

Banco de Registradores e ALU

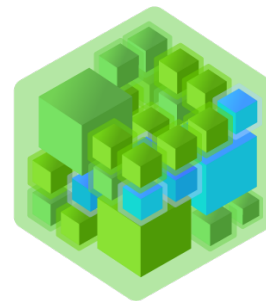
Prof. Alberto F. De Souza

LCAD/DI/UFES

sp1@lcad.inf.ufes.br

Apresentação baseada em:

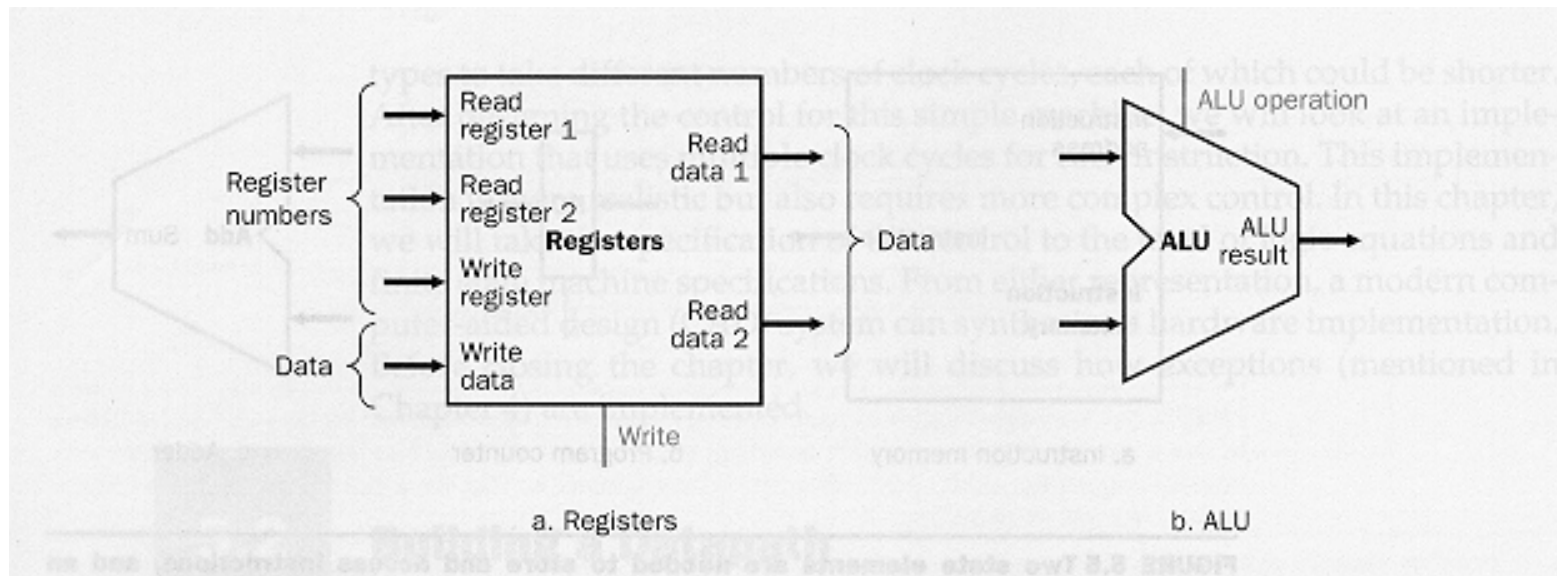
<http://www.cburch.com/logisim/docs/2.7/pt/html/guide/tutorial/index.html>



