UNIVERSIDADE FEDERAL DA FRONTEIRA SUL
CAMPUS CHAPECÓ
CURSO DE CIÊNCIA DA COMPUTAÇÃO

TRABALHO ASSEMBLY - RISC-V:

DESENVOLVIMENTO DO JOGO DE CARTAS BLACKJACK

MAIQUELI EDUARDA DAMA MINGOTI (20230004643)

CHAPECÓ, SANTA CATARINA

2024

1. VISÃO GERAL

A estrutura para guardar as cartas do jogador e do dealer foi construída na seção .data como um array estático de tamanho fixo. Utilizando a diretiva .space 40, foram reservados 40 bytes de memória contígua para cada participante (jogadorCartas e dealerCartas). Como o valor de cada carta é tratado como um inteiro word de 4 bytes, essa alocação criou um vetor capaz de armazenar até 10 cartas. O acesso e a adição de novas cartas a este espaço são gerenciados por uma variável contadora (jogadorNumCartas e dealerNumCartas), que funciona como um índice para calcular o endereço exato onde a próxima carta deve ser escrita.

1.1. REGISTRADORES

Registradores Salvos (s0-s3): Estado Persistente do Jogo. Esses aqui eu uso pra guardar o que é essencial entre as funções, já que o valor fica salvo até o fim da rodada e não some quando chama outra função. Inicializo em main. Usei como ponteiros e também para carregar os valores direto.

Registrador	**Propósito Específico no Jogo**
s0	Ponteiro para a variável jogadorVitorias.
s1	Ponteiro para a variável dealerVitorias.
s2	Armazena a Pontuação do Jogador na rodada atual.
s3	Armazena a Pontuação do Dealer na rodada atual.

Registradores Temporários (t0-t6): utilizei para cálculos e lógicas rápidas, tipo contar cartas, calcular somas, salvar endereços temporários, controlar loops, comparar valores. Não me importo de perder o valor quando chamo função, porque é coisa momentânea.

exemplos de uso: guardar valor sorteado (randnum → t0); contadores em loops (tipo resetar o baralho); endereços calculados pra salvar cartas nos vetores; e valores pra bne, beq etc.

Registradores de Argumento e Sistema (a0, a7): utilizei o a0 multifunção, a0 recebeu valor de carta sorteada, passa número a ser impresso, lê entrada do teclado, recebe retorno de funções. a7 definiu qual syscall executar: 4 pra printar string, 1 pra printar int, 5 pra ler int, 10 pra sair.

Registradores de Controle (ra, sp): para não bagunçar as funções e não voltar pro lugar errado: ra (return address) guardei para onde voltar depois que a função acaba. Toda vez que chamo função dentro de outra, salvo ra na pilha (sw ra, 0(sp)) e restauro depois (lw ra, 0(sp)). sp (stack pointer): cuida do topo da pilha, sempre que salvo ra e outros registradores, diminuo sp (addi sp, sp, -4) e recupero depois.

1.2. FLUXO PRINCIPAL

main: Inicializa o estado do jogo e gerenciar o loop principal. Salva os registradores s0-s3 na pilha para uso persistente ao longo do jogo (placar e pontuações). Zera os contadores de vitória e pontuação. Exibe a mensagem inicial e pergunta se o usuário deseja jogar. Se a resposta for afirmativa, transfere o controle para distribuicao. Caso contrário, salta para fim.

distribuicao: Chama adicionaCartaJogador duas vezes e adicionaCartaDealer duas vezes. Em seguida, exibe as cartas do jogador e a primeira carta do dealer.

turnoJogLoop: Pergunta a ação do jogador (1 para hit, 2 para stand). Se "hit", chama adicionaCartaJogador, exibe a nova mão e verifica se a pontuação excedeu 21 (estouroJog). O loop continua até que o jogador escolha "stand" ou sua pontuação ultrapasse 21.

turnoDeaLoop: O dealer deve pedir cartas ("hit") enquanto sua pontuação for inferior a 17. A função verifica a pontuação do dealer (dealerPontuacao). Se for menor que 17, chama adicionaCartaDealer e verifica se a pontuação excedeu 21 (estouroDea). Quando a pontuação for 17 ou superior, o dealer para e o controle passa para a rotina de comparacao.

endRound: Exibe o placar de vitórias. Pergunta se o usuário quer uma nova partida. Se sim, zera as mãos e pontuações da rodada e verifica se o baralho precisa ser reembaralhado (se totalCartas < 12). Em seguida, volta para distribuicao. Se não, salta para fim.

1.3. FUNÇÕES DO BARALHO

randnum: Utiliza a ecall de serviço 42 para gerar um número inteiro aleatório no intervalo [0, 12]. Soma 1 ao resultado para obter um valor de carta entre 1 (Ás) e 13 (Rei). O resultado é retornado no registrador a0.

controleCartas: Chama randnum para sortear uma carta. Verifica no array cartas se a quantidade daquela carta é maior que zero. Se for zero, sorteia novamente. Se estiver disponível, decrementa a quantidade dessa carta no array cartas e também decrementa o contador totalCartas. Retorna o valor da carta sorteada em a0.

resetBaralho: Define totalCartas de volta para 52. Percorre o array cartas e redefine a contagem de cada uma das 13 posições para 4.

adicionaCartaJogador/adicionaCartaDealer: Adiciona uma carta à mão e atualiza a pontuação. Chama controleCartas para obter uma carta válida. Armazena a carta no próximo espaço livre do array jogadorCartas ou dealerCartas. Incrementa jogadorNumCartas ou dealerNumCartas. Atualiza a pontuação (jogadorPontuacao em s2, dealerPontuacao em s3), aplicando a lógica do Blackjack: Cartas de 2 a 10 valem seu número. Cartas de figura (J, Q, K - valores 11, 12, 13) valem 10 pontos. O Ás (valor 1) vale 11 pontos, a menos que isso faça a pontuação total exceder 21, caso em que passa a valer 1 ponto.

estouroJog/estouroDea: Lida com a condição de estouro (pontuação > 21). Exibe a mensagem de vitória do oponente, incrementa o placar de vitórias correspondente e salta para endRound.

comparacao: Compara as pontuações finais. Compara os valores em s2 (pontuação do jogador) e s3 (pontuação do dealer). Salta para vitoriaJog, vitoriaDea ou exibe a mensagem de empate, conforme o resultado.

