Mark Gameng

Lab 2

Step 5

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Register Number | Register Name | Before sll | After sll | Changed |
| PC | PC | 0x0040001c | 0x00400020 | **Yes** |
| 0 | $r0 | 0x00000000 | 0x00000000 | No |
| 1 | $at | 0x10000000 | 0x10000000 | No |
| 2 | $v0 | 0x00000004 | 0x00000004 | No |
| 3 | $v1 | 0x00000000 | 0x00000000 | No |
| 4 | $a0 | 0x00000000 | 0x00000000 | No |
| 5 | $a1 | 0x00000000 | 0x00000000 | No |
| 6 | $a2 | 0x00000000 | 0x00000000 | No |
| 7 | $a3 | 0x00000000 | 0x00000000 | No |
| 8 | $t0 | 0x00000004 | 0x00000004 | No |
| 9 | $t1 | 0x00000000 | 0x00000010 | **Yes** |
| 10 | $t2 | 0x00000000 | 0x00000000 | No |
| 11 | $t3 | 0x00000000 | 0x00000000 | No |
| 12 | $t4 | 0x00000000 | 0x00000000 | No |
| 13 | $t5 | 0x00000000 | 0x00000000 | No |
| 14 | $t6 | 0x00000000 | 0x00000000 | No |
| 15 | $t7 | 0x00000000 | 0x00000000 | No |
| 16 | $s0 | 0x00000000 | 0x00000000 | No |
| 17 | $s1 | 0x00000000 | 0x00000000 | No |
| 18 | $s2 | 0x00000000 | 0x00000000 | No |
| 19 | $s3 | 0x00000000 | 0x00000000 | No |
| 20 | $s4 | 0x00000000 | 0x00000000 | No |
| 21 | $s5 | 0x00000000 | 0x00000000 | No |
| 22 | $s6 | 0x00000000 | 0x00000000 | No |
| 23 | $s7 | 0x00000000 | 0x00000000 | No |
| 24 | $t8 | 0x00000000 | 0x00000000 | No |
| 25 | $t9 | 0x00000000 | 0x00000000 | No |
| 26 | $k0 | 0x00000000 | 0x00000000 | No |
| 27 | $k1 | 0x00000000 | 0x00000000 | No |
| 28 | $gp | 0x10008000 | 0x10008000 | No |
| 29 | $sp | 0x7fffeffc | 0x7fffeffc | No |
| 30 | $fp | 0x00000000 | 0x00000000 | No |
| 31 | $ra | 0x00400018 | 0x00400018 | No |

sll is shift left logical and it shifts the bits to the left. For example, sll **$t1, $t0, 2**, shifts **$t0** to the left **2** times and assigns it to **$t1**.

Step 7

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Register Number | Register Name | Before lui | After lui | Changed |
| PC | PC | 0x00400008 | 0x0040000c | **Yes** |
| 0 | $r0 | 0x00000000 | 0x00000000 | No |
| 1 | $at | 0x00000000 | 0x10000000 | **Yes** |
| 2 | $v0 | 0x00000004 | 0x00000004 | No |
| 3 | $v1 | 0x00000000 | 0x00000000 | No |
| 4 | $a0 | 0x00000000 | 0x00000000 | No |
| 5 | $a1 | 0x00000000 | 0x00000000 | No |
| 6 | $a2 | 0x00000000 | 0x00000000 | No |
| 7 | $a3 | 0x00000000 | 0x00000000 | No |
| 8 | $t0 | 0x00000000 | 0x00000000 | No |
| 9 | $t1 | 0x00000000 | 0x00000000 | No |
| 10 | $t2 | 0x00000000 | 0x00000000 | No |
| 11 | $t3 | 0x00000000 | 0x00000000 | No |
| 12 | $t4 | 0x00000000 | 0x00000000 | No |
| 13 | $t5 | 0x00000000 | 0x00000000 | No |
| 14 | $t6 | 0x00000000 | 0x00000000 | No |
| 15 | $t7 | 0x00000000 | 0x00000000 | No |
| 16 | $s0 | 0x00000000 | 0x00000000 | No |
| 17 | $s1 | 0x00000000 | 0x00000000 | No |
| 18 | $s2 | 0x00000000 | 0x00000000 | No |
| 19 | $s3 | 0x00000000 | 0x00000000 | No |
| 20 | $s4 | 0x00000000 | 0x00000000 | No |
| 21 | $s5 | 0x00000000 | 0x00000000 | No |
| 22 | $s6 | 0x00000000 | 0x00000000 | No |
| 23 | $s7 | 0x00000000 | 0x00000000 | No |
| 24 | $t8 | 0x00000000 | 0x00000000 | No |
| 25 | $t9 | 0x00000000 | 0x00000000 | No |
| 26 | $k0 | 0x00000000 | 0x00000000 | No |
| 27 | $k1 | 0x00000000 | 0x00000000 | No |
| 28 | $gp | 0x10008000 | 0x10008000 | No |
| 29 | $sp | 0x7fffeffc | 0x7fffeffc | No |
| 30 | $fp | 0x00000000 | 0x00000000 | No |
| 31 | $ra | 0x00000000 | 0x00000000 | No |

Lui is load upper immediate. Lui **$1, a** ; the **a** value is shifted left 16 bits and is stored in **$1**, and the lower 16 bits are zeroes.

