➕ Arithmetic

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

🔹 ADDI — *Add Signed Integer*

- **Opcode**: `13`
- **Operand Accepted**: 64-bit signed integer
- **Operation**:
  Adds the operand to the value in **register L2**.
  The result is **stored back in L2**.

??? example "Example: ADDI"
`sasm ADDI 1 ; Adds 1 to the current value in L2`

> L2 is treated as a 64-bit signed integer for this operation.

Arithmetic

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

All the arithmetic instructions accept only a single operand.
The other operand(also the destination) is taken from the Link register L0, L1, L2, L3

SEE {REFERENCE}

REFERENCE NEEDED `registers-Link_registers`

Addition

The addition instructions are
- ADDI (ADD Integer)

- ADDU (ADD Unsigned)
- ADDF (ADD Floating point values)

1. ADDI:
   Opcode: 13
   Operand Accepted: 64-bit signed integers
   Adds the passed 64-bit signed int value to the value stored in the register L2(also considered to be an 64-bit signed integer value)
   The result is stored in the L2 Register

!!! Example

```
ADDI 1 ;Will add 1 to whatever value is in register L2
```

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