

---

## The MULS and DIVS instructions

---

- **Multiply instruction in M68000**

- **Warning:**

- The **M68000** can **only multiply** two **16-bits integer numbers**  
(This **limitation** is due to the **technological limitation** at the time - circa 1980)
- 

- The **syntax** of the **multiply** instruction of M68000 is:

```
MULS <ea>, Dn      Multiply the 16 bit integer value in
                    the operand specified by <ea> to
                    the 16 bit value in data register Dn

                    The product is always 32 bits and it is
                    stored in data register Dn

                    In other words:

                    Dn(32 bits) = Dn(16 bits) * <ea>(16 bits)
```

- Notice that you do **not** have any choice for operand size.

---



---



---

- **Example of the MULS instruction**

- Suppose you have the following bit pattern in D0:

```
+-----+-----+-----+-----+
D0 = | 10101010 | 01010101 | 00000000 | 00001001 |
+-----+-----+-----+-----+
```

The 16 bit operand in D0 is equal to:

$$1001_{(2)} = 1 + 8 = 9_{(10)}$$

The **16 bit** representation in **D0** represents the value **9<sub>(10)</sub>**

**Suppose** we

```
MULS #3, D0
```

The data register **D0** will contain:

```
D0 = | 00000000 | 00000000 | 00000000 | 00011011 |
```

The 32 bit result in D0 is equal to:

$$11011_{(2)} = 1 + 2 + 8 + 16 = 27_{(10)}$$

Notice that **all 32 bits** in the register D0 are updated because the product of two 16-bit binary number is always 32 bits.

---

---

- **Divide instruction in M68000**

- **Important note:**

- The **M68000** can *only* divide a **32-bits integer number** by a **16-bits integer number**

(This, again, is due to the **technological limitation** at the time in 1980)

- The **syntax** of the **DIVS** instruction is:

```
DIVS <ea>, Dn    Divides a 32 bit value in data register Dn
                  by a 16-bit value specified by <ea>
```

In other words:

$$Dn_{(32 \text{ bits})} / \langle ea \rangle_{(16 \text{ bits})}$$

Result:

- (1) the **quotient** is stored in the **lower 16 bits** of data register Dn
- (2) the **remainder** is stored in the **upper 16 bits** of data register Dn

---

---

- **Warning:**

- the **DIVS** instruction can result in **error** !!!

**E.g.:**

**1 / 0 !!!!**

- The **DIVS** is **voided** if the **execution** results in **error**

- **Example of the DIVS instruction**

- Suppose register **D0** contains the value **9** (in binary !):

```

D0 = | 00000000 | 00000000 | 00000000 | 00001001 |
-----+-----+-----+-----+
The 32 bits represents the value 9(10)

```

Then **after executing** the following **DIVS** instruction:

```

DIVS #4, D0

9 / 4 --> Quotient = 2
          Remainder = 1

```

The **data register D0** will contain:

```

D0 = | 00000000 | 00000001 | 00000000 | 00000010 |
-----+-----+-----+-----+

```

- The **quotient** is stored in the **lower half** of the register
- The **remainder** is stored in the **upper half** of the register
- **NOTE:** when you use **EGTAPI** and display register D0, you need to display it in **half word** quantity to see the quotient and remainder parts !

- **The SWAP instruction**

- **Notice that:**

- the **remainder** of a **division** is **stored** in the **upper half** of the **data register**.

- **However:**

- **Word size operands** are **stored** in the **lower half** in the **data registers**

- **Fact:**

- The **SWAP** instruction is used to:
  - **make** the **remainder** of the **division** **available** as **operand** in a **data register**

- The **SWAP** instruction:

- The **SWAP D<sub>n</sub>** instruction **exchanges** the **upper** and the **lower halves** of the data register **D<sub>n</sub>**.

- **Example:**

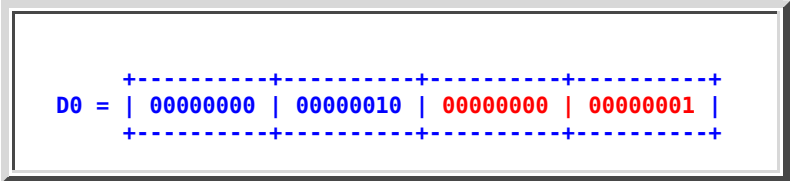
- Suppose data register **D0** contains the following:

D0 = | 00000000 | 00000001 | 00000000 | 00000010 |

- After **executing** the following unstruction:

**SWAP D0** (= exchanges the upper and lower halves of D0)

The data register **D0** will contain the following:



```
D0 = | 00000000 | 00000010 | 00000000 | 00000001 |
```

---

---

---

- **Teaching notes**

- We will **learn** how to **use** the **MULS**, **DIVS** and **SWAP** instructions **later**

---

- We need to **cover**:

- **Converting operand sizes** first !!!