Trabalho Prático 2
Implementação da Arquitetura "Lobo-Guará"
(Vetorial ou VLIW)

Objetivo:

O objetivo deste trabalho é projetar e implementar uma versão vetorial ou VLIW (Very Long Instruction Word) da arquitetura de computador de 8 bits "Lobo-Guará", utilizando a ferramenta de simulação **Logisim Evolution**. A arquitetura deve ser capaz de executar um conjunto específico de instruções e deve ser organizada de acordo com o modelo de processamento vetorial ou VLIW.

Descrição Geral das Arquiteturas "Guará Matilha" e "Guará Focinho Longo"

As arquiteturas "Guará Matilha" e "Guará Focinho Longo" são variantes da arquitetura "Lobo-Guará", cada uma com suas características específicas e conjunto de instruções.

Guará Matilha (Vector Architecture):

A arquitetura "Guará Matilha" é uma implementação vetorial da arquitetura "Lobo-Guará", projetada para processamento eficiente de operações em vetores de dados. Suas principais características incluem:

- LD (Load): Carrega um valor da memória para um registrador escalar.
- ST (Store): Armazena um valor de um registrador escalar na memória.
- MOVH (Move High): Move os bits de alta ordem de um valor imediato para um registrador escalar.
- MOVL (Move Low): Move os bits de baixa ordem de um valor imediato para um registrador escalar.
- Operações aritméticas e lógicas em registradores escalares e vetoriais.

Guará Focinho Longo (VLIW Architecture):

A arquitetura "Guará Focinho Longo" é uma implementação VLIW da arquitetura "Lobo-Guará", projetada para processamento de instruções em paralelo. Suas principais características incluem:

- BRZR (Branch On Zero Register): Desvia o fluxo de execução se um registrador for zero.
- BRZI (Branch On Zero Immediate): Desvia o fluxo de execução se um valor imediato for zero.
- JR (Jump Register): Salto para um endereço especificado por um registrador.
- JI (Jump Immediate): Salto para um endereço calculado a partir do endereço atual e um valor imediato.
- LD (Load): Carrega um valor da memória para um registrador.
- ST (Store): Armazena um valor de um registrador na memória.
- MOVH (Move High): Move os bits de alta ordem de um valor imediato para um registrador.
- MOVL (Move Low): Move os bits de baixa ordem de um valor imediato para um registrador.
- Operações aritméticas, lógicas e de deslocamento em registradores.

Ambas as arquiteturas "Guará Matilha" e "Guará Focinho Longo" oferecem desempenho e eficiência em suas respectivas abordagens de processamento vetorial e VLIW, adaptando-se às demandas de computação moderna.

Componentes a serem Implementados:

Os alunos devem implementar os seguintes componentes da arquitetura "Lobo-Guará" na versão vetorial ou VLIW no Logisim Evolution:

1. Diagrama de caixas do projeto do caminho de dados (datapath) do processador.
2. Diagrama de caixas do projeto interno da Unidade Lógico-Aritmética (ULA), com o código da ULA a ser usado para cada operação.

3. Tabela de sinais de controle para gerenciamento das etapas do pipeline.
4. Implementação do pipeline no Logisim Evolution, com os componentes e interconexões necessários.

Recomendações de Desenvolvimento:

Recomenda-se a seguinte ordem para o desenvolvimento do trabalho:

1. Desenvolvimento do diagrama de caixas do projeto do caminho de dados (datapath).
2. Desenvolvimento do diagrama de caixas do projeto interno da ULA.
3. Desenvolvimento da tabela de sinais de controle.
4. Teste das instruções de forma individual e de pequenos trechos de código para verificar o comportamento com múltiplas instruções.

Entrega:

O trabalho deverá ser entregue em duas partes, conforme detalhado a seguir:

1. **Parte 1 - Diagrama e Tabela (Peso: 50%):**
    - Diagrama em PDF contendo o diagrama da arquitetura, o projeto da ULA, a tabela de sinais de controle com eventuais detalhes do projeto.
2. **Parte 2 - Projeto no Logisim Evolution (Peso: 50%):**
    - Projeto completo da arquitetura na versão vetorial ou VLIW no formato Logisim Evolution, incluindo todos os componentes implementados e a interconexão entre eles.