Step 8

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Register Number | Register Name | Before jal | After jal | Changed |
| PC | PC | 0x00400014 | 0x00400034 | **Yes** |
| 0 | $r0 | 0x00000000 | 0x00000000 | No |
| 1 | $at | 0x10000000 | 0x10000000 | No |
| 2 | $v0 | 0x00000004 | 0x00000004 | No |
| 3 | $v1 | 0x00000000 | 0x00000000 | No |
| 4 | $a0 | 0x10000000 | 0x10000000 | No |
| 5 | $a1 | 0x00000000 | 0x00000000 | No |
| 6 | $a2 | 0x00000000 | 0x00000000 | No |
| 7 | $a3 | 0x00000000 | 0x00000000 | No |
| 8 | $t0 | 0x00000000 | 0x00000000 | No |
| 9 | $t1 | 0x00000000 | 0x00000000 | No |
| 10 | $t2 | 0x00000000 | 0x00000000 | No |
| 11 | $t3 | 0x00000000 | 0x00000000 | No |
| 12 | $t4 | 0x00000000 | 0x00000000 | No |
| 13 | $t5 | 0x00000000 | 0x00000000 | No |
| 14 | $t6 | 0x00000000 | 0x00000000 | No |
| 15 | $t7 | 0x00000000 | 0x00000000 | No |
| 16 | $s0 | 0x00000000 | 0x00000000 | No |
| 17 | $s1 | 0x00000000 | 0x00000000 | No |
| 18 | $s2 | 0x00000000 | 0x00000000 | No |
| 19 | $s3 | 0x00000000 | 0x00000000 | No |
| 20 | $s4 | 0x00000000 | 0x00000000 | No |
| 21 | $s5 | 0x00000000 | 0x00000000 | No |
| 22 | $s6 | 0x00000000 | 0x00000000 | No |
| 23 | $s7 | 0x00000000 | 0x00000000 | No |
| 24 | $t8 | 0x00000000 | 0x00000000 | No |
| 25 | $t9 | 0x00000000 | 0x00000000 | No |
| 26 | $k0 | 0x00000000 | 0x00000000 | No |
| 27 | $k1 | 0x00000000 | 0x00000000 | No |
| 28 | $gp | 0x10008000 | 0x10008000 | No |
| 29 | $sp | 0x7fffeffc | 0x7fffeffc | No |
| 30 | $fp | 0x00000000 | 0x00000000 | No |
| 31 | $ra | 0x00000000 | 0x00400018 | **yes** |

jal **x**, copies the address of the next instruction into **$ra** and jumps to the address **x**

Step 9

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Register Number | Register Name | Before jr | After jr | Changed |
| PC | PC | 0x0040003c | 0x00400018 | **Yes** |
| 0 | $r0 | 0x00000000 | 0x00000000 | No |
| 1 | $at | 0x10000000 | 0x10000000 | No |
| 2 | $v0 | 0x00000004 | 0x00000004 | No |
| 3 | $v1 | 0x00000000 | 0x00000000 | No |
| 4 | $a0 | 0x10000000 | 0x10000000 | No |
| 5 | $a1 | 0x00000000 | 0x00000000 | No |
| 6 | $a2 | 0x00000000 | 0x00000000 | No |
| 7 | $a3 | 0x00000000 | 0x00000000 | No |
| 8 | $t0 | 0x00000000 | 0x00000000 | No |
| 9 | $t1 | 0x00000000 | 0x00000000 | No |
| 10 | $t2 | 0x00000000 | 0x00000000 | No |
| 11 | $t3 | 0x00000000 | 0x00000000 | No |
| 12 | $t4 | 0x00000000 | 0x00000000 | No |
| 13 | $t5 | 0x00000000 | 0x00000000 | No |
| 14 | $t6 | 0x00000000 | 0x00000000 | No |
| 15 | $t7 | 0x00000000 | 0x00000000 | No |
| 16 | $s0 | 0x00000000 | 0x00000000 | No |
| 17 | $s1 | 0x00000000 | 0x00000000 | No |
| 18 | $s2 | 0x00000000 | 0x00000000 | No |
| 19 | $s3 | 0x00000000 | 0x00000000 | No |
| 20 | $s4 | 0x00000000 | 0x00000000 | No |
| 21 | $s5 | 0x00000000 | 0x00000000 | No |
| 22 | $s6 | 0x00000000 | 0x00000000 | No |
| 23 | $s7 | 0x00000000 | 0x00000000 | No |
| 24 | $t8 | 0x00000000 | 0x00000000 | No |
| 25 | $t9 | 0x00000000 | 0x00000000 | No |
| 26 | $k0 | 0x00000000 | 0x00000000 | No |
| 27 | $k1 | 0x00000000 | 0x00000000 | No |
| 28 | $gp | 0x10008000 | 0x10008000 | No |
| 29 | $sp | 0x7fffeffc | 0x7fffeffc | No |
| 30 | $fp | 0x00000000 | 0x00000000 | No |
| 31 | $ra | 0x00400018 | 0x00400018 | No |

Jr **$ra** jumps to the address **$ra**. So, it jumps to 0x00400018. At PC register, 0x0040003c, it then jumps to 0x00400018

The difference between **jal** and **jr** is that **jal** copies the next instruction into **$ra** and then jumps to another instruction, while **jr** just jumps to the address **$ra**.