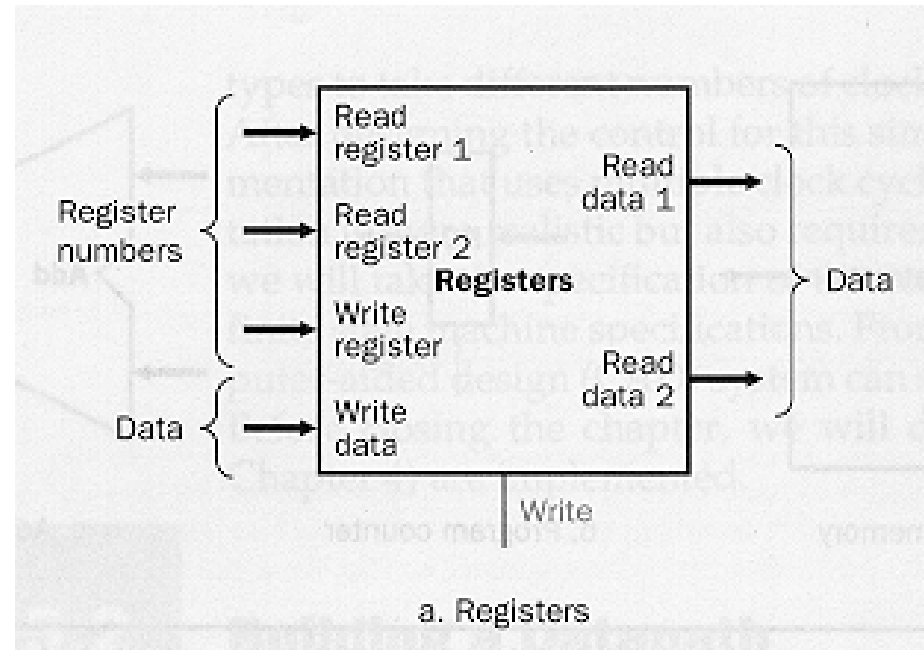
lcad

LABORATÓRIO DE COMPUTAÇÃO
DE ALTO DESEMPENHO

- Os componentes abaixo podem ser usados para compor a parte do datapath de um processador MIPS responsável por implementar a maioria das instruções lógicas e aritméticas:
 - add, sub, and, or, xor, etc.



- Vamos implementar o banco de registradores
- Ele contém 32 registradores de 32 bits
- Dois deles podem ser lidos simultaneamente
- Um pode ser escrito
- \$0 sempre é lido como zero





- No Logisim, um circuito menor que seja usado em outro maior é chamado de **subcircuito**
- Cada projeto Logisim é realmente uma biblioteca de circuitos e subcircuitos
- Em sua forma mais simples, cada projeto terá um único circuito (chamado "Principal" por padrão)
- Mas é fácil adicionar mais: basta selecionar Adicionar Circuito ... a partir do menu Project, e digitar qualquer nome válido
- Você poderá reaproveitar, então, o novo circuito que criar
- Vamos criar o circuito “Registers”
- Para projetar nosso Registers vamos precisar usar **cabos e distribuidores**

