Immediate Arithmetic Instructions

- **Load upper immediate** loads `imm` value shifted by 12 bits into `$rd` and **add immediate** adds `imm` to the content of `$rs1` and stores the result in `$rd`.

- These two instructions are used to **initialize registers** (`$rs1 = $zero`) and to **load addresses** into a register in order to read values from memory.

- `lui` is used to load the upper and `addi` to load the lower bits - see `load_integer(uint64_t value)`.

- A special case of `addi` is `nop`, with `$zero = $zero + 0`.

lui	$rd	imm	
addi	$rd	$rs1	imm

Arithmetic Instructions

`add`	`$rd`	`$rs1`	`$rs2`
`sub`	`$rd`	`$rs1`	`$rs2`
`mul`	`$rd`	`$rs1`	`$rs2`
`Divo`	`$rd`	`$rs1`	`$rs2`
`Dive`	`$rd`	`$rs1`	`$rs2`
`remu`	`$rd`	`$rs1`	`$rs2`

```
$rd = $rs1 +,-,*,/,% $rs2
```

- The processor executes these instructions using unsigned integer arithmetic with wrap-around semantics.

Immediate Arithmetic Instructions

`lui`	`$rd`	`imm`	
`addi`	`$rd`	`$rs1`	`imm`

- **Load upper immediate** loads `imm` value shifted by 12 bits into `$rd` and **add immediate** adds `imm` to the content of `$rs1` and stores the result in `$rd`.

- These two instructions are used to **initialize registers** (`$rs1 = $zero`) and to **load addresses** into a register in order to read values from memory.

- `lui` is used to load the upper and `addi` to load the lower bits - see `load_integer(uint64_t value)`.

- A special case of `addi` is `nop`, with `$zero = $zero + 0`.