M68000 (Address Register) Indirect with Displacement Mode

- Recall the context that the address mode is used within the MOVE instruction:

```
MOVE <EA>,<EA>
      ^    ^
      |    |
      |    +--- Source operand 2 and Destination
      +-------- Source operand 1
```

- **Indirect mode (with displacement)**
  - **Syntax** to **specify** the (address register) **indirect mode** (with displacement):

```
   m(An)     where n = 0, 1, 2, 3, 4, 5, 6, or 7
                   m = a number between -32768 and 32767

Examples:

   4(A1)
  -8(A1)
```

  - **Semantics (meaning):**
    - The **operand** specified by `m(An)` is located in **memory** at the **memory address** given by:

```
    m + An (32 bits)
```

  - **Note:**

```
    0(An)   can be shorted to:    (An)
```

  - Examples:

```
MOVEA.L #1000,A0    (set up address register)
MOVE.L 4(A0), D0    will move a long word from
                    memory location at address 1004
                    (because A0 contains 1000,
                    so 1000 + 4 = 1004) into D0
```

```
MOVEA.L #1000,A0    (set up address register)
MOVE.L -4(A0), D0   will move a long word from
                    memory location at address 996
                    into D0

MOVEA.L #5678,A0    (set up address register)
MOVE.L 10(A0), D0   will move a long word from
                    memory location at address 5688
                    into D0
```

- **Advanced examples:**

```
(1a) Copy the value of a byte array element A[0] into register D0

       A:  DS.B 10       A byte array: int A[10]

       MOVEA.L #A,A0     A0 = base address of array A
       MOVE.B 0(A0), D0  Move element A[0] into reg. D0

(1b) Copy the value of a short array element A[0] into register D0

       A:  DS.W 10       A short array: int A[10]

       MOVEA.L #A,A0     A0 = base address of array A
       MOVE.W 0(A0), D0  Move element A[0] into reg. D0
                         (each element in a short array is 2 bytes long)

(1c) Copy the value of an int array element A[0] into register D0

       A:  DS.L 10       An integer array: int A[10]

       MOVEA.L #A,A0     A0 = base address of array A
       MOVE.L 0(A0), D0  Move element A[0] into reg. D0
                         (each element in a short array is 4 bytes long)
```

```
(2a) Copy the value of a byte array element A[3] into register D0

       A:  DS.B 10       A byte array: int A[10]

       MOVEA.L #A,A0     A0 = base address of array A
       MOVE.B 3(A0), D0  Move element A[3] into reg. D0

(2b) Copy the value of a short array element A[3] into register D0

       A:  DS.W 10       A short array: int A[10]

       MOVEA.L #A,A0     A0 = base address of array A
       MOVE.W 6(A0), D0  Move element A[3] into reg. D0
                         (each element in a short array is 2 bytes long)

(2c) Copy the value of an int array element A[3] into register D0

       A:  DS.L 10       An integer array: int A[10]

       MOVEA.L #A,A0     A0 = base address of array A
       MOVE.L 12(A0), D0  Move element A[3] into reg. D0
```

```
                          (each element in a short array is 4 bytes long)
```

---

(3) Suppose we define the class:

```
  class MyClass
  {
     int x;
     int y;
     short z;
  }
```

And an object of the type MyClass:

```
  MyClass A;
```

Then the object A is defined in assembler using:

```
  A: DS.B 10       * because object of MyClass has 2 int's
                   * 1 short variables, for a total of 10 bytes
```

And the following statements have the following equivalent in M68000 assembler instructions:

```
  A.x = 4000;  ->   MOVEA.L #A,A0
                    MOVE.L  #4000,0(A0)   because x has offset 0

  A.y = 8100;  ->   MOVEA.L #A,A0
                    MOVE.L  #8100,4(A0)   because y has offset 4

  A.z = 123;   ->   MOVEA.L #A,A0
                    MOVE.W  #123,8(A0)    because z has offset 8
                                          Make sure you use the
                                          right operand size !
```