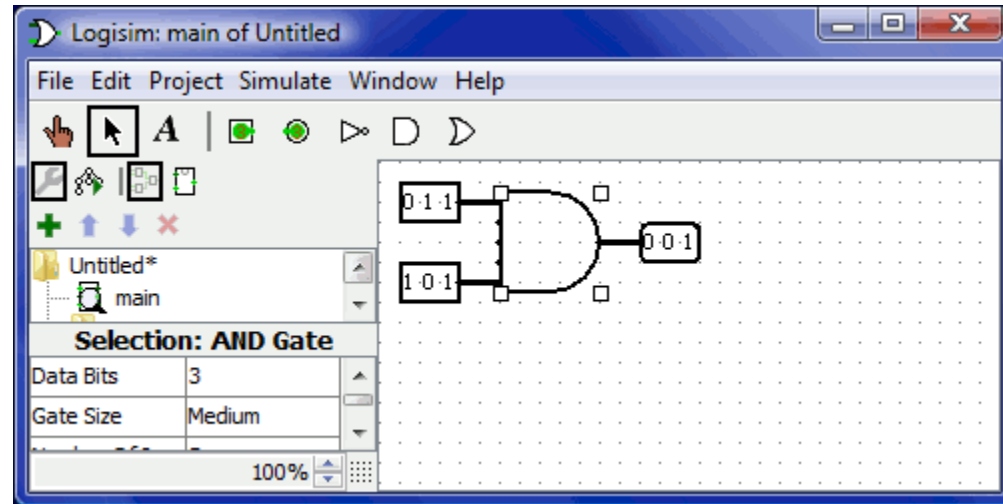
Logisim – *Para Criar Cabos*




lcad

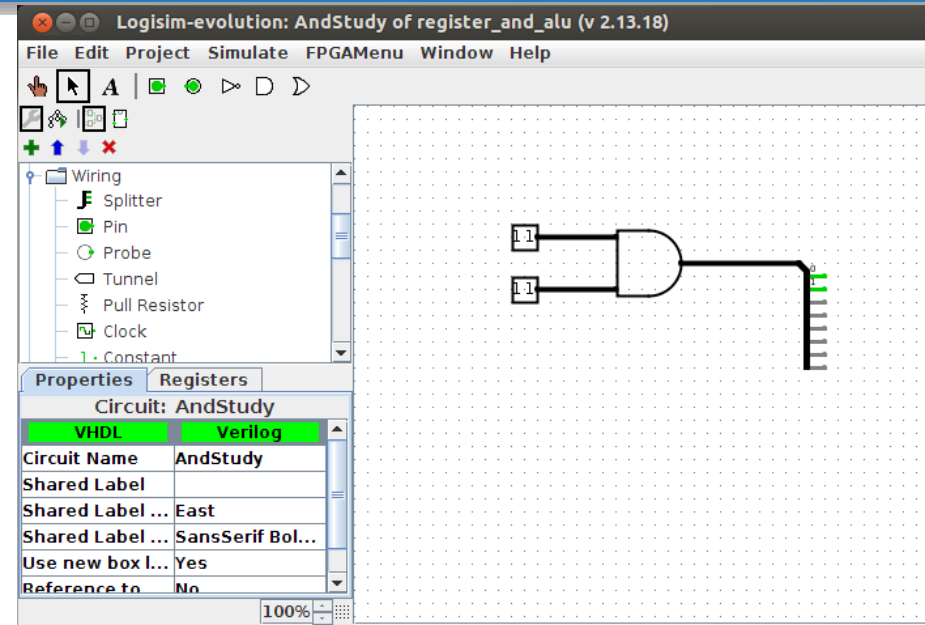


- No Logisim, cada entrada e saída em cada componente de um circuito tem uma largura de bits associada a ela
- Muitas vezes a largura de bits será 1
- Mas muitos dos componentes predefinidos do Logisim incluem o atributo de número de bits de suas entradas e saídas
- Você pode conectar com fios dois componentes com larguras de bits iguais. Mas, se um fio conectar dois componentes que exijam larguras diferentes, o Logisim irá reclamar que são “larguras incompatíveis”
- Para conexões de um único bit, é possível ver o valor no fio
- Conexões multi-bit podem ser examinadas clicando com a ferramenta Testar 🖱️ (Poke)



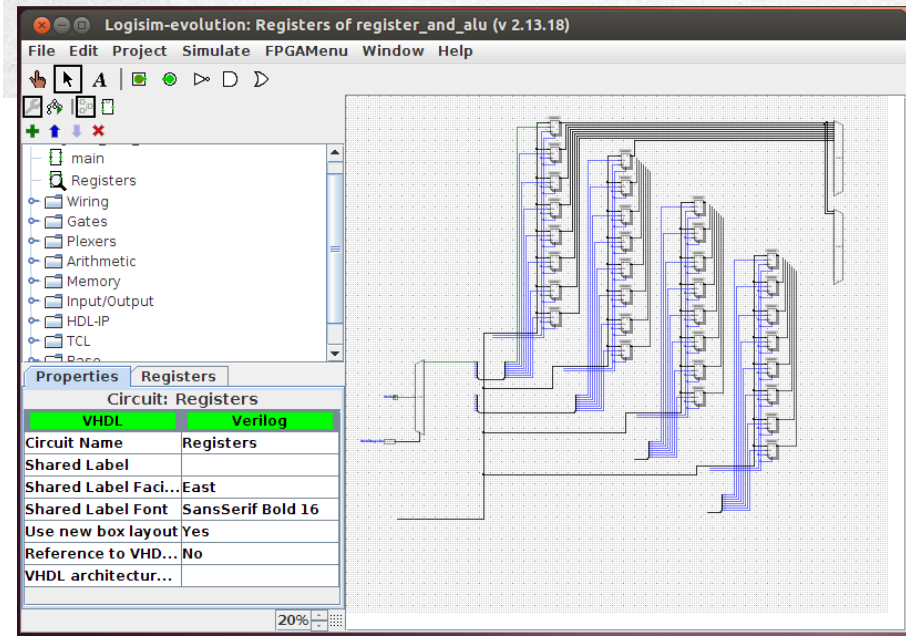
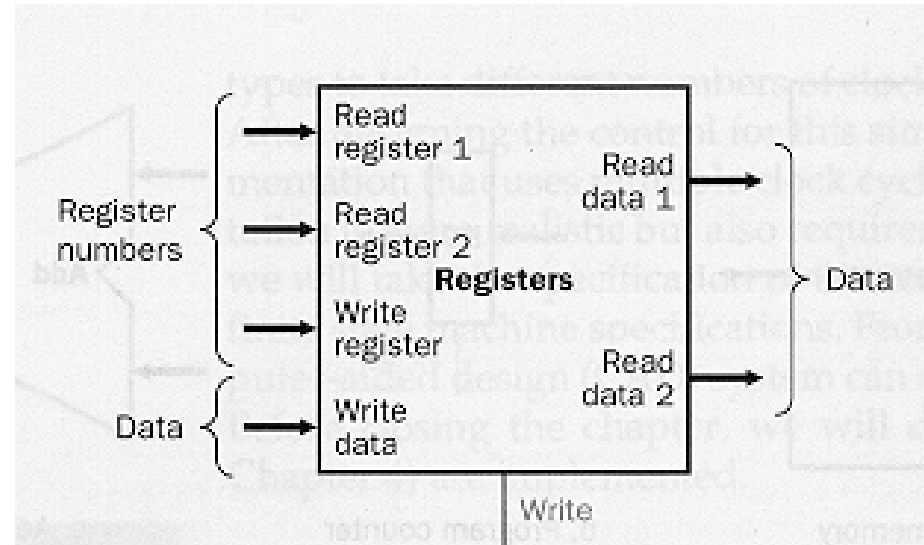


