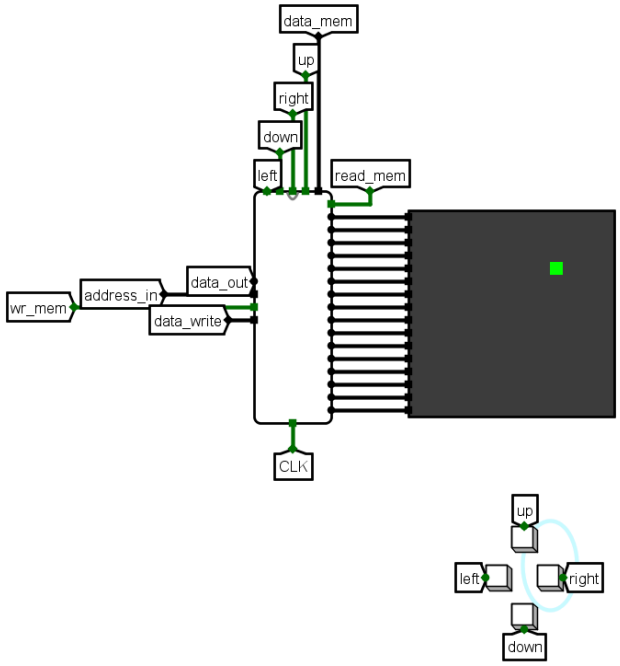

Snake Game

03/01/2026

prof. Ramon Santos Nepomuceno
UNIVERSIDADE FEDERAL DO CARIRI (UFCA) - CAMPUS JUAZEIRO DO NORTE

Visão geral

Este trabalho tem como objetivo integrar os conceitos de Arquitetura de Computadores, organização de processadores e programação em baixo nível por meio do desenvolvimento do jogo da cobrinha (Snake) utilizando um processador didático inspirado na arquitetura MIPS.

O projeto será desenvolvido sobre um processador já implementado no Logisim, fornecido pelo professor, que executa um subconjunto da arquitetura MIPS. A atividade envolve a extensão desse processador com novas instruções e o desenvolvimento do jogo da cobrinha em linguagem assembly compatível com essa arquitetura.

Processador Didático Disponibilizado

O processador fornecido já possui suporte aos seguintes formatos e instruções:

Formato - R:

opcode 6 bits	rs 5 bits	rt 5 bits	rd 5 bits	shamt 5 bits	funct 6 bits

As instruções já implementadas nesse formato são as seguintes:

INSTRUÇÃO	OPCODE	FUNCT	SIGNIFICADO
ADD rd,rs,rt	0	0	BR[rd] = BR[rs] + BR[rt]
SUB rd,rs,rt	0	1	BR[rd] = BR[rs] - BR[rt]
MULT rd,rs,rt	0	2	BR[rd] = BR[rs] * BR[rt]
DIV rd,rs,rt	0	3	BR[rd] = BR[rs] / BR[rt]

Formato-I:

opcode 6bits	rs 5 bits	rt 5bits	imediato 16 bits

Instruções já implementadas:

INSTRUÇÃO	OPCODE	SIGNIFICADO
ADDI rt, rs, Imediato	1	BR[rt]=BR[rs] + Imediato
SUBI rt, rs, Imediato	2	BR[rt]=BR[rs] - Imediato
MULTI rt, rs, Imediato	3	BR[rt]=BR[rs] * Imediato
DIVI rt, rs, Imediato	4	BR[rt]=BR[rs] / Imediato
LOAD rt, Imediato (rs)	6	BR[rt] = DATA_MEMORY[BR[rs] + Imediato]
STORE rt, Imediato (rs)	7	DATA_MEMORY[BR[rs] + Imediato] = BR[rt]

Formato - J:

opcode 6 bits	não usado 10 bits	label 16 bits

Instrução já implementada

INSTRUÇÃO	OPCODE	SIGNIFICADO
JUMP label	8	PC = label

Sugestões de Instruções a serem implementadas

- BEQ rs, rt, imediato (FORMATO I)
  Se BR[rs] == BR[rt], então PC = imediato, senão PC = PC + 1

- BNE rs, rt, imediato (FORMATO I)
  Se BR[rs] != BR[rt], então PC = imediato, senão PC = PC + 1

- SLT rd, rs, rt (Formato R)
  Se BR[rs] < BR[rt], então BR[rd] = 1, senão BR[rd] = 0

- JAL label (FORMATO J)
  BR[31] = PC + 1
  PC = Label

- JR rs (FORMATO R)
    - PC = BR[rs]

Essas instruções serão essenciais para a implementação de estruturas de controle, funções e decisões lógicas no jogo.

4. Interface Gráfica e Dispositivos de Entrada/Saída

O projeto já inclui uma interface gráfica implementada no Logisim, composta por:

4.1 Matriz de LEDs 16x16

- Mapeada a partir do endereço FF00
- Cada endereço representa um LED
- A escrita inicia no canto superior esquerdo
- Escrever valor diferente de zero acende o LED

4.2 Gerador de Números Aleatórios

- Leitura no endereço FEFE
- Cada leitura retorna um número pseudoaleatório entre 0 e 15

4.3 Leitura de Botões

- Leitura no endereço FEFF
- Valores possíveis:
    - 1 – Esquerda
    - 2 – Baixo
    - 4 – Direita
    - 8 – Cima

5. Código de Teste da Arquitetura

O seguinte código assembly será fornecido para testes iniciais da arquitetura e dos dispositivos de entrada e saída:

```
v2.0 raw
18010000
18020001
1803FEFF
00622002
```

00832000
1C81FF00
20000000

Esse código permite verificar:

- Funcionamento de LOAD e STORE
- Leitura de botões
- Escrita na matriz de LEDs
- Funcionamento do JUMP

6. Jogo da Cobrinha – Modelo em C

Será fornecido um código em linguagem C que representa a lógica básica do jogo da cobrinha. Esse código não deve ser traduzido linha a linha para assembly, mas utilizado como referência conceitual.

Analogias importantes:

- Variáveis globais em C correspondem a posições fixas na memória de dados.
- Estruturas de controle como if, while e switch devem ser implementadas com BEQ, BNE, JUMP e SLT.
- Funções em C podem ser implementadas usando JAL e JR.
- A matriz grid[16][16] não precisa existir explicitamente: a matriz de LEDs já representa visualmente o estado do jogo.
- A leitura de teclado em C corresponde à leitura do endereço FEFF.
- A função rand() corresponde à leitura do endereço FEFE.
- O conceito de delay em C pode ser implementado com laços de repetição em assembly.

7. Requisitos do Trabalho

- Implementar corretamente todas as instruções solicitadas no processador.
- Desenvolver o jogo da cobrinha em assembly.
- O jogo deve:
    - Movimentar o jogador corretamente
    - Ler entradas do usuário
    - Gerar comida em posições aleatórias
    - Atualizar a matriz de LEDs
- O código deve estar comentado e organizado.

8. Entregáveis

- Arquivo do Logisim com o processador modificado
- Código assembly do jogo da cobrinha

- Relatório explicando:
    - As instruções implementadas
    - As decisões de projeto
    - A lógica do jogo

9. Critérios de Avaliação

- Correta implementação das instruções (40%)
- Funcionamento do jogo (40%)
- Organização, clareza e documentação (20%)

.