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

- MULI — Multiply Signed Integer
- MULU — Multiply Unsigned Integer
- MULF — Multiply Floating Point

MULI — *Multiply Signed Integer* {#MULI}

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

=== "MULI Example"

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

```
~~~

=== "MULI Properties"

```
| Opcode | Operand Type           | Destination   |
|--------|------------------------|---------------|
| 15     | Signed 64-bit integer | L2 (implicit) |

Identified as memonic [#MULI](#MULI), MULI is used to
multiply the L2 register with a 64-bit integer.
```

MULU — *Multiply Unsigned Integer* {#MULU}

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

=== "MULU Example"

~~~
```linenums="1" hl_lines="1 3 5"
    ; imm +ve
        MULU    1
    ; reg val
        MULU    val(QT)
    ; const
        MULU    SOME_CONST_VAL

```
~~~

=== "MULU Properties"

```
| Opcode | Operand Type           | Destination    |
|--------|------------------------|----------------|
| 20     | Unsigned 64-bit value  | L3 (implicit) |

Identified as memonic [#MULU](#MULU), MULU is used to
multiply the L3 register with a 64-bit unsigned value
```

MULF — *Multiply Float value* {#MULF}

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

=== "MULF Example"

~~~
```linenums="1" hl_lines="1 3 5"
    ; imm float
        MULF    3.14
    ; reg val
        MULF    val(QT)
    ; const
        MULF    SOME_CONST_VAL

```
~~~

=== "MULF Properties"

```
| Opcode | Operand Type       | Destination   |
|--------|--------------------|---------------|
| 25     | 64-bit Float Value | L1 (implicit) |

Identified as memonic [#MULF](#MULF), MULF is used to
multiply the L1 register with a 64-bit float value
```
