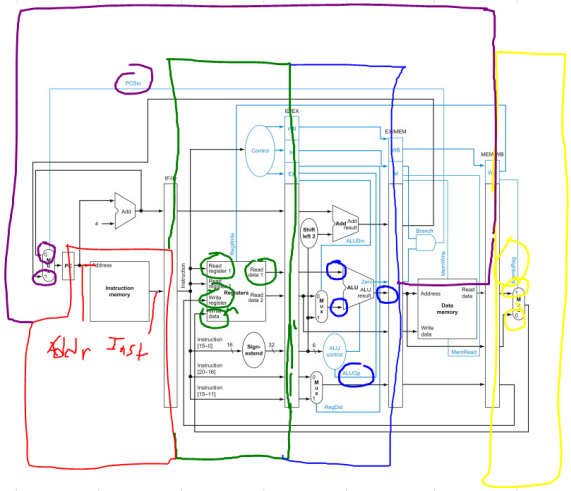
Cycle	Addr	Instr	Read reg 1	Read data 1	Write reg	Write data	instr (JOE add)	A	B	Op	ALU result	instr(JOE add)	1	0	s	instr(JOE add)	1	0	s	instr(JOE add)
1	0x00000010	addi	x	x	x	x	NONE	x	x	x	x		x	x	x		x	0x00000014	0	addi
2	0x00000014	lui	0x00	0x00000000	x	x	addi $t7, $zero, 63	x	x	x	x		x	x	x		x	0x00000018	0	lui
3	0x00000018	xor	0x00	0x00000000	x	x	lui $s7, 0x1010	0x00000000	0x0000003f	addi	0x0000003F	addi $t7, $zero, 63	x	x	x		x	0x0000001c	0	xor
4	0x0000001c	sub	0x06 ($a2)	0x000003FF	x	x	xor $t5, $a2, $a3	0x00000000	0x10100000	lui	0x10100000	lui $s7, 0x1010	x	x	x		x	0x00000020	0	sub
5	0x00000020	sll	0x04 ($a0)	0x00000003	0x0F ($t7)	0x0000003F	sub $t6 $a0, $a1	0x000003ff	0xFFFFFFFF	xor	0xFFFFFC00	xor $t5, $a2, $a3	x	0x0000003F	0	addi $t7, $zero, 63	x	0x00000024	0	sll
6	0x00000024	ori	0x00	0x00000000	0x17 ($s7)	0x10100000	sll $zero, $zero, 0	0x00000003	0x00000400	sub	0xFFFFFC03	sub $t6 $a0, $a1	x	0x10100000	0	lui $s7, 0x1010	x	0x00000028	0	ori
7	0x00000028	beq	0x17	0x10100000	0x0D ($t5)	0xFFFFFC00	ori $s7, $s7, 0x0040	0x00000000	0x00000000	sll	0x00000000	sll $zero, $zero, 0	x	0xFFFFFC00	0	xor $t5, $a2, $a3	x	0x0000002C	0	beq
8	0x0000002C	sll	0x0F ($t7)	0x0000003F	0x0E ($t6)	0xFFFFFC03	beq $t7, $a3, Exit	0x10100000	0x00000040	ori	0x10100040	ori $s7, $s7, 0x0040	x	0xFFFFFC03	0	sub $t6 $a0, $a1	x	0x00000030	0	sll
9	0x00000030	sll	0x00	0x00000000	0x00	0x00000000	sll $zero, $zero, 0	0x0000003F	0xFFFFFFFF	beq	0x00000040	beq $t7, $a3, Exit	x	0x00000000	0	sll $zero, $zero, 0	x	0x00000034	0	sll
10	0x00000034	sll	0x00	0x00000000	0x17($s7)	0x00000038	sll $zero, $zero, 0	0x00000000	0x00000000	sll	0x00000000	sll $zero, $zero, 0	x	0x10100040	0	ori $s7, $s7, 0x0040	x	0x00000038	0	sll
11	0x00000038	sw	0x00	0x00000000	x	x	sll $zero, $zero, 0	0x00000000	0x00000000	sll	0x00000000	sll $zero, $zero, 0	x	0x00000040	0	beq $t7, $a3, Exit	x	0x00000040	0	sw
12	0x00000040	none	0x17 ($s7)	0x00000038	0x00	0x00000000	sw $t7, 0($s7)	0x00000000	0x00000000	sll	0x00000000	sll $zero, $zero, 0	x	0x00000000	0	sll $zero, $zero, 0	x	0x00000044	0	none
13	0x00000044	none	x	x	0x00	0x00000000	none	0x00000038	0x00000000	sw	0x00000038	sw $t7, 0($s7)	x	0x00000000	0	sll $zero, $zero, 0	x	0x00000048	0	none
14	0x00000048	none	x	x	0x00	0x00000000	none	x	x	none	x	none	x	0x00000000	0	sll $zero, $zero, 0	x	0x0000004C	0	none
15	0x0000004C	none	x	x	x	x	none	x	x	none	x	none	x	x	x	sw $t7, 0($s7)	x	0x00000050	0	none

	INSTRUCTION MEM		REG FILE					ALU					MEMtoReg MUX				PCSrc MUX			

addi $t7, $zero, 63
lui $s7, 0x1010
xor $t5, $a2, $a3
sub $t6 $a0, $a1
sll $zero, $zero, 0
ori $s7, $s7, 0x0040
beq $t7, $a3, Exit
sll $zero, $zero, 0
sll $zero, $zero, 0
sll $zero, $zero, 0
sw $t7, 0($s7)

THERE IS ONE MORE CLOCK CYCLE FROM ALU TO MEMtoREG MUX
*Assuming negative edge MIPSregfile



#Assume that $a0 = 3, $a1 = 1024, $a2 = 1023, $a3 = -1
at the start of your manual simulation.
Assume that lui is supported by the lui operation in
the ALU and that the value shifted for lui is the B
input of the ALU (note that this is likely different
than your project implementation and that's OK).
The following instructions start at address 0x00000010: