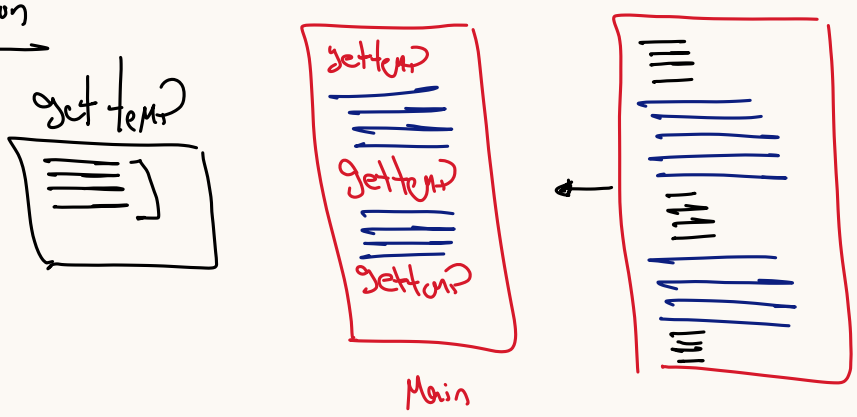
Function



Function type

- Normal Function → Must used
- Macro Function → #define → Used
- Inline Function

Normal Function

Create Function

① Prototype [Declaration]

return type FunctionName (Arguments);

↳ write Prototype → before Main / in header file

② Implementation [Definition]

return type FunctionName (Arguments)

- Naming Rule
- Same var Name

{

Codes

}

↳ write Prototype → After main() in Source file

③ Call → Used

FunctionName (Parameters);

↳ Call for Function → in side Main Function or inside other Function

return

- return → keyword → function → value.
- return one value only
- type (void , int, float, char, Array, double)

```
void Display ( );
int Main ( )
{ char x = Display( );
}
```

Compilation error ←

Arguments (inputs)

1. Can function not take any input
2. Can function take one input or multi input
3. input → Declartion for var

   Datatype InputName , Datatype IN

4. input type →
   - ① char ② int ③ float ④ double ⑤ struct ⑥ union

     Call by value

   - ① Pointer (int* x)
   - ② Array (int arr[ ])
   - ③ Function

     Call by Reference (Address)

Context Switch

```
int sum(int x, int y);
int main()
{
   int No1 = 0;
   int No2 = 0;
   // Assume to scan No1, No2;
   int Add = sum(No1, No2);
   Printf("Add = %d\n", Add);
}
int sum(int x, int y)
{
   return x+y;
}
```

Context Switching (Run Time)

before Call → Processor store for some var

- PC → Program Counter
- SP → stack Pointer
- GP → general Purpose Register

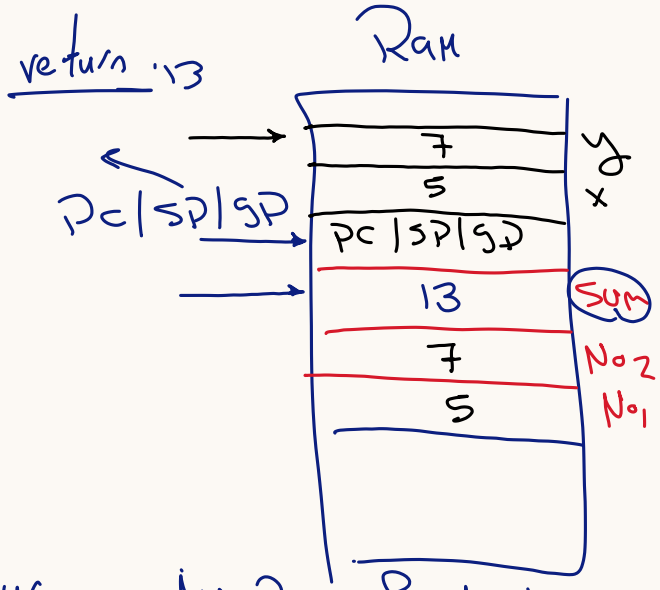
↳ var store in Ram in stack section

↳ load for this var

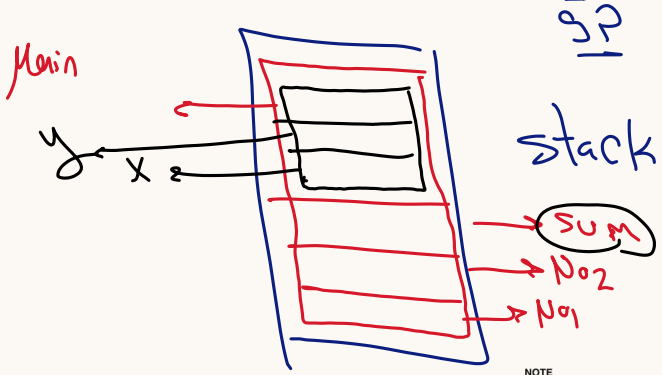
Memory Handle

```
int Add(int x, int y)
int main()
{
   int No1 = 0;
   int No2 = 0;
   int sum = Add(No1, No2);
}
```