- Quando você trabalha com valores multi-bit, muitas vezes poderá querer rotear bits em direções diferentes
- A ferramenta Distribuidor (Splitter ) da biblioteca Wiring lhe permitirá fazer isso.
- A chave para o entendimento dos distribuidores são seus atributos
 - **Direção (Facing):** posição relativa das terminações
 - **Distribuição (Fan Out):** quantidade de bits de saída
 - **Largura em Bits à Entrada (Bit Width In):** quantidade de bits de entrada



Registers

- Vamos adicionar 32 registradores de 32 bits (lembre-se que o \$0 sempre contém zero)
- Precisamos de dois multiplexadores de saída, um para cada saída Read data
- E um demultiplexador para o Write register



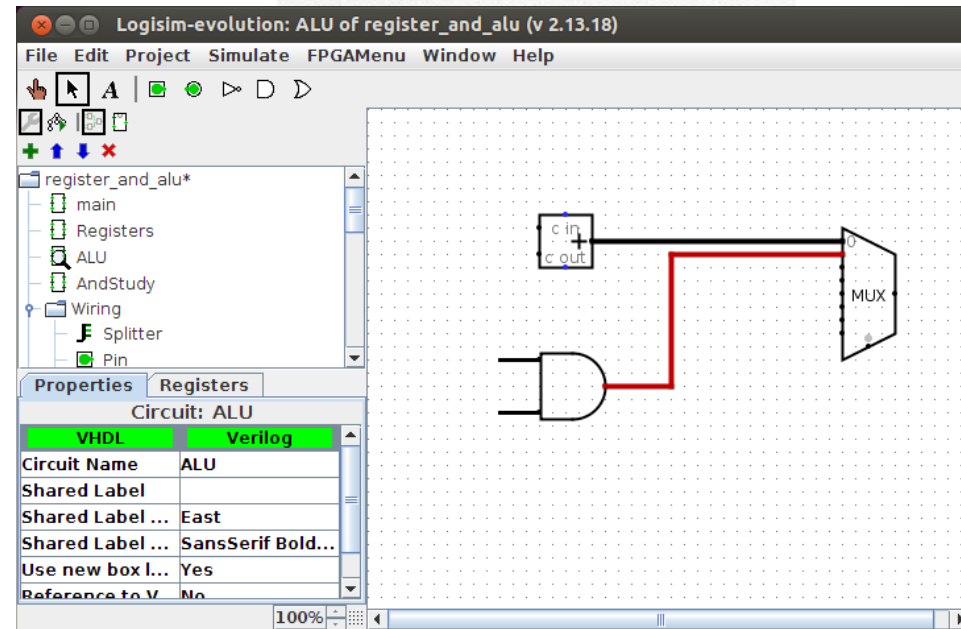
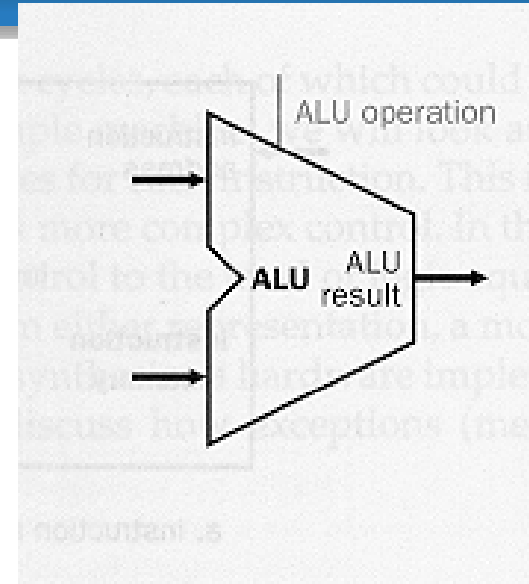
ALU



