Registers Versus Variables

A key to mastering assembly language programming is thinking about your variables differently from how you may think about them presently. In your previous experience, variables are things that live in RAM. Apart from dynamically allocated memory, you've really had few other worries to consider.

Your job as an assembly language programmer is to plan out how to keep as much of your computational needs in registers for as long as possible.

As seen in the previous section, reaching out to RAM is enormously costly in terms of time and delay. Learning to maximize register use is critical.

There are 31 general purpose registers of which you can use a majority (a small number are reserved, in general, for particular purposes). There are also as many floating point and vector processing registers with fewer of these reserved for special purposes.

Consider the typical function you have written. Chances are there are no more than a handful of local variables. Typically you can plan out how to keep all of your local variables in registers for the entire life of your function. Exceptions include local arrays since arrays don't map well to registers.

There **are** some constraints placed on your register use. These are described in the next section.