This section describes the available arithmetic **opcodes/mnemonics** and their corresponding operations.

> All arithmetic instructions accept **only a single operand**.
> The **other operand**, as well as the **destination**, is taken from one of the **Link registers**:
> `L0`, `L1`, `L2`, `L3`.

📎 See: Register Reference – Link Registers

🔢 Addition

The following opcodes are used for **addition**:

- `ADDI` — Add Signed Integer
- `ADDU` — Add Unsigned Integer
- `ADDF` — Add Floating Point

??? abstract "ADDI — *Add Signed Integer*"

```
=== "Properties"

    | Property        | Value                       |
    |-----------------|-----------------------------|
    | **Opcode**      | 13                          |
    | **Type**        | Arithmetic                  |
    | **Operand Type**| Signed 64-bit integer       |
    | **Destination** | `L2` (implicit)             |

=== "Algorithm"

    ```
    L2 = L2 + <signed_imm>
    ```

=== "Example"

    ```
        ADDI 1
    ```
```

??? abstract "ADDU — *Add Unsigned Integer*"

```
=== "Properties"

    | Property        | Value                       |
    |-----------------|-----------------------------|
```

```
    | **Opcode**      | 18                          |
    | **Type**        | Arithmetic                  |
    | **Operand Type**| uNSigned 64-bit value       |
    | **Destination** | `L3` (implicit)             |

=== "Algorithm"

    ```
    L3 = L3 + <signed_imm>
    ```

=== "Example"

    ```
        ADDU 1
    ```
```

- ADDU — Add Unsigned Integer

...

- ADDF — Add Floating Point

Opcode	**Code**	**Operand Count**	**Opernads**	**Description**
SUBI	
MULI	
DIVI	
MODI	
ADDU	
SUBU	
MULU	
DIVU	
MODU	
ADDF	
SUBF	
MULF	
DIVF	