lcad



- Vamos implementar uma versão preliminar da ALU:
 - Crie um subcircuito chamado ALU
- Vamos fazer uma ALU de 32 bits capaz de somar, subtrair, e fazer operações lógicas
- Ela deve também computar um bit de Zero, que terá nível lógico 1 quando todos os 32 bits de ALUresult forem iguais a zero



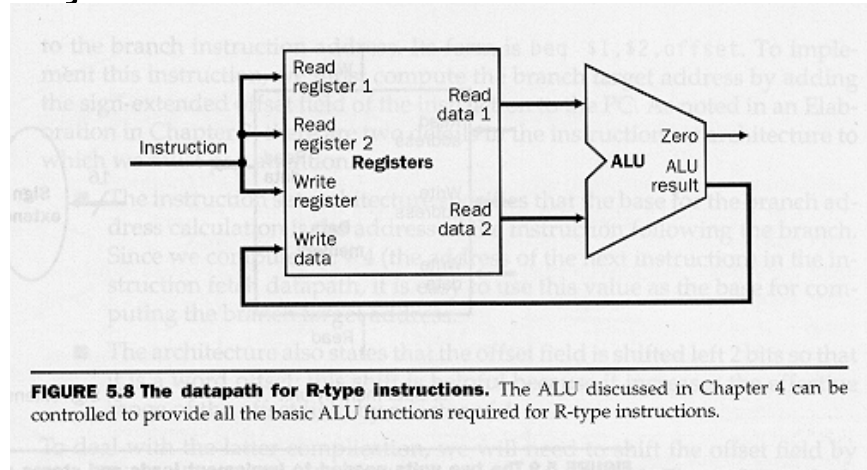
Trabalho 02



lcad



- Implementar o circuito abaixo de parte do datapath de um processador MIPS usando o Logisim (FAÇA SEU TRABALHO USANDO A VERSÃO CERTA DO LOGISIM)



- Instruction é uma instrução lógica ou aritmética MIPS (ligue os bits de acordo com instruções deste formato de instrução). Implemente uma ALU capaz de executar as instruções add, sub, and, or e nor, e a saída Zero
- Os trabalhos podem ser feitos em grupos de até 3 alunos e devem ser enviados para sp1@lcad.inf.ufes.br
- O e-mail deve conter o nome completo dos alunos do grupo

