Address Register Operands

- Operands in ***address registers***
  - **Fact:**
    - Address registers are mainly used to **store address values** needed to **access variables** in memory
    - We will **not** use an **address register** to store **intermediate results** of **computations**
    - Therefore, I will present a set of **simplified rules** on using **address registers**
  - Rules for using **address registers**
    - You **cannot** use the `.b` mode (byte size operand) with **address registers**

      Example:

```
move.b   #-1, a0              not allowed

add.b    #-1, a0              not allowed

add.b    d0, a0               not allowed
```

    - When you **store a value** into an **address register**, the value is **always** stored in **32 bits representation**

      (That is because an **address** in **M68000** is **always** represented by **32 bit**)

      Example:

```
movea.w  #-1, a0         ; stores 11111111 1111111 11111111 11111111 in a0

movea.l  #-1, a0         ; stores 11111111 1111111 11111111 11111111 in a0
```

      Note: the ***difference*** between these **2 instructions** is the **encoding**.

      The first instruction uses 6 bytes, the second uses 10 bytes. (See a.lst in demo)

    - The **status flags (N,Z,V,C)** are ***not* updated** when an operation **updates an address register**

      (That is because the operation is not consider a ***data* computational operation**)
  - Example **word size** operation with **address register as *destination***

```
Before operation:
          +----------+----------+----------+----------+
   D0 =   | 00000000 | 00001001 | 10010010 | 10111000 |
          +----------+----------+----------+----------+
          +----------+----------+----------+----------+
   A0 =   | 00000000 | 00000000 | 00000000 | 00000000 |
          +----------+----------+----------+----------+

Operation: MOVE.W D0, A0              (A0 is the destination)
```

```
After operation:
         +----------+----------+----------+----------+
   D0 =  | 00000000 | 00001001 | 10010010 | 10111000 |
         +----------+----------+----------+----------+
         +----------+----------+----------+----------+
   A0 =  | 11111111 | 11111111 | 10010010 | 10111000 |
         +----------+----------+----------+----------+

   Flags in PSR are unchanged
```

Note:

- Because the ***16 bit* representation** indicates that the value is **negative**, it is automatically converted to a **32 bit representation** for a **negative value** by **prepending 1 bits** before the representation.

- Example **long word size** operation with **address register as *destination***

```
Before operation:
         +----------+----------+----------+----------+
   D0 =  | 00000000 | 00001001 | 10010010 | 10111000 |
         +----------+----------+----------+----------+
         +----------+----------+----------+----------+
   A0 =  | 00000000 | 00000001 | 00000000 | 00000000 |
         +----------+----------+----------+----------+

Operation: MOVE.L D0, A0

After operation:

         +----------+----------+----------+----------+
   D0 =  | 00000000 | 00001001 | 10010010 | 10111000 |
         +----------+----------+----------+----------+
         +----------+----------+----------+----------+
   A0 =  | 00000000 | 00001001 | 10010010 | 10111000 |
         +----------+----------+----------+----------+

   Flags in PSR are unchanged
```

- **DEMO:** click here

- **Remember that:**

  - **Any *byte* size** operation with **address register as destination** are **NOT allowed** !!!

    Examples:

```
      MOVEA.B   #3, A0           (MOVEA = move to Address reg) - not allowed

      ADDA.B    #3, A0           (ADDA = Add to Address register) - not allowed

      SUBA.B    #3, A0           (SUBA = Subtract from Address reg) - not allowed
```

  - **Word size** and **long size** operation with **address register as destination** are ***allowed*** !!!

    Examples:

```
      MOVEA.W   #3, A0         allowed
      ADDA.W    #3, A0         allowed
      SUBA.W    #3, A0         allowed

      MOVEA.L   #3, A0         allowed
      ADDA.L    #3, A0         allowed
```

```
    SUBA.L   #3, A0         allowed
```

- **Any data size** operation with ***data* register as destination** are ***allowed*** !!!

  Examples:

```
    MOVE.B  #3, D0         allowed
    ADD.B   #3, D0         allowed
    SUB.B   #3, D0         allowed

    MOVE.W  #3, D0         allowed
    ADD.W   #3, D0         allowed
    SUB.W   #3, D0         allowed

    MOVE.L  #3, D0         allowed
    ADD.L   #3, D0         allowed
    SUB.L   #3, D0         allowed
```

Notice also that the instruction **mnemonic** for ***data* destination registers *do not*** have the **trailing letter A** !!!