Passing parameters to & getting return value from a function

- Charateristics of Functions:
  - often have one or more input parameters
  - often return a value
- Note that:
  - If a function returns a value, then that is also the **very last** action that the function will do.
- Consider the **opportunity** to pass parameters to a function...

```
Example:

   main()                  int func(int x, int y)
   {                       {
      int a, b;               ....
                              ....
      func(a, b);             ....
   }                       }

Assembler code:

  main:  xxxxx           ->  func:  xxxxx
         xxxxx          /
         xxxxx         /
         xxxxx        /
         xxxxx       /
         BSR func -
         xxxxx
```

Because the **"BSR func"** will make the CPU jump to the first instruction in the function **"func"**, you **must** pass the parameters (if any) to the function **"func" BEFORE** the **"BSR func"** instruction

- What happens when a "parameter is passed":

```
Example:

   main()                  int func(int x, int y)
   {                       {
      int a, b;               ....
                              ....
      func(a, b);             ....
   }                       }
```

  - The **caller function `main`** passes two parameters (a and b) to the **callee function `func`**
  - In high level programming laguage terminology:
    - The **value** of the **actual parameters** (a & b) are **copied** to the **formal parameters** (x & y)
  - In assembler level, things are done quite differently....
  - The **ultimate goal** is to instruct the **callee function** to work with a specific set of input values.
  - This is achieved by:

- Having the **caller function `main`** and the **callee function `func`** agree on a **common** location where to find the parameters

- How do you pass parameters in an assembler program:
  - Prior to writing the **caller function `main`** and the **callee function `func`**, you must first **fix** (= agree) on the **location** to pass **each parameter**
  - When you write the **caller function `main`** in assembler, before the **BSR func** instruction, you must **copy** (= ***pass***) the values of the **actual parameters** to the **locations** that you have fixed previously.
  - When you write the **callee function `func`** in assembler, each time the code needs the value of some parameter variable, you must go get it from the previously fixed location.

- Passing **return value** from **callee function** back to **caller function**
  - Same principle as passing parameter
  - Caller and callee must agree on a fixed location for the callee to return the value
  - Note that the caller can (and should) immediately save the returned value in one of its (caller's) local variables, or else the return value may be overwritten and lost !

- **Example:**

```
  main()                  int func(int x, int y)
  {                       {
     int a, b, c;            return (x*x + y*y);
     int i, j, k;         }

     c = func(a, b);
     k = func(i, j);
  }

func() has 2 parameters and 1 return value

Agreement:  parameter 1 in register D0
            parameter 2 in register D1

            return value in register D7

The asembler program will look like this:

 main:
       MOVE.L a, D0  (pass a through D0)
       MOVE.L b, D1  (pass b through D1)
       BSR    func   ----------------------------+
                                                 |
       MOVE.L D7, c                              |
                                                 |
                                                 |
       MOVE.L i, D0  (pass i through D0)         |
       MOVE.L j, D1  (pass j through D1)         |
       BSR    func   -------------------------+  |
                                              |  |
       MOVE.L D7, k                           V  V
                                   func:  (func(x,y))
```

```
MULS  D0, D0   (Use the input parameters
MULS  D1, D1    put in the agreed locations)
ADD.L D1, D0    ** D0 = x^2 + y^2

MOVE.L D0, D7 (put return value
                in previously
                agreed location)
RTS
```

- **Example Program:** (Demo above code)

  Example

  - Prog file: click here

  How to run the program:

  - **Right click** on link and **save** in a scratch directory
  - To compile: `as255 subroutine1`
  - To run: use `m68000`

- **Warning:** this **solution** is ***flawed***

  - This **technique** (using **registers** to pass **parameters** ***only*** work for **one-level** of **subroutine call**

  - It **fails** will **more** than **1 level**:

    - If **A( )** calls **B( )** (and **B( )** uses **registers** to **pass** its **parameters**)

      and then B( ) calls **another function C( )** (and **C( )** uses ***also*** uses **registers** to pass its **parameters**) then:

      - Subroutine **C( )** will ***erase*** the **values** stored in **registers** by subroutine **B( )** !!!!!

        (because if you call **enough functions** deep enough, you will use up ***all* registers** sooner or later....)