Pontuação Adicional:

A implementação bem-sucedida do pipeline dará um ponto adicional à nota final do trabalho.

Critérios de Avaliação:

Os trabalhos serão avaliados com base nos seguintes critérios:

- Corretude da implementação da arquitetura vetorial ou VLIW.
- Organização e clareza dos diagramas e da tabela de sinais de controle.
- Funcionamento adequado do projeto no Logisim Evolution, incluindo o pipeline implementado.
- Cumprimento dos prazos de entrega.

Observações:

- Este trabalho deve ser realizado **individualmente**.
- Qualquer tentativa de plágio resultará em nota zero para todos os envolvidos.
- Em caso de dúvidas, entre em contato com o professor .

Os trabalhos deverão ser apresentados de forma oral pelo aluno. A nota irá considerar domínio do tema, robustez da solução e rigorosidade da metodologia.

Guará em Matilha (Vetorial)

Vector Architecture				
Opcode	**Tipo**	**Menemonico**	**Nome**	**Operação**
Scalar				
0000	R	ld	Load	SR[ra] = M[SR[rb]]
0001	R	st	Store	M[SR[rb]] = SR[ra]
0010	I	movh	Move High	SR[1] = {Imm., SR[1](3:0)}
0011	I	movl	Move Low	SR[1] = {SR[1](7:4), Imm.}
0100	R	add	Add	SR[ra] = SR[ra] + SR[rb]
0101	R	sub	Sub	SR[ra] = SR[ra] - SR[rb]
0110	R	and	And	SR[ra] = SR[ra] & SR[rb]
0111	R	brzr	Branch On Zero Register	if (SR[ra] == 0) PC = SR[rb]
Vector				
1000	R	ld	Load	VR[ra] = M[VR[rb]]
1001	R	st	Store	M[VR[rb]] = VR[ra]
1010	I	movh	Move High	VR[1] = {Imm., VR[1](3:0)}
1011	I	movl	Move Low	VR[1] = {VR[1](7:4), Imm.}
1100	R	add	Add	VR[ra] = VR[ra] + VR[rb]
1101	R	sub	Sub	VR[ra] = VR[ra] - VR[rb]
1110	R	and	And	VR[ra] = VR[ra] & VR[rb]
1111	R	or	Or	VR[ra] = VR[ra] | VR[rb]
			4x Vector PE	**Scalar PE**
SR -> Scalar register			4 Regs por PE (1º = ID, 3x GP)	4 Regs (1º = ZERO, 3x GP)
VR -> Vectorial register			VR0 = {0,1,2,3} dependendo do PE	SR0 = 0
			1 Memória por PE	1 Memória exclusiva
			Apenas 1 dos PEs atuam, ou VPE ou SPE	

Tipo R							
7	**6**	**5**	**4**	**3**	**2**	**1**	**0**
opcode				**Ra**		**Rb**	

Tipo I							
7	**6**	**5**	**4**	**3**	**2**	**1**	**0**
opcode				**Imm**			

Guará Focinho Longo (VLIW)

