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

- DIVI — Divide by Signed Integer
- DIVU — Divide by Unsigned Integer
- DIVF — Divide by Floating Point

DIVI — *Divide by Signed Integer*

```
    L2 = L2 / <signed_imm>
    L2 = L2 / <reg_val>
    L2 = L2 / <const>
```

=== "DIVI Example"

```
```linenums="1" hl_lines="1 3 5 7"
; imm +ve
    DIVI    1
; imm -ve
    DIVI    -123
; reg val
    DIVI    val(QT)
; const
    DIVI    SOME_CONST_VAL
```
```

=== "DIVI Properties"

```
| Opcode | Operand Type          | Destination   |
|--------|-----------------------|---------------|
| 16     | Signed 64-bit integer | L2 (implicit) |

Identified as memonic [#16](#DIVI), DIVI is used to
divide the L2 register by a 64-bit signed value
```

DIVU — *Divide by Unsigned Integer*

```
    L3 = L3 / <unsigned_imm>
    L3 = L3 / <reg_val>
    L3 = L3 / <const>
```

```
=== "Properties"

    | Property        | Value                       |
    |-----------------|-----------------------------|
    | **Opcode**      | 21                          |
    | **Type**        | Arithmetic                  |
    | **Operand Type**| Unsigned 64-bit value       |
    | **Destination** | `L3` (implicit)             |
```

=== "Example"

```
    ```
    ; imm +ve
        DIVU   1
    ; reg val
        DIVU   val(QT)
    ; const
        DIVU   SOME_CONST_VAL

    ```
```

DIVF — *Divide by Float value*

```
=== "Properties"

    | Property        | Value                       |
    |-----------------|-----------------------------|
    | **Opcode**      | 26                          |
    | **Type**        | Arithmetic                  |
    | **Operand Type**| 64-bit float value          |
    | **Destination** | `L1` (implicit)             |

=== "Algorithm"

    ```
    L1 = L1 / <float>
    L1 = L1 / <reg_val>
    L1 = L1 / <const>
    ```

=== "Example"

    ```
    ; imm float
        DIVF    3.14
    ; reg val
        DIVF    val(QT)
```
```

```
; const
    DIVF    SOME_CONST_VAL

```