CI112 Arquitetura de Computadores - 2023/1 - UFPR
Especificação do Trabalho 2

O aluno deverá desenvolver o projeto de implementação da arquitetura Sagui (Vetorial ou VLIW). Trata-se de uma arquitetura de 8-bits, do tipo load-store endereçada por byte, que foi desenvolvida especialmente para essa disciplina. A lista de instruções bem como seus formatos são dados nas próximas páginas.

O trabalho deve ser desenvolvido individualmente. Recomenda-se a seguinte ordem para o desenvolvimento do trabalho:

1. Desenvolvimento com diagrama de caixas do projeto do caminho de dados (datapath) do processador.
2. Desenvolvimento com diagrama de caixas do projeto interno da ULA, pensando no código da ULA a ser usado para cada operação.
3. Desenvolver uma tabela com sinais de controle
4. Desenvolvimento no Logisim Evolution do projeto com componentes e fios.
5. Teste das instruções de forma individual (testar cada instrução separadamente) e depois teste de pequeno código para ver o comportamento com múltiplas instruções.

Regras Gerais de Entrega e Apresentação

A implementação será feita no logisim evolution, onde poderão ser utilizados todos os componentes pré-prontos ali existentes. Cada aluno deverá entregar o diagrama do Sagui bem como o projeto no logisim evolution. A entrega será feita pelo Moodle dividida em duas partes

- Diagrama em PDF contendo o **diagrama do Sagui**, o **projeto da ULA, a tabela de sinais de controle** com eventuais detalhes do projeto.
- Projeto no formato **Logisim Evolution**

As datas limite de entrega serão sempre às 6h (a.m.) do dia, impreterivelmente. (não confundir com 18h)

Casos não tratados no enunciado deverão ser discutidos com o professor.
Os trabalhos devem ser feitos individualmente. **A cópia do trabalho (plágio), acarretará em nota igual a Zero para todos os envolvidos.**

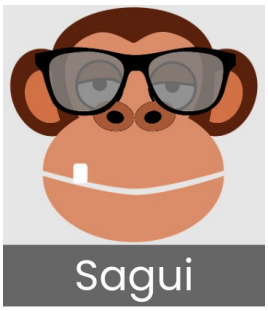
Os trabalhos deverão ser apresentados de forma oral pelo aluno. A nota irá considerar domínio do tema, robustez da solução e rigorosidade da metodologia.

Sagui em Bando (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**			



Sagui de Rabo Longo (VLIW)

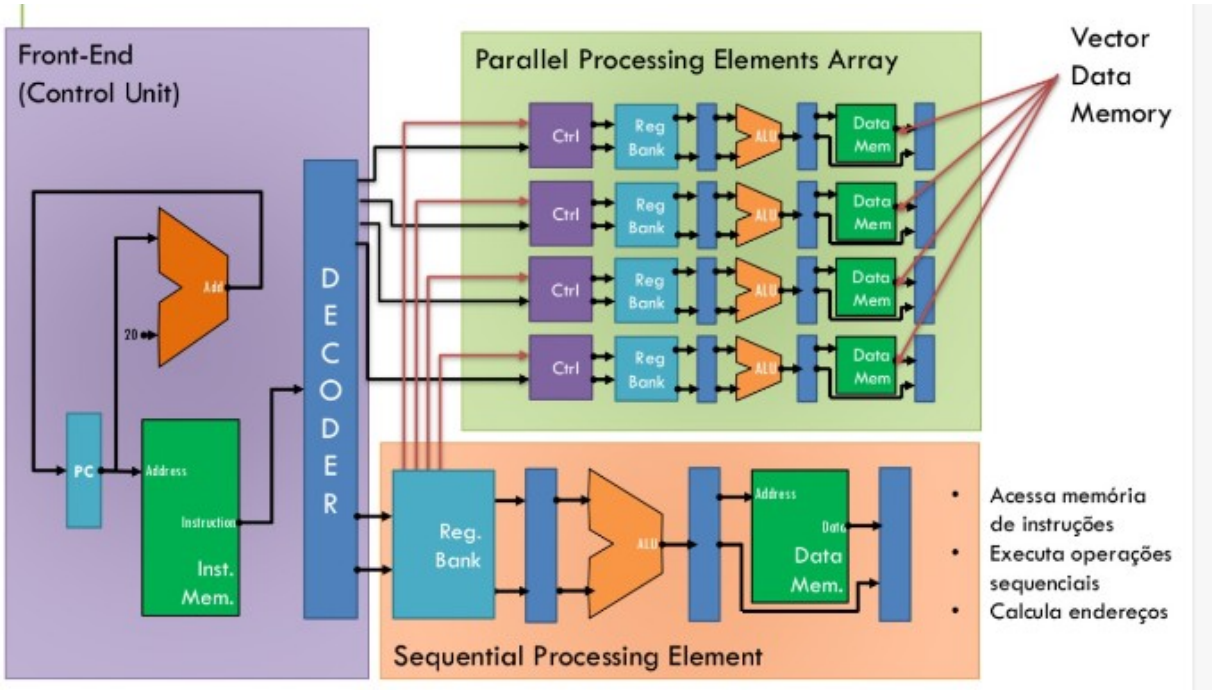
VLIW Architecture				
Opcode	**Tipo**	**Menemonico**	**Nome**	**Operação**
Controle				
0000	R	brzr	Branch On Zero Register	if (R[ra] == 0) PC = R[rb]
0001	I	brzi	Branch On Zero Immediate	if (R[0] == 0) PC = PC + Imm.
0010	R	jr	Jump Register	PC = R[rb]
0011	I	ji	Jump Immediate	PC = PC + Imm.
Dados				
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 Sagui:

Vetorial



VLIW