return → will erase Function Block



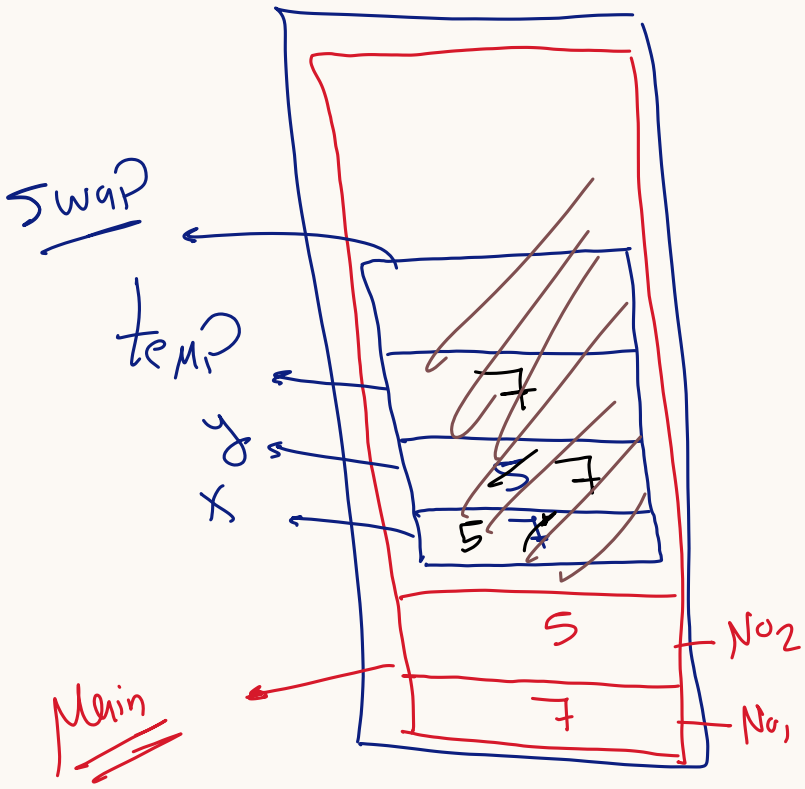
return → jump → for location store SP|PC|GP



Call by value

```
void swap (int x, int y);
void swap (int x, int y)
{
    int temp = x;
    x = y;
    y = temp;
}
```

```
int main ()
{
    int No1 = 7;
    int No2 = 5;
    // Print (No1);
    // Print (No2);
    swap (No1, No2);
    // Print (No1);
    // Print (No2);
}
```



// x = No1
// y = No2

Call by Reference

```
void swap (int * x, int *y);
void swap (int * x, int *y)
{
    int temp = *x;
    *x = *y;   =>  *(0x00) = *(0x04)
    *y = temp; <-  *(0x04) = temp;
}
```

```
int main ()
{
    int No1 = 7;
    int No2 = 5;
    // Print (No1);
    // Print (No2);
    swap (&No1, &No2);
    // Print (No1);
    // Print (No2);
```

temp

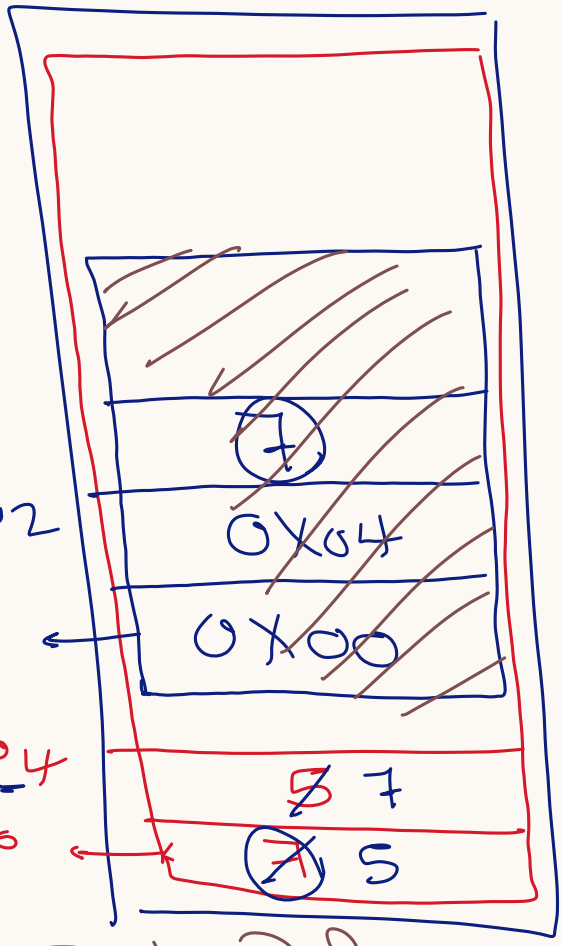
y = &No2

x = &No1

No2 0x04

-> 0x00

No1



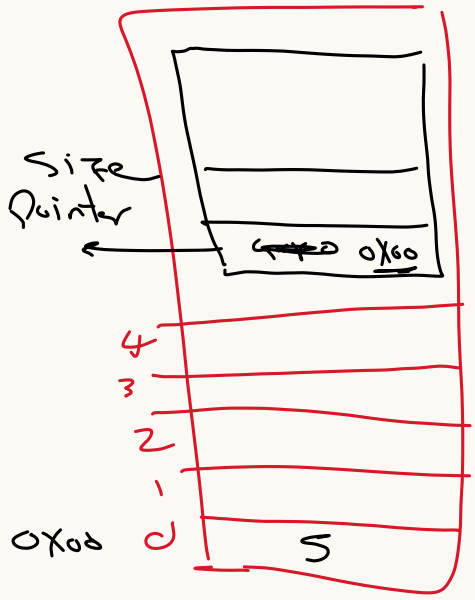
Call by Reference

Call by reference for Array

```
void SGenArray(int* arr, int size)
{
}

int main()
{
  int arr[5];
  SGenArray(arr, 5);
}
```

int arr[] → Compiler → Pointer



Lab1

↳ C Functions Add & Sub & Multi for two variable