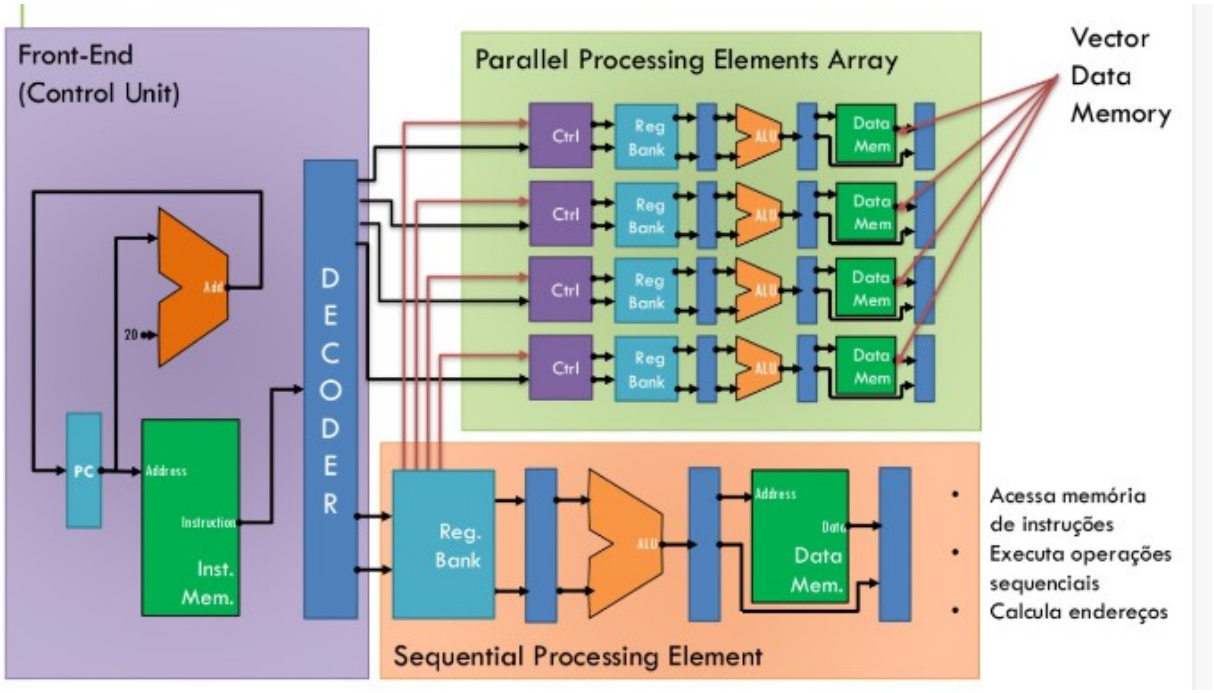
VLIW Architecture				
Opcode	**Tipo**	**Menemonico**	**Nome**	**Operação**
Controle				
0000	R	brzr	Branch On Zero Register	if (R[ra] == 0) PC = R[rb]
0001	I	brzi	Branch On Zero Immediate	if (R[0] == 0) PC = PC + Imm.
0010	R	jr	Jump Register	PC = R[rb]
0011	I	ji	Jump Immediate	PC = PC + Imm.
Dados				
0100	R	ld	Load	R[ra] = M[R[rb]]
0101	R	st	Store	M[R[rb]] = R[ra]
0110	I	movh	Move High	R[0] = {Imm., R[0](3:0)}
0111	I	movl	Move Low	R[0] = {R[0](7:4), Imm.}
Aritmética				
1000	R	add	Add	R[ra] = R[ra] + R[rb]
1001	R	sub	Sub	R[ra] = R[ra] - R[rb]
Lógica				
1010	R	and	And	R[ra] = R[ra] & R[rb]
1011	R	or	Or	R[ra] = R[ra] | R[rb]
1100	R	not	Not	R[ra] = ! R[rb]
1101	R	slr	Shift Left Register	R[ra] = R[ra] << R[rb]
1110	R	srr	Shift Right Register	R[ra] = R[ra] >> R[rb]
NOP = Free Slot				
1111	R	nop	No Operation	
			VLIW	**4 Lanes Fixos**
			4 Lanes = 8*4 bits	1º LD / ST / MOV
			4 Regs GP (General Purpose)	2º BR / JUMP
				3º ULA
				4º ULA

- A memória pode ser endereçada em 4 bytes para facilitar a obtenção dos dados.
- Podemos ter 1 controle por FU a fim de reduzir a complexidade do controle.
- Consideramos que não haverão instruções dependentes na mesma palavra e que o compilador não permitirá duas ou mais operações de escrita ao mesmo registrador na mesma instrução.

Detalhes específicos:

As figuras abaixo mostram datapaths de inspiração para o projeto do Lobo-Guará:

Guará em Matilha (Vetorial)



Guará Focinho Longo (VLIW)