MIPS Pseudo-Instructions

Pseudo-instructions are legal MIPS assembly language instructions that do not have a direct hardware implementation. They are provided as a convenience for the programmer. When you use pseudo-instructions in a MIPS assembly language program, the assembler translates them into equivalent real MIPS instructions.

Here is a list of the some commonly used pseudo-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" stands for `load immediate`.	`li <dest> <immed>`	`li $s0, 10`	`ori $s0, $zero, 10`
Load the *word* stored in a named memory location into a register. `Variable` is a label that the programmer has attached to a memory location. 12 is the offset of that memory location from the beginning of the data segment. It is calculated by the assembler for you.	`lw <reg> <label>`	`lw $s0, variable`	`lui $at, 0x1001` `lw $s0, 12($at)`

Task	**Pseudo-Instruction**	**Programmer Writes**	**Assembler Translates To**
Load the *address* of a named memory location into a register. `Value` is a label that the programmer has attached to a memory location. 16 is the offset of that memory location from the beginning of the data segment. It is calculated by the assembler for you.	`la <dest> <label>`	`la $s0, variable`	`lui $at, 0x1001` `ori $s0, $at, 16`
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`