aumentar, aumenta a aposta para 9, e calcular a probabilidade de o jogador 2 aceitar ou fugir de 12
            - Se fugir, o jogador 1 ganha a mão;
            - Se aceitar, aumenta a aposta para 12

J1pedirTruco e J2pedirTruco

Tais módulos servem para encapsular a chamada e verificação de flags dos módulos opcaoJ1pedirTruco e opcaoJ2pedirTruco, respectivamente.

pausaExecucao

Apenas controla uma pausa para prosseguir com o ENTER fim de que o operador possa realizar a leitura mais controlada das informações da tela com menos confusão.

calculaProbabilidade

Calcular a probabilidade de determinada ação a partir das cartas do jogador computador. Requer um valor setado no rótulo *probabilidadeBase*, e apartir desse rótulo calcular uma probabilidade, verificando cada possibilidade das cartas do jogador a partir desse valor. O valor base também é o valor mínimo de probabilidade de o jogador realizar a ação, caso todas as cartas sejam contra ele.
Para cada:

- Ás o jogador possui *2x probabilidadeBase* chances de realizar a ação;
- Dois o jogador possui 3x *probabilidadeBase* chances de realizar a ação;
- Três o jogador possui 4x *probabilidadeBase* chances de realizar a ação;
- Manilha o jogador possui 5x *probabilidadeBase* chances de realizar a ação;

criaOrdemCrescenteJ2

Tem a função de ordenar as cartas do Jogador 2 para ordem crescente, a fim de facilitar na escolha de “melhor carta” e uma carta “suficiente para matar a do adversário”. Primeiramente verifica qual a menor carta, e depois compara as duas restantes (utilizando a função *maiorEntreDuasCartas*); sabendo qual a maior entre as duas restantes, e anteriormente já conhecendo a menor, ordena-se em ordem crescente.
A ordenação é abstrata, ou seja, é feita em forma de índice. Existem três rótulos *ordemCrescenteX* que indicam qual a menor carta, a segunda maior e a maior (1, 2 e 3 respectivamente). Esses rótulos são utilizados como índice, de maneira que seu valor possui 1 (quando a carta correspondente ao índice é a primeira), 2 (quando a carta correspondente ao índice é a segunda) ou 3 (quando a carta correspondente ao índice é a terceira).

menorEntre2Cartas

Primeiramente calcular o peso das duas cartas, e verifica se as cartas são manilhas.

- Se a primeira for manilha, verifica se a segunda é manilha;
    - Se a segunda for manilha, compara-se os naipes
    - Se a segunda não for manilha, a primeira vence a comparação e o resultado é a segunda carta (como menor)
- Se a primeira não for manilha, verifica se a segunda é manilha;
    - Se a segunda for manilha, ela vence e o resultado é a primeira carta (como menor)
    - Se a segunda não for manilha, a carta com maior peso vence e o resultado é a outra carta (como menor)

J2mataComMenorPossivel

Para cada uma das cartas do jogador, começando pela menor (carta apontada pelo índice 1 da ordenação abstrata), verifica se a carta está disponível para ser utilizada (se já não foi usada antes), e posteriormente se ela é suficiente para matar a carta do adversário. Se ela já foi usada, então passa a verificar a próxima carta da ordenação; se ainda não foi, verifica se ela é suficiente e se não for, segue à próxima carta também.

Esquema de funcionamento

As cartas do jogo são mapeadas de forma numérica, como o jogo de truco é composto por 40 cartas, nesta implementação as cartas são numeradas no intervalo de 0 a 39.
Foi criado então um vetor de mapeamento, para ser usado para imprimir o nome de determinada carta dado seu número. O vetor tem 40 posições e é disposto de forma crescente, tanto no valor das cartas como em seus naipes, levando em consideração as seguintes ordens:

Cartas em ordem crescente: 4, 5, 6, 7, Q, J, K, A, 2, 3
Naipes em ordem crescente: Ouros, Espadas, Copas, Paus

Portanto o vetor inicia com 4 Ouros até 3 Ouros, na sequência 4 Espadas até 3 Espadas e assim por diante até a última posição ocupada por 3 de Paus.

Nesta estrutura é necessário estabelecer uma forma de igualdade entre cartas com o mesmo valor porém naipes diferentes, tendo em vista que o naipe é levando em consideração apenas na comparação das manilhas.
Então as cartas 2, 12, 22, e 32 respectivamente 6 Ouros, 6 Espadas, 6 Copas e 6 Paus por exemplo devem ter o mesmo valor de comparação.

Para realizar esta igualdade utilizamos uma forma de “peso” que foi determinada da seguinte maneira: O peso de uma carta é equivalente ao resto da divisão de seu número por 10. Assim, as cartas do exemplo acima teriam todas o peso 2 e poderiam assim ser comparadas.

