

## Function

التاريخ :

unit 2 lesson 5 lecture

الموضوع :

Difference between variable definition and declaration?

definition: a variable defined when the compiler generates instructions to allocate the storage of the variable.

memory → جاء في المكان الذي تم تعيين المتغير له

declaration: a variable is declared when the compiler is informed that a variable exists along with its type.

The compiler does not generate instructions to allocate the storage for the variable at this point.

جاء scope ١٢٣ لغة في كلّ دلالة هي بيان  
ويكتب كذلك على نفس بيان النوع لـ هذا  
declare alias lib & get sets في بيان  
خواص

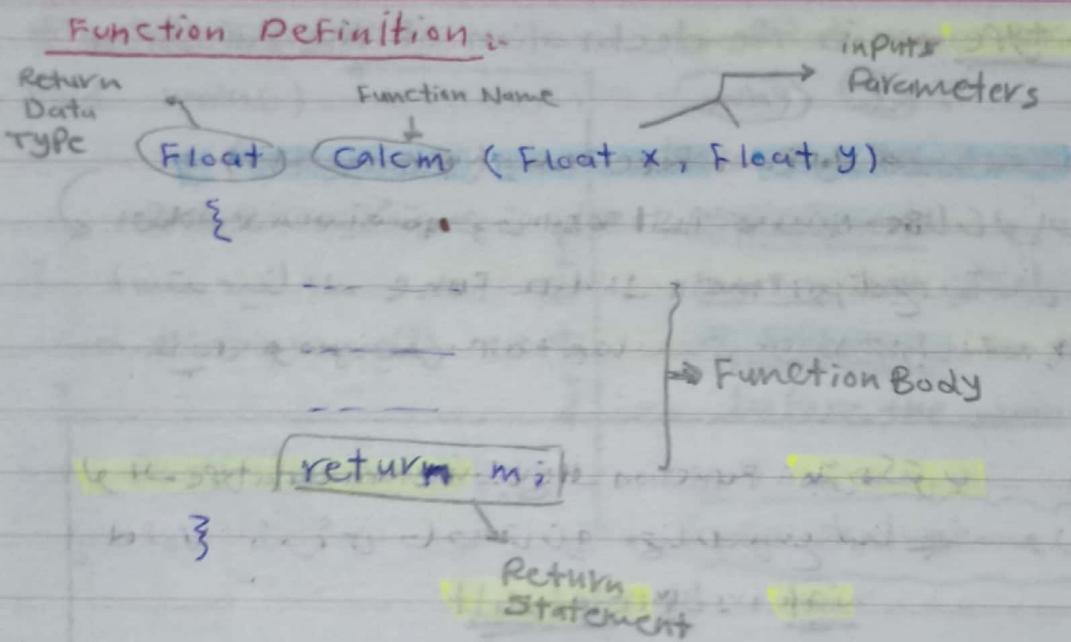
(note) Variable can be declared many times in a program  
But definition can happen only one time in the program

(note) A variable definition is also declaration, but not all variable declarations are definitions.

ex Extern int x; → declaration without definition

// The variable was already defined somewhere else  
and it's just recalled here.

int x; → definition and declaration.

Function Definition:Function Name → like Variable.

• الاسم الذي يمثل المخرج المطلوب من المدخلات.

Input Parameters → supplied parameters types and names.If there is no parameters then we write **Void**.Return TYPE → the data type of the Function output.if the Function has no output use **Void**.Function Body → performs specific Function operation

• كتابة الكود الذي يتحقق منه الهدف.

Return Statement → this statement tells the computer that the Function execution is completed.

Func will return data type like int or float

مثلاً جمل مثل طابعه return void يعني لا يوجد له قيمة

• that's all

Proto type → is the declaration of the Function

ex → `return datatype Func_Name(Parameters)`

العلام ده بيكتب خوف دالة main  
daring golnayi main چوار Func لابد main  
عالي، error occurs

**note** Function بقى مكتوب return datatype  
أمورى حاجة تانية خليها برجو ليها Void  
int by default

**note** Function definition فوق دالة  
Prototype يمكنها في العادة دى عارض main  
ما يقدر main دى تعرف بالdefinition  
error appears because Prototype  
لا زم ندخل دا

