Part 3: MIPS Registers

What is a CPU register?

You can think of a CPU register abstractly as a variable or by analogy a box, it's something you can store values in.

A register is a special area of storage in a CPU that is used for math (addition, subtraction), moving values, and storage of pointers (references to memory)

In hardware, they are a bunch of flip flops, which you can see here

What are the MIPS register conventions?

MIPS features 32 general-purpose registers which can be addressed using register numbers $0 through $31 or by their alternative names such as $sp, $gp, etc.

This table is adapted from this site with modifications.

Register Number	Alternative Name	Conventional Usage
$0	$zero	The value zero, used to move values / NOP
$1	$at	(assembler temporary) reserved by the assembler
$2	$v0	Used to store syscall number / to store return values of function calls like eax in x86
$3	$v1	(value) from expression evaluation and function results
$4 - $7	$a0 - $a3	(arguments) first four parameters passed to a function, not expected to be saved across function calls
$8 - $15	$t0 - $t7	(temporaries) expected for functions to use to store temporary values, values not expected to be saved across function calls.
$16 - $23	$s0 - $s7	(saved values) callee saved, preserved across function calls. functions that use one of these registers are expected to save and restore previous values.
$24	$t8	(temporaries) the same as $t0 - $t7

$25	$t9	(temporaries) the same as $t0 - $t8; however, $t9 is also used to store the address of a shared library during library function calls (example printf).
$26 - $27	$k0 - $k1	(kernel) reserved for use by the interrupt handler
$28	$gp	(global pointer) points to static data area of program memory
$29	$sp	(stack pointer) points to the top of the stack
$30	$s8 / $fp	(saved value or frame pointer) pretty much the base pointer (ebp) in x86
$31	$ra	(return address) stores the return address during a function call

Finally there is the program counter ($pc), which contains the current memory address of the instruction that the CPU is executing.

The table may be a lot to look at once so keep it handy. Here's a link of the table by itself for you to print out.

As a summary for the most important registers to know, $a0, $a1, $a2, and $a3 are used to pass arguments to functions, any register starting with **t** is **temporary** and thus when used, the value isn't expected to be saved across function calls, whereas any register beginning with **s** is **saved** and expected to persist across function calls.