MIPS Pseudo-Instructions

Pseudo-instructions are legal MIPS assembly language instructions that do not have a direct hardware implementation. Instead, they are provided as a convenience for the programmer. The assembler translates them into equivalent MIPS instructions.

Task	**Pseudo-Instruction**	**Programmer Writes**	**Assembler Translates To**
Move the contents of one register to another.	`move <dest> <source>`	`move $t0, $s0`	`addu $t0, $zero, $s0`
Load a constant into a register. (Negative values are handled slightly differently.)	`li <dest> <immed>`	`li $s0, 10`	`ori $s0, $zero, 10`
Load the value of a memory location into a register. Here, 60 is the offest of the variable from the beginning of the data segment.	`lw <dest> <label>`	`lw $s0, variable`	`lui $at, 0x1001` `ori $s0, $at, 0x1001`

Task	**Pseudo-Instruction**	**Programmer Writes**	**Assembler Translates To**
Load the address of something from the data segment into a register. Here, 60 is the offset of the variable from the beginning of the data segment.	`la <dest> <label>`	`la $s0, variable`	`lui $at, 0x1001` `ori $s0, $at, 60`
If r1 < r2, branch to label.	`blt <r1>, <r2>, <label>`	`blt $t0, $t1, for_exit`	`slt $at, $t0, $t1` `bne $at, $zero, for_exit`
If r1 <= r2, branch to label.	`ble <r1>, <r2>, <label>`	`ble $t0, $t1, for_exit`	`slt $at, $t1, $t0` `beq $at, $zero, for_exit`
If r1 > r2, branch to label.	`bgt <r1>, <r2>, <label>`	`bgt $t0, $t1, for_exit`	`slt $at, $t1, $t0` `bne $at, $zero, for_exit`
If r1 >= r2, branch to label.	`bge <r1>, <r2>, <label>`	`bge $t0, $t1, for_exit`	`slt $at, $t0, $t1` `beq $at, $zero, for_exit`