### Function call

`Func_Name (arguments list)`

يقدر دى Function N اسمها دى  
function 10 return ممكن تخزن  
لها return datatype دا لو ملأ فخونك  
عالي بدون ما تخزن قيمتها في ذا متغير

**note**

Prototype equivalent to function declaration

```
#include "stdio.h"
```

```
Void main()
```

```
{
```

```
    PrintWelcome();
```

```
}
```

```
Void PrintWelcome()
```

```
{
```

```
    printf("Welcome");
```

```
}
```

here  
+  
+  
+

Compilation error

why ?

because PrintWelcome is undefined

Solution we should put

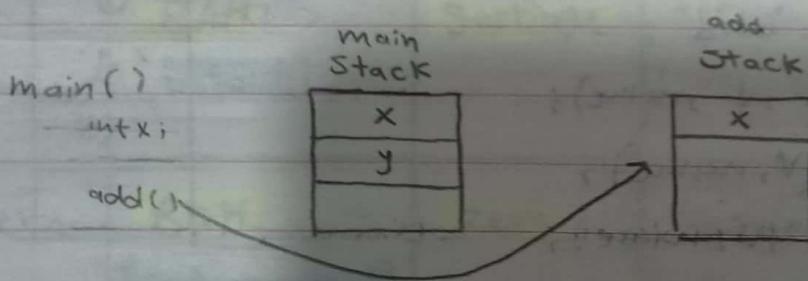
a prototype for the function before the main.

Void printWelcome()

### Pass by Value

1. This method copies the actual value of an argument into the formal parameter.

أو إذا بنيت نسخة من argument في main() Function،  
حصل تغير لبعضها البعض في Function.  
خ) المدخلة المطلوبة موجودة في main()



ما يحصل (ما ندعى Func نعمل him) هي  
عاتقوه، وسيطرن فيه سمات من المدخلات  
ال موجودين جوا الـ main

حياته، في حالة إننا كنا بنيت struct حجم 100 بايت فهو سيحمل نسخة  
من كل ممكنا فدورة هياربليس تأخير الـ Calling time وده هيئدى  
لأن تتحقق السمات.

Pass by Reference :-

This method **copies** the address of an argument into the formal parameter.

فهي تأخذ بعنوان الـ argument address.

ولو حصل تغير في قيمة جوا الـ Function فالتعديل

يؤثر على main() الذي يحتوي على نفس الـ memory.

```
#include <stdio.h>
```

```
Void tryToModify(int x, char text[])
```

```
{
```

```
x += i;
```

```
text[i] = j;
```

```
}
```

```
Void main()
```

```
{
```

```
int v = 5;
```

```
char name[5] = "Good";
```

```
printf("%d %s", v, name);
```

```
tryToModify(v, name);
```

```
printf("%d %s", v, name);
```

```
}
```

"Good" ، string ، 5 ، v .

فإذا ما طبعنا قيمة v الأول سيكونوا زائد ، اثنان.

وبعد كل اثناء دخول func نعود إلى main() .

الـ string v .

أما المتغير v فسيكون له قيمة الـ 7 .

لذلك طبعنا string في main() .

outPut → 5 Food

## Storage classes in C :-

The storage classes in C determine the

→ Scope of a Variable

\* يعني مدين الى يقدر يحصل على ال access

→ life time of a variable

\* يعني الوقت الى الى ال variable بيدرس خارج من وقت ما ينسلمه

destroy after it is define

→ visibility of a variable or Function.

\* يعني الـ visibility بـ الـ variable او متعرف ما فيه و

### C Storage classes



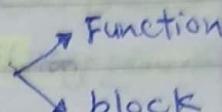
storage classes	Storage place	Default value	Scope	Lifetime
auto	RAM Stack	garbage	local (within block)	within Function (end of block)
extern	RAM [Data segment]	zero	global (multiple files)	Till the end of the main. Maybe declared anywhere
static	RAM [Data segment]	zero	local global	Till the end of the main
register	CPU general purpose register	garbage Zero	local	within Function (within block)

Auto:

→ are allocated memory automatically at runtime.

auto variable's memory will be deallocated at runtime.

initial value will be initialized to garbage value.

→ default storage → declared inside a  block

`int x; == auto int x;`

→ Visibility or scope

is limited to the block in which it is defined.

