Isabela Escolaro Delboni RGM:32599048

Projeto de Arquitetura de Sistemas Computacionais

Simulador de Processador de uma Instrução MIPS:

- Bibliotecas incluídas:
  - <stdio.h> para as funções usadas tradicionalmente em código em C, como printf,scanf, %d, %s e etc.
  - <stdlib.h> para conversão de string para inteiro (atoi),para a comparação e manipulação de strings (strcmp) e para a limpeza de saída do buffer (fflush).
  - <string.h> para a manipulação de strings, no caso copiar e até mesmo comparar.
  - <unistd.h> para usar a função sleep e dar um tempo entre uma função e outra.

- Struct: A struct definida como instrução, ela foi criada para armazenar os dados dos tipos de instruções, endereços, e valores, facilitando quando se trata passar os dados de um função a outra.
- Definição global das etapas do processador, a qual vai rastrear as etapas, juntamente com a constante, informando qual etapa atual está sendo processada.
- Constante: ela foi do tipo char, declarada como um array e já com as etapas do processador inseridas como string.
- Protótipo das funções: declaração das funções antes do main, pois a função está logo após o fim do mesmo.
- Main:
  - Declaração da struct: eu declarei a struct instrução logo no início para facilitar a passagem dos dados para as funções.
  - Declaração das variáveis como inteiro para obter o resultado das opções do switch, nos ajudando na organização do código.
  - O printf sobre os tipos de instruções está sendo no início da main, pois é uma variável que vai ser passada para as funções e vai ajudar a diferenciar os tipos de instruções.
  - Chamadas de funções: As funções de cada etapa do processador é chamada em sua ordem.
  - Funções: chamadas de funções como sleep, foram chamadas para ter um tempo entre uma impressão e outra, já o system(“cls”) foi usado para apagar as impressões anteriores.

- Funções:
  - Imprimir etapas: nela chamamos a etapa atual, que está como constante para informar qual etapa do processador está e busca a constante e usa o número da etapa atual para buscar o nome dela entre as strings declaradas no array.
  - Leitura de Instrução: usa a opção escolhida no main para o switch, assim usando o formato de cada tipo de instrução, sem formar nenhum conflito. Vai pedir o opcode, e os valores do rt e rd.

    - Impressão de instrução: com a escolha da opção ele vai imprimir com na forma estrutural, e na forma estrutural com valores de acordo com o formato do tipo escolhido.
    - Decodificação: nesta etapa usei a condição if e else com a função strcmp para comparar o opcode e achar a instrução, achando ele salva o código da instrução em binário na struct.
    - Execução: eu declaro resultado como int, para salvá-los e declaro efetivo, utilizado nas instruções i e j. Na parte da execução eu utilizo o switch, usando o binário que define na decodificação das cases. Eu adiciono ao valor de PC (4 bytes), antes da execução dos operadores lógicos ou matemáticos, para não ter nenhum erro no valor final do PC. Logo após já é executado a conta de acordo com a instrução escolhida pelo usuário e armazenado no em resultado, o qual voltou no final da função.
    - Exibir Resultados: na última etapa eu utilizo a opção escolhida no main novamente para eu conseguir exibir os resultados, e utilizo o switch para isso. Nos resultados eu exibo a forma estrutural final, o valor final, e o valor atualizado do PC.

- As passagens de valores entre a struct e as funções e o main foi utilizado o parâmetro, facilitando o acesso e para salvar valores.
- As funções feitas após o main foram feitas em ordem, para não ter nenhuma confusão e ficar mais simples de se organizar.
- O código é feito com comentários para facilitar a quem o veja.
- O simulado também está preparado para caso precisasse se tornar um simulador que pudesse receber mais de uma instrução por vez, só é necessário adicionar um laço de repetição e dependendo um vetor.
- Não houve o uso de define nem include para conseguir fazer a comparação na decodificação.