Da mesma forma o “peso” das manilhas é definidos. Se a carta vira for por exemplo a 28, seu peso será 8 e portanto, todas as cartas com peso 9 (9, 19, 29 e 39) serão manilhas no jogo.

Já na comparação de duas manilhas, utilizando o exemplo acima, entre 19 e 29 por exemplo, é levado em consideração seu número e não seu peso, uma vez que as cartas são mapeadas em ordem crescente de valor e naipe no vetor de mapeamento. Neste caso a manilha 29 ganharia da manilha 19, já que o valor de seu naipe é superior.

Tendo já definidas a estrutura de mapeamento das cartas e suas comparações bastou montar a estrutura de rodadas, mãos e por fim do jogo, utilizando os módulos já descritos na seção anterior.

Capacidades

- O jogo implementado pode ser jogado por um Jogador Humano, que jogará contra o computador (Jogador Máquina).
- As cartas de cada mão são distribuídas de forma aleatória, sem permitir repetições.
- A comparação de duas cartas é realizada levando em conta as manilhas da mão atual, determinada a partir da carta vira.
- O Jogador Humano não conhece as cartas do Jogador Máquina, e a recíproca também é válida.
- As opções de escolha do Jogador Humano são validadas pelo programa, exibido quando necessário que a opção escolhida foi inválida, solicitando uma nova entrada.
- O Jogador Humano ao jogar uma carta, tem a possibilidade de jogá-la de forma Aberta ou Fechada. Caso escolha jogar a carta fechada (virada para baixo) ele entrega a rodada atual da mão ao adversário.
- A possibilidade de jogar uma carta fechada é válida apenas a partir da segunda rodada, não sendo possível jogar uma carta fechada na primeira rodada.
- O Jogador Máquina também pode jogar uma carta fechada, de forma moderada.
- Ambos os jogadores ao jogarem uma carta, não poderão jogar a mesma carta em outra rodada, controle realizado por meio de flags. A carta que já foi jogada sequer aparecerá como opção para uso, porém, mesmo que o jogador tente usá-la será bloqueado pela validação.

- A cada rodada, ambos os jogadores têm a possibilidade de pedir/aceitar/fugir de uma aposta de Truco, 6, 9 e 12. O jogo trata todos os fluxos possíveis.
- Foi implementada a sequência completa de um jogo, com diversas mãos até que um dos jogadores atinja a quantia de 12 tentos ou mais.
- Ao final de cada mão é exibido o placar atual com a pontuação de cada Jogador.
- Na primeira mão do jogo o Jogador que inicia é decidido de forma aleatória, e nas seguintes mãos a ordem vai se alternando entre os dois Jogadores.
- Sempre, o ganhador de uma rodada da mão deve ser o mesmo a tornar, ou iniciar, a próxima rodada, se existir.
- O jogador que empatar uma rodada da mão deve ser o mesmo a tornar, ou iniciar, a próxima rodada da mão, se existir.
- O jogo trata todas as possibilidades de empate durante as rodadas para decidir o vencedor da mão atual.
- Quando um dos jogadores atinge a pontuação de exatos 11 tentos, é executada a mão de 11, onde o mesmo tem a possibilidade de ver suas cartas e decidir se vai jogar ou não a mão de 11.
  Se fugir ele dá ao adversário 1 tento.

Se aceitar a mão já inicia valendo 3 tentos, não havendo a possibilidade de aumentar tal aposta.

- Quando ambos os jogadores atingem a pontuação de 11 tentos é executada uma mão onde não existe a possibilidade de aumentar a aposta.
- As decisões do Jogador Máquina com relação a pedir/aceitar/fugir de apostas é tomada levando em conta uma probabilidade que é calculada com base no poder de suas cartas atuais.
- O Jogador Máquina sempre tenta matar a carta do adversário com sua menor carta possível
- Caso o Jogador Máquina não consiga matar a carta do adversário, ele descarta sua carta de menor valor.
- Caso a lance anterior tenha empatado, o Jogador Máquina joga sua carta de maior valor.
- Para fim de testes o programa também pode exibir as cartas da Máquina.

Conclusão

Ao concluirmos este trabalho podemos observar a importância de modularizar funções componentes de um programa, separando cada uma por suas finalidades específicas, algo que foi de grande ajuda nesta implementação em Assembly com as chamadas de subrotinas (call/ret).
A implementação Assembly possibilitou também uma visão mais próxima da arquitetura do computador, como também sobre o funcionamento das demais linguagens que já utilizamos anteriormente.
Tivemos também a oportunidade de exercitar a lógica de programação para tratar os fluxos possíveis do programa proposto, através de comparações, saltos condicionais e incondicionais.