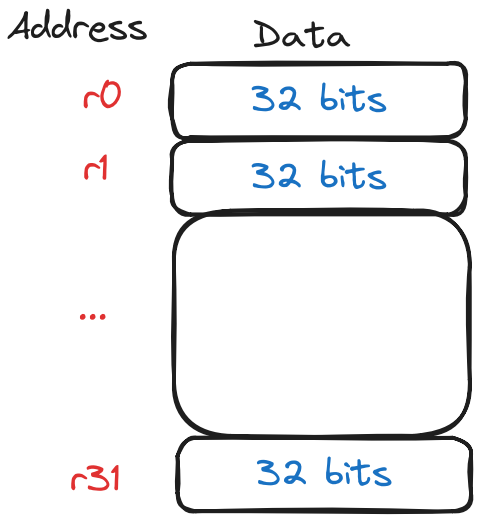
The computer has three parts:

- processor
- memory
- input/output

Let's take a closer look at the registers inside the processor, and how it is modelled in Assembly on the Nios II:



Note that some of these bits are "special". The r0 address always has the value 0, and cannot be written to.

We should stick to only freely using the registers r8-r15.

Think about the C code:

```
int A, B, C, D;
A = 1;
B = 8;
C = A + B;
D = B - A;
```

How do we turn this code into assembly (on the Nios II)?

```
.global start
_start:
        movi r8, 1
        movi r9, 9
        add r10, r8, r9
        sub r11, r9, r8
iloop: br iloop
```

Let's break it down:

- `movi` stands for "move immediate". The two arguments it takes here are an address and a 32-bit number. `movi r8, 1` will put the integer `1` in the register with address `r8`.
- `add` takes three arguments, all addresses for registers. `add r10, r8, r9` will store the sum of the values in `r8` and `r9` into `r10`.
- similarly, `sub r11, r9, r8` will store the result of the subtraction of the values `r9 - r8` in `r10`.
- `iloop: br iloop` is an instruction told to the computer to stop performing other instructions by being forcing it to be stuck in an infinite loop in this line. `br` stands for "branch", and will come up more later.