Category	Instruction	Example	Meaning	Comments
Arithmetic	add	add \$s1,\$s2,\$s3	$\$s1 = \$s2 + \$s3$	Three operands; overflow detected
	subtract	sub \$s1,\$s2,\$s3	$\$s1 = \$s2 - \$s3$	Three operands; overflow detected
	add immediate	addi \$s1,\$s2,100	$\$s1 = \$s2 + 100$	+ constant; overflow detected
	add unsigned	addu \$s1,\$s2,\$s3	$\$s1 = \$s2 + \$s3$	Three operands; overflow undetected
	subtract unsigned	subu \$s1,\$s2,\$s3	$\$s1 = \$s2 - \$s3$	Three operands; overflow undetected
	add immediate unsigned	addiu \$s1,\$s2,100	$\$s1 = \$s2 + 100$	+ constant; overflow undetected
	move from coprocessor register	mfc0 \$s1,\$epc	$\$s1 = \epc	Copy Exception PC + special regs
	multiply	mult \$s2,\$s3	$Hi, Lo = \$s2 \times \$s3$	64-bit signed product in Hi, Lo
	multiply unsigned	multu \$s2,\$s3	$Hi, Lo = \$s2 \times \$s3$	64-bit unsigned product in Hi, Lo
	divide	div \$s2,\$s3	$Lo = \$s2 / \$s3$, $Hi = \$s2 \bmod \$s3$	Lo = quotient, Hi = remainder
Data transfer	divide unsigned	divu \$s2,\$s3	$Lo = \$s2 / \$s3$, $Hi = \$s2 \bmod \$s3$	Unsigned quotient and remainder
	move from Hi	mfhi \$s1	$\$s1 = Hi$	Used to get copy of Hi
	move from Lo	mflo \$s1	$\$s1 = Lo$	Used to get copy of Lo
	load word	lw \$s1,100(\$s2)	$\$s1 = \text{Memory}[\$s2 + 100]$	Word from memory to register
	store word	sw \$s1,100(\$s2)	$\text{Memory}[\$s2 + 100] = \$s1$	Word from register to memory
	load half unsigned	lhu \$s1,100(\$s2)	$\$s1 = \text{Memory}[\$s2 + 100]$	Halfword memory to register
	store half	sh \$s1,100(\$s2)	$\text{Memory}[\$s2 + 100] = \$s1$	Halfword register to memory
	load byte unsigned	lbu \$s1,100(\$s2)	$\$s1 = \text{Memory}[\$s2 + 100]$	Byte from memory to register
Logical	store byte	sb \$s1,100(\$s2)	$\text{Memory}[\$s2 + 100] = \$s1$	Byte from register to memory
	load upper immediate	lui \$s1,100	$\$s1 = 100 * 2^{16}$	Loads constant in upper 16 bits
	and	and \$s1,\$s2,\$s3	$\$s1 = \$s2 \& \$s3$	Three reg. operands; bit-by-bit AND
	or	or \$s1,\$s2,\$s3	$\$s1 = \$s2 \$s3$	Three reg. operands; bit-by-bit OR
	nor	nor \$s1,\$s2,\$s3	$\$s1 = \sim (\$s2 \$s3)$	Three reg. operands; bit-by-bit NOR
	and immediate	andi \$s1,\$s2,100	$\$s1 = \$s2 \& 100$	Bit-by-bit AND with constant
	or immediate	ori \$s1,\$s2,100	$\$s1 = \$s2 100$	Bit-by-bit OR with constant
Conditional branch	shift left logical	sll \$s1,\$s2,10	$\$s1 = \$s2 \ll 10$	Shift left by constant
	shift right logical	srl \$s1,\$s2,10	$\$s1 = \$s2 \gg 10$	Shift right by constant
	branch on equal	beq \$s1,\$s2,25	if ($\$s1 == \$s2$) go to PC + 4 + 100	Equal test; PC-relative branch
	branch on not equal	bne \$s1,\$s2,25	if ($\$s1 != \$s2$) go to PC + 4 + 100	Not equal test; PC-relative
	set on less than	slt \$s1,\$s2,\$s3	if ($\$s2 < \$s3$) $\$s1 = 1$; else $\$s1 = 0$	Compare less than; two's complement
	set less than immediate	slti \$s1,\$s2,100	if ($\$s2 < 100$) $\$s1 = 1$; else $\$s1 = 0$	Compare < constant; two's complement
Unconditional jump	set less than unsigned	sltu \$s1,\$s2,\$s3	if ($\$s2 < \$s3$) $\$s1 = 1$; else $\$s1 = 0$	Compare less than; natural numbers
	set less than immediate unsigned	sltiu \$s1,\$s2,100	if ($\$s2 < 100$) $\$s1 = 1$; else $\$s1 = 0$	Compare < constant; natural numbers
	jump	j 2500	go to 10000	Jump to target address
Unconditional jump	jump register	jr \$ra	go to \$ra	For switch, procedure return
	jump and link	jal 2500	$\$ra = PC + 4$; go to 10000	For procedure call



lcad

