

Notes for 9/11/09

MIPS addressing types:

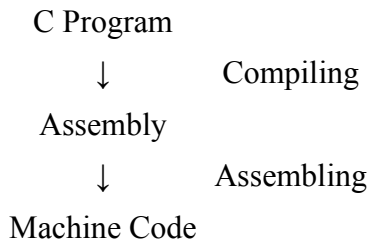
Addressing	Example	Bits	Range
Immediate Addressing	<code>addiu sp, sp, -32</code>	16	$-2^{15} \dots 2^{15} - 1$
Register Addressing	<code>jalr t9</code>	5	0 ... 31 (32 registers)
Base Addressing	<code>sw 16(sp)</code>	Immediate offset: 16 Register: 5 (contains 32)	Can point to anywhere in memory, since register can contain any 32-bit value
PC-relative Addressing	<code>bne \$t0, \$t1, Label</code>	16	Since the lowest two bits of address are assumed zero, label gets shifted left by two, and effective range is: $PC \pm -2^{17} \dots 2^{17} - 1$
Pseudodirect Addressing	<code>j Label</code>	26	Anywhere in current 256MB chunk

Note that the pseudodirect label is formed by the highest two bits from the PC or'd with the Label shifted left by two, i.e.:

PPLLLLLLLLLLLLLLLLLLLLLLLLLLLLLL00

In the PC-relative and pseudodirect addressing cases, it is safe to assume that the lower two bits are zero because, in MIPS, all instructions are word aligned.

Life cycle of code:



MIPS has no microcode, since it is already a simple enough of an architecture. Other architectures, especially CISC architectures like the wretched x86, tend to convert machine code to microcode in the processor. Microcode tends to be processor-specific, even within the same architecture, and most processors, such as Intel's x86 processors, do not expose the microcode to software, so you cannot see the microcode or program it yourself.