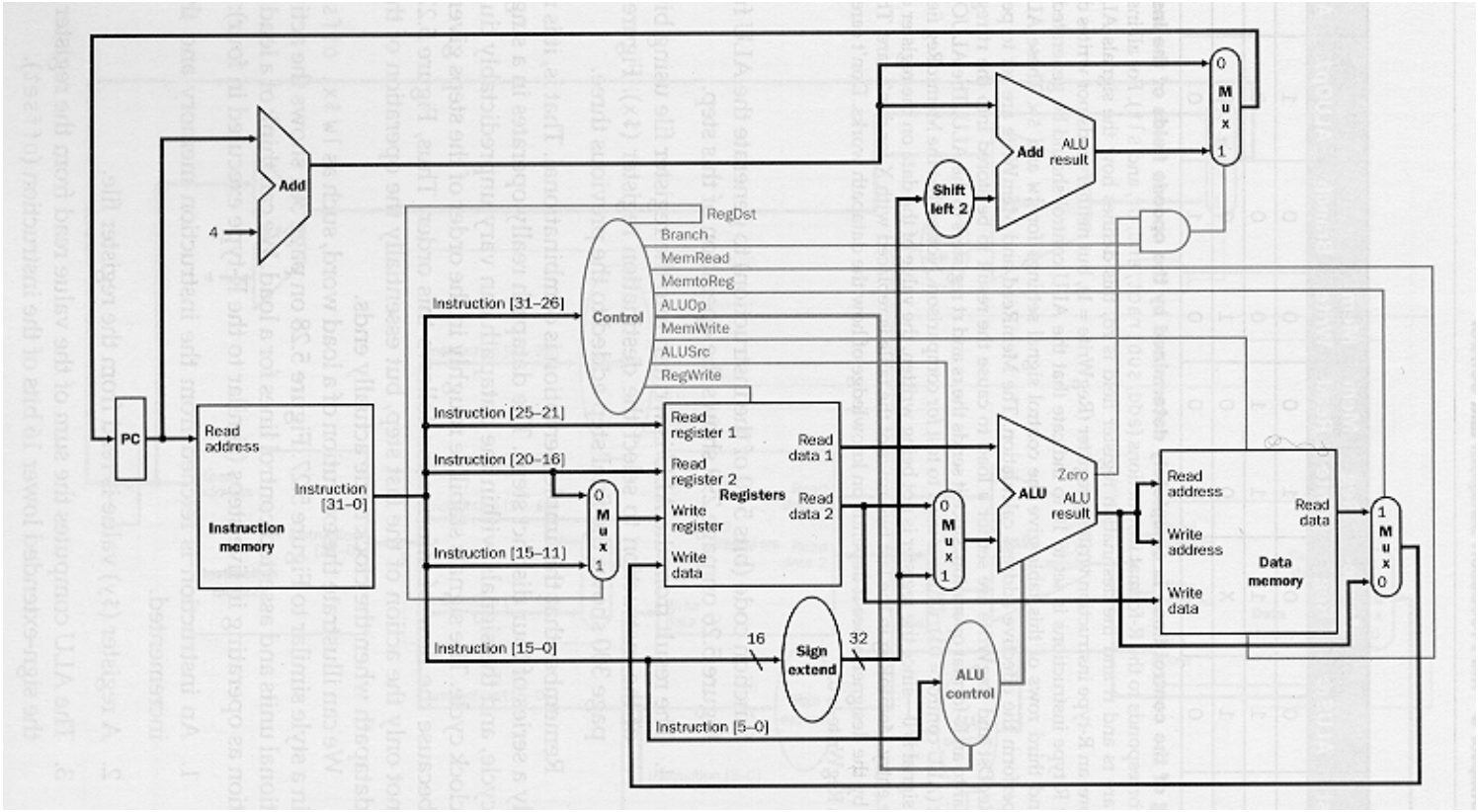
Unidade de Controle

Prof. Alberto F. De Souza
LCAD/DI/UFES
sp1@lcad.inf.ufes.br

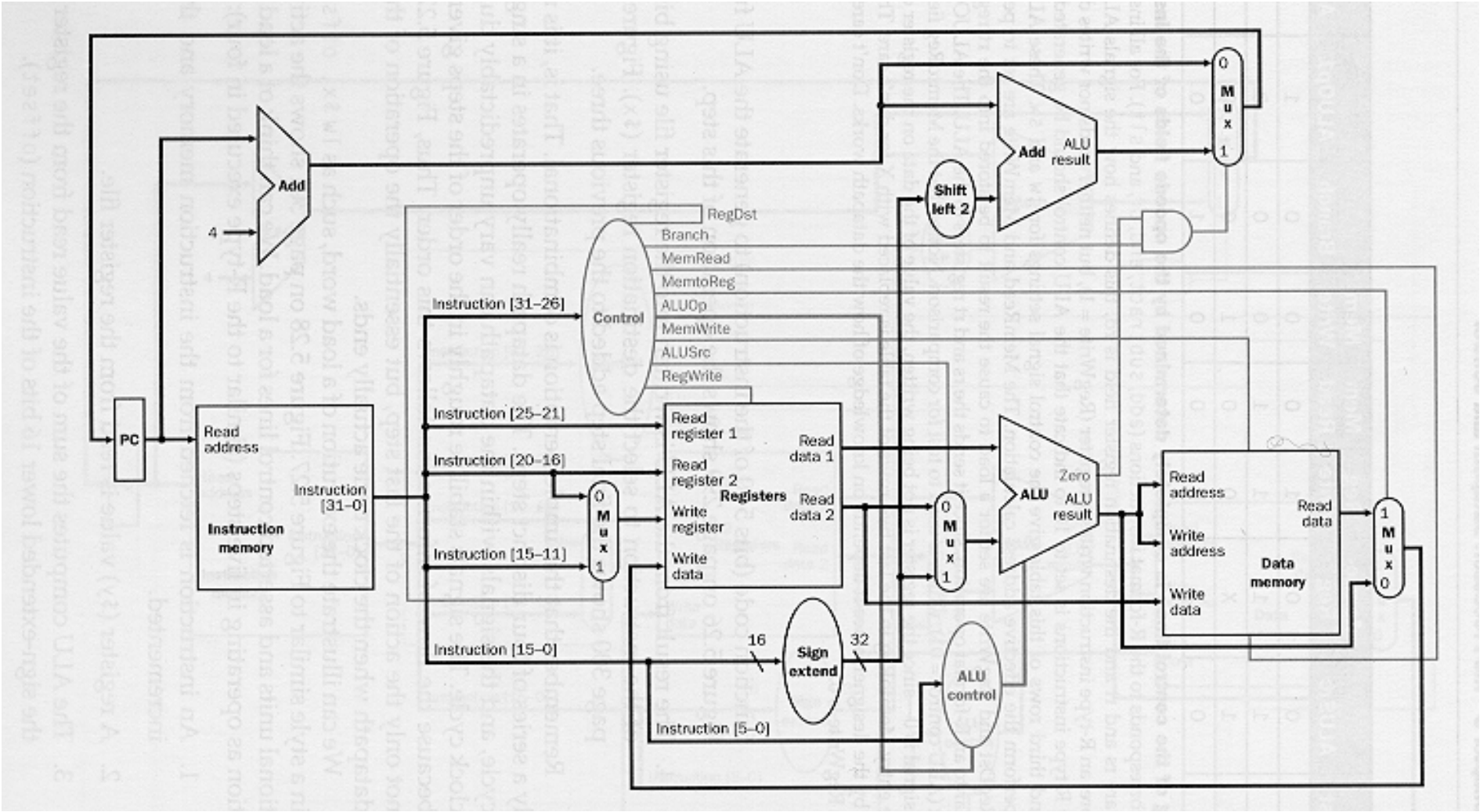
lcad
LABORATÓRIO DE COMPUTAÇÃO
DE ALTO DESEMPENHO

Unidade de Controle

- A unidade de controle gera sinais que, de acordo com a instrução corrente, controlam todos os elementos do processador

