Hello World in SPARC Assembler

(assuming your SPARC is running Solaris)

hi.s:

```
        .section        ".text"
        .global         _start
_start:
        mov     4,%g1                   ! 4 is SYS_write
        mov     1,%o0                   ! 1 is stdout
        set     .msg,%o1                ! pointer to buffer
        mov     (.msgend-.msg),%o2      ! length
        ta      8

        mov     1,%g1                   ! 1 is SYS_exit
        clr     %o0                     ! return status is 0
        ta      8

.msg:
        .ascii  "Hello world!\n"
.msgend:
```

```
$ as hi.s
$ ld -o a.out -dn -s hi.o
$ ./a.out
Hello world!
$
```

Linker flags

These aren't strictly necessary, but make the resulting executable a lot smaller:

-dn = produce a static (i.e. not dynamically linked) binary
-s = strip symbol and debugging information

What the code does

We use two system calls - `write()` then `exit()`.
To use a system call:

- Put the syscall number into the `%g1` register.
- Put the arguments into the output registers (`%o0`-`%o8`).
- Trap to kernel (trap 8).

You can find the syscall numbers in `/usr/include/sys/syscall.h`

Arguments for **`SYS_write`**:
`%o0` - file descriptor to write to

`%o1` - pointer to buffer
`%o2` - length of string

Arguments for **`SYS_exit`**:
`%o0` - return status

Synthetic Instructions

To set a register to point to a memory location, we have an interesting situation. The register is 32 bits wide, the memory location is 32 bits wide, and all our instructions are 32 bits wide, so we actually need two instructions to initialize the register.

`set const, %reg` is a "synthetic instruction" that the assembler expands into:

```
        sethi   %hi(const), %reg
        or      %reg, const, %reg
```

A few other synthetic instructions:

`mov %reg1, %reg2` is actually `or %g0, %reg1, %reg2`.
On the SPARChitecture, global register zero always has value zero.

To zero out a register, we used `clr %reg`. What the assembler actually produces is `or %g0, %g0, %reg`.

SPARC's idea of a `nop` is `sethi 0, %g0`
