decode tables

Fibonacci.asm

line number	program	Rgdst	Branch	MemToReg	ALUsrc	MemWrite	RegWrite	ALUCtrl	Jump	rt	rs	rd
	fibonacci10.asm	picks Rd vs Rt	Are we branching?	writing memory or ALU output to register?	determines 2nd input to ALU - imm or reg?	Write to memory or not?	Write to register?	ALU operation select	Are we jumping			
1	li $s0, 2 # $s0 is current index of the sequence	0	0	0	1	0	1	add	0	0	0	0
2	li $s1, 0 # $s1 is the 1st value of our sequence	0	0	0	1	0	1	add	0	0	0	1
3	li $s2, 1 # $s2 is the 2nd value of our sequence	0	0	0	1	0	1	add	0	0	0	2
4	li $s4, 0 # $s4 is where we hold our current index value	0	0	0	1	0	1	add	0	0	0	4
									0		0	
5	Loop: beq $s0, 10, End # if index > 10, jump to 'End'	0	1	0	1	0	0	subtract	0	0	0	0
6	add $s4, $s1, $s2 # current index value is the sum of the previous two index values	0	0	0	0	0	1	add	0	2	1	4
7	move $s1, $s2 # update the n-2 index value to be previous n-1 value	0	0	0	1	0	1	add	0		index	rs
8	move $s2, $s4 # update the n-1 index value to be previous n value	0	0	0	1	0	1	add	0		index	rs
9	addi $s0, $s0, 1 # increment index by 1	0	0	0	1	0	1	add	0	DC	0	0
10	j Loop # jump back to 'Loop'	DC	DC	DC	DC	DC	DC	DC	1			
11	End: move $s5, $s4 # move the final odd sum to $s3	0	0	0	1	0	1	add	0		index	rs

decode tables

Fibonacci.asm

line number	program	Rgdst	Branch	MemToReg	ALUsrc	MemWrite	RegWrite	ALUCtrl	Jump	rt	rs	rd
	oddeven100.asm	picks Rd vs Rt	Are we branching?	writing memory or ALU output to register?	determines 2nd input to ALU - imm or reg?	Write to memory or not?	Write to register?	ALU operation select	Are we jumping			
	li $s0, 1 # $s0 is our index = 1	0	0	0	1	0	1	add	0			
	li $s1, 0 # $s1 is our odd sum = 0	0	0	0	1	0	1	add	0			
	li $s2, 0 # $s2 is our even sum = 0	0	0	0	1	0	1	add	0			
	li $t0, 2 # $t0 is our divider to see if number is odd or even	0	0	0	1	0	1	add	0			
	Loop: beq $s0, 101, End # if index > 100, jump to 'End'	0	1	0	1	0	0	subtract	0			
	rem $s4, $s0, $t0 # remainder of $s0/$t0 is stored in $s4	0	0	0	0	0	1	opcode	0			
	beq $s4, 0, Even # if the remainder = 0 then jump to 'Even'	0	1	0	1	0	0	subtract	0			
	add $s1, $s1, $s0 # if $s0 is odd we add to $s1	0	0	0	0	0	1	add	0			
	addi $s0, $s0, 1 # increment index by 1	1	0	0	1	0	1	add	0			
	j Loop # jump back to 'Loop'	DC	DC	DC	DC	DC	DC	DC	1			
	Even: add $s2, $s2, $s0 # if $s0 is even we add to $s2	0	0	0	0	0	1	add	0			
	addi $s0, $s0, 1 # increment index by 1											
	j Loop # jump back to 'Loop'	DC	DC	DC	DC	DC	DC	DC	1			
	End: move $s6, $s1 # move the final odd sum to $s6	0	0	0	1	0	1	add	0			
	move $s7, $s2 # move the final even sum to $s7	0	0	0	1	0	1	add	0			