Recall what a stack is and what push and pop do.

We need a pointer register to tell the program where in memory the top of the stack is (what address).
Nios II uses `r27` as the "top of stack pointer", also called `sp`. All programs will use the stack, and you must initialize it at the beginning.

We will start at a "high" address and grow the stack "downwards", or towards "lower" addresses.

We can use this to initialize our stack:

> `movia sp, 0x20000`

Let's push a word onto the stack. To push it, we need to send it to an address that is 4 bytes lower than the one currently pointed to by `sp`, which in this case is `0x1fffc`

```
movia r8, 0x1234f678
subi sp, sp, 4
stw r8, (sp)
```

To pop off of the stack, we have to do the reverse, in reverse order:

```
ldw r9, (sp)
addi sp, sp, 4
```

The LIFO behaviour of a stack is what allows subroutines to call other subroutines:

```
movia sp, 0x20000
movi r4, 1
call sub_1

done: br done

sub_1:
        subi sp, sp, 4 /* push ra onto the stack */
        stw ra, (sp)

        call sub_2

        ldw ra, (sp) /* pop stack into ra before returning*/
        add sp, sp, 4
        ret

sub_2:
        subi sp, sp, 4 /* push ra onto the stack */
        stw ra, (sp)

        /* sub_2 code */

        ldw ra, (sp) /* pop stack into ra before returning*/
        add sp, sp, 4
        ret
```

At the beginning of every single subroutine, you **should always push** the return address on the stack to prevent it from being overwritten.
You **should always pop** the top of the stack into the return address at the end of every sub routine.