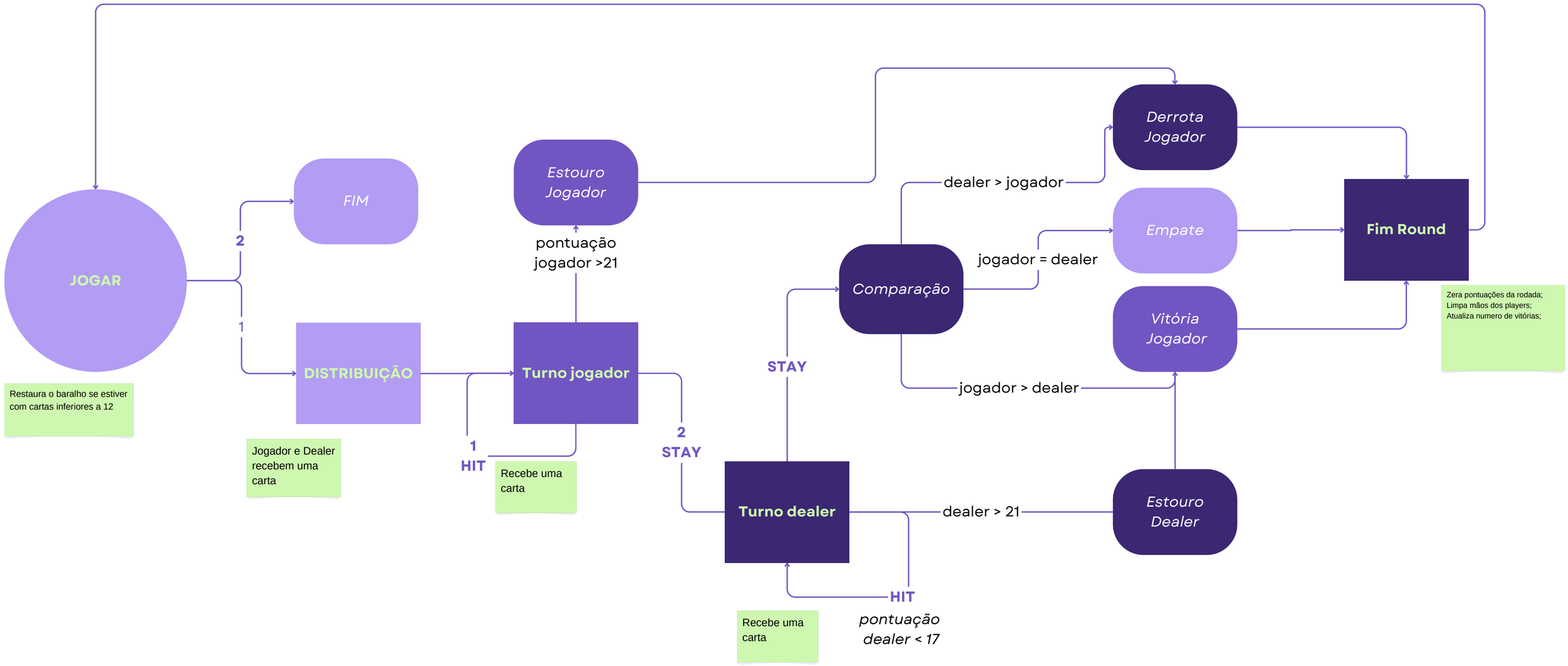
vitoriaJog/vitoriaDea: Processa a vitória. Exibe a mensagem apropriada, incrementa o contador de vitórias (jogadorVitorias ou dealerVitorias) e salta para endRound.

printInt / printString: Funções genéricas de impressão para encapsular as ecalls para impressão de inteiros (serviço 1) e strings (serviço 4).

printMsg...: Funções específicas para imprimir as mensagens do jogo, como printMsgMaoJogador, que exibe as cartas na mão do jogador e a soma total.

Há uma série de funções *printMsgMaoJogadorX* e *printMsgMaoDeaX* que lidam com a impressão de mãos com diferentes quantidades de cartas. Foi necessário criar pois houve problemas na criação do loop, e que não consegui resolver em tempo, então precisei fazer desse jeito.

1.4. FLUXOGRAMA


JOGAR
Restaura o baralho se estiver com cartas inferiores a 12
2
FIM
1
DISTRIBUIÇÃO
Jogador e Dealer recebem uma carta
Turno jogador
1
HIT
Recebe uma carta
Estouro Jogador
pontuação jogador >21
2
STAY
Turno dealer
HIT
Recebe uma carta
pontuação dealer < 17
STAY
Comparação
dealer > jogador
Derrota Jogador
jogador = dealer
Empate
jogador > dealer
Vitória Jogador
dealer > 21
Estouro Dealer
Fim Round
Zera pontuações da rodada;
Limpa mãos dos players;
Atualiza numero de vitórias;