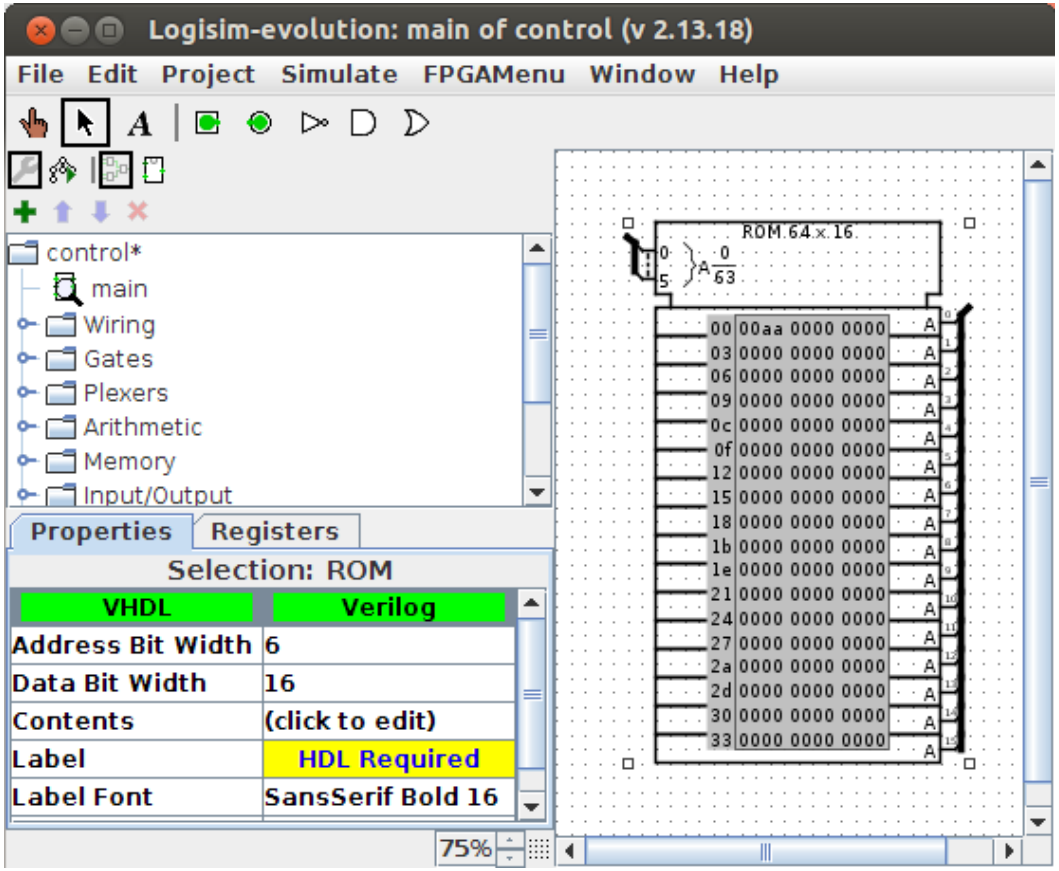


Unidade de Controle



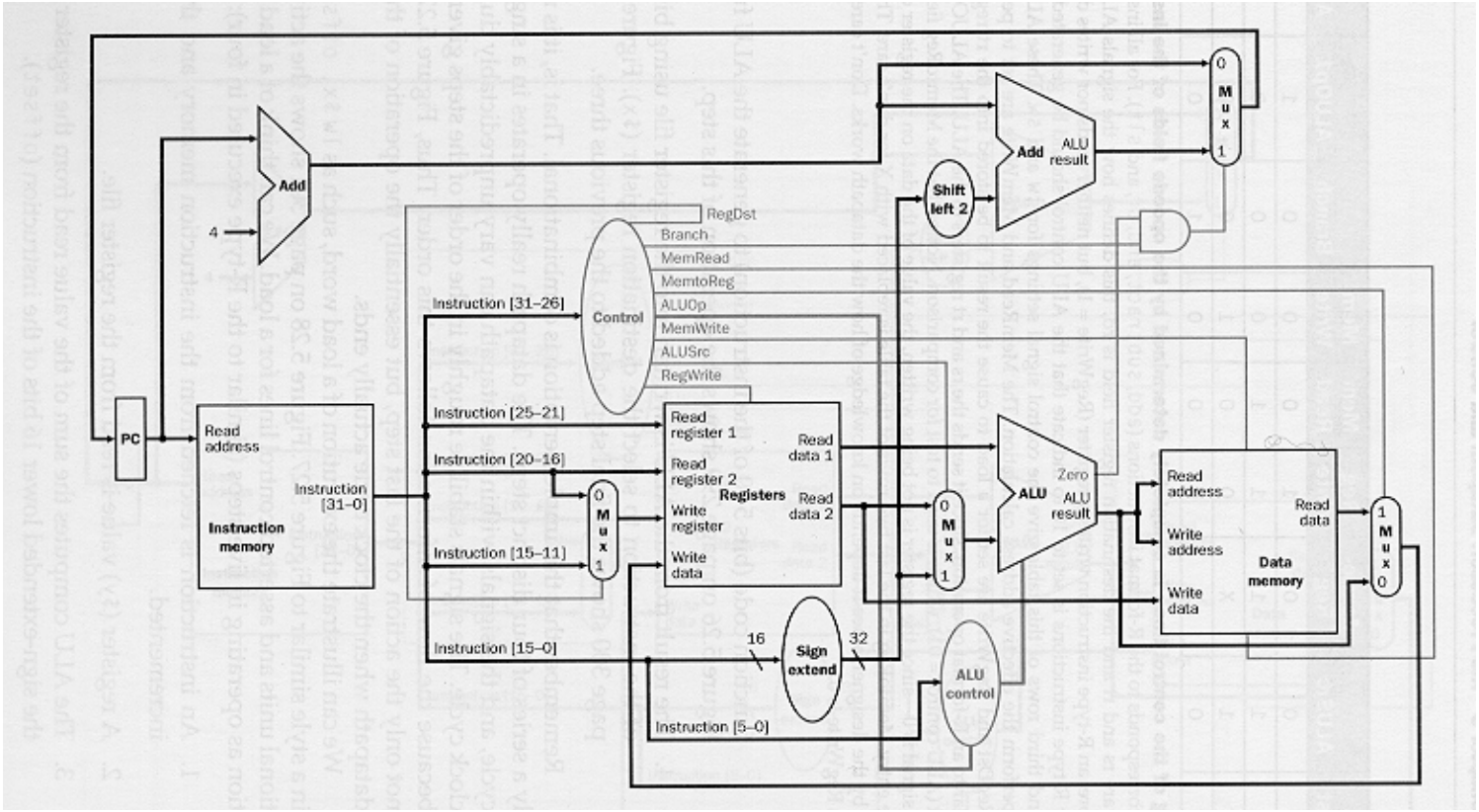
Implementação da Unidade de Controle

- Para implementar a unidade *Control* podemos usar uma memória ROM
- Neste caso, dado um endereço de 6 bits (Instruction[31-26]), o conteúdo correspondente da ROM seriam os bits que vão controlar o processador



