MAL Integer Instructions — Explanations

MAL instructions fall into five main categories.

- Data movement instructions move data from one place to another.
- Data transformation instructions modify data.
- Data comparison instructions compare data items.
- Sequence control instructions modify the normal sequential execution of instructions.
- System access instructions provide access to operating system services.

The data comparison and sequence control functionalities are combined in the integer conditional branch instructions. Jump instructions are control instructions used for subprogram calls and returns.

The following operand notation indicates the type of operands that can be used with the various instructions.

Symbol	Meaning
label	an instruction label
addr	a memory address (see next table)
const	a constant
R	a general purpose register
Rc	a general purpose register or a constant
R_{targ}	register that contains the target address of a register jump
R_{save}	register that receives the return address of a jalr instruction

The following modes can be used for memory addresses.

Mode	Notation	Meaning
direct	*label*	The operand address is the address assigned to *label*
register direct	(*R*)	The operand address is the contents of register *R*
base displacement	*const*(*R*)	The operand address is the sum of the contents of register *R* and *const*

Jump Instructions

Jump instructions are control instructions that are usually used for subprogram calls and returns. The `jal` (jump and link) instruction is usually used for calling a subprogram. In addition to transferring execution to a target address, it save the address of the instruction following the `jal` instruction in register `$ra`. Then the subprogram can return with the code `jr $ra`.

In object-oriented programs, method addresses are stored in either object or class structures. The methods are invoked by loading the method address into a register and then using the `jalr` (jump and link register) instruction.

System Access

The MIPS processor supports a large number of operating system calls by using the register $v0 to indicate the particular operating system service that is needed. Thus all system calls have at least the following code.

```
li      $v0, system call code
syscall
```

An additional instruction or two may be needed before the syscall instruction to set up parameters for the system call. These instructions will load registers with the required parameters. Some system calls will return a value in a register.