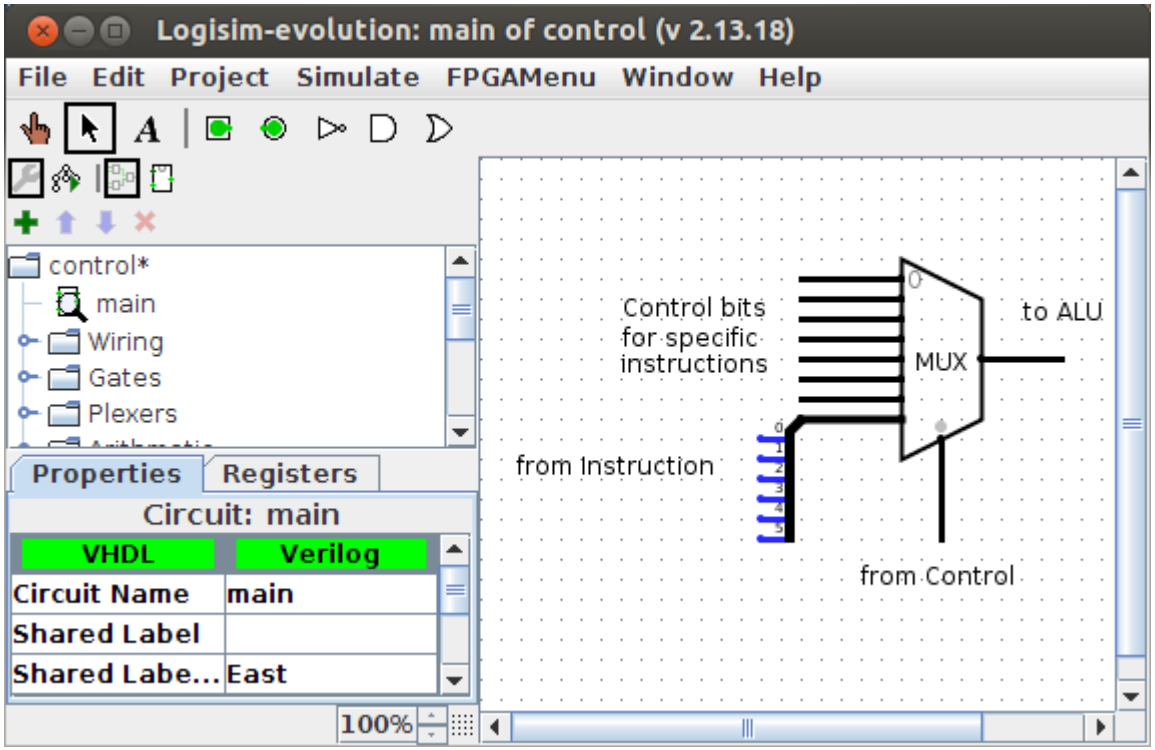
ALU control

- O módulo *ALU control* gera os sinais de controle da ALU a partir dos 6 bits de mais baixa ordem da instrução e de bits da unidade *Control*



Implementação de ALU control

- Para implementar *ALU control* podemos usa um multiplexador que, dependendo dos sinais de controle vindos de *Control*, seleciona os sinais que vão para a ALU



Trabalho 04

- Implemente os circuitos *Control* e *ALU control*, e preencha o conteúdo das entradas pertinentes da memória ROM de *Control* de modo a implementar (viabilizar a execução correta de) todas as instruções do próximo slide, com exceção das instruções: mfc0, as instruções em azul, e as instruções j, jr, jal, slt, slti, sltu, sltiu, sll, slr. As instruções sem sinal podem ser iguais às com sinal.
- Os formatos e códigos das instruções podem ser encontrados em: http://www.inf.ufes.br/~alberto/APENDICE-A.PDF
- Você pode usar o Simulador PCspim (http://spimsimulator.sourceforge.net/) para verificar os códigos e o funcionamento correto das instruções (note que este simulador as vezes troca uma instrução por outra equivalente)
- Os trabalhos podem ser feitos em grupos de até 3 alunos e devem ser enviados para sp1@lcad.inf.ufes.br
- **O email deve conter o nome completo dos alunos componentes 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 × $s3	64-bit signed product in Hi, Lo
	multiply unsigned	multu $s2,$s3	Hi, Lo = $s2 × $s3	64-bit unsigned product in Hi, Lo
	divide	div $s2,$s3	Lo = $s2 / $s3, Hi = $s2 mod $s3	Lo = quotient, Hi = remainder
	divide unsigned	divu $s2,$s3	Lo = $s2 / $s3, Hi = $s2 mod $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
Data transfer	load word	lw $s1,100($s2)	$s1 = Memory[$s2 + 100]	Word from memory to register
	store word	sw $s1,100($s2)	Memory[$s2 + 100] = $s1	Word from register to memory
	load half unsigned	lhu $s1,100($s2)	$s1 = Memory[$s2 + 100]	Halfword memory to register
	store half	sh $s1,100($s2)	Memory[$s2 + 100] = $s1	Halfword register to memory
	load byte unsigned	lbu $s1,100($s2)	$s1 = Memory[$s2 + 100]	Byte from memory to register
	store byte	sb $s1,100($s2)	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
Logical	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	$s1 = ~ ($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
	shift left logical	sll $s1,$s2,10	$s1 = $s2 << 10	Shift left by constant
	shift right logical	srl $s1,$s2,10	$s1 = $s2 >> 10	Shift right by constant
Condi-tional branch	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
	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
Uncondi-tional jump	jump	j 2500	go to 10000	Jump to target address
	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