→ every local variable is automatic by default

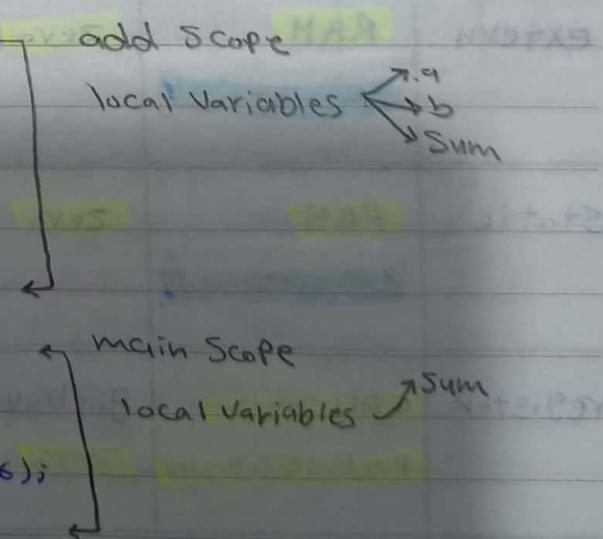
→ automatic variables can't be accessed outside their scope directly but we can do this by using Pointers.

→ automatic variables are initialized to garbage by default

Ex

```
int add(int a, int b) {
    int sum = a + b;
    return sum;
}

void main() {
    int sum = add(5, 6);
}
```



add Scope: `sum`  
local Variables: `a, b`  
main Scope: `sum`  
local Variables: `sum`



العنوان المسئولة عن المحتوى المكتوب

Func add ١١ المعرفة في main() var  
متعرف ك local و بالعكس في main() var  
و بالعكس دلائله في duplicate var  
كل var مختلف عن الثاني وبالعكس البرنامج يستعمل عارض

what is the output ? why ?

```
#include "stdio.h"
```

```
int main()
```

```
{
```

```
    int a=10;
```

```
    printf("%d ",++a);
```

```
{
```

```
    int a=20;
```

```
    printf("%d ",a);
```

```
}
```

```
    printf("%d ",a);
```

```
    return 0;
```

output → 11 20 11

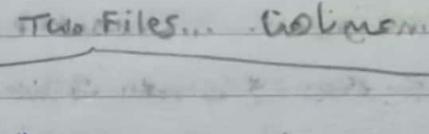
why, this variable is limited  
by entire block so  
it can't visible out the  
entire block.

Whenever try to print  
the value of (a) we take  
the last value of (a)  
in the main block

Static

- Static Variables can hold their Value between the multiple Functions call as it is stored in Data Memory
- Static local Variables are visible only to the Function or the block in which they defined
- Default initial Value of the static integral Variable is zero otherwise null
- the visibility of the static global variable is limited to the file in which it has declared.

Example

Two Files... 

```

main.c
#include "stdio.h"
static int x;
Void File2_Func1(Void);
int main()
{
    printf("main.c x=%d\n",x);
    File2_Func1();
    printf("main.c x=%d\n",x);
    x=8
    File2_Func1();
    printf("main.c x=%d\n",x);
}
  
```

```

File1.c
int x=1;
Void File1_Func1(Void)
{
    printf("File1_Func1 x=%d\n",x);
    x+=3;
}
  
```

outPut → main.c  $x = 0$

+ add + File1 Func2  $x = 1$

main.c  $x = 0 + 1 = 1$

File1 Func1  $x = 4$

main.c  $x = 8$

Why  $x$  static global var  $\rightarrow$  غير حاجة عينها  $\rightarrow$  بصفتها static

لما خبرناها مرة مطلع عينها  $\rightarrow$

بعدين عيناها اسندت لها func مع التغيرات الـ

موجودة في ملف تانى و بالعكس اسندت لها  $x$  static var

يتبين في الملف الثاني عينها هو مجرد بقى الـ main

فلم يدخل الملف الثاني لقيها متغير  $x$   $\rightarrow$  دراجة قيمه  $x$

فهي تطبع قيمة  $x = 8$   $\rightarrow$  وبعدين تزور قيمه  $x$   $\rightarrow$

فابعد قيمه  $x = 8$   $\rightarrow$  ولما نرجع الى main

تحصل في الملف الثاني مثل ما يطبع في الملف main

الـ  $x$  صو static global var  $\rightarrow$  فليطلع عينها  $\rightarrow$  صنعوا عينها سعى للناس قيمة

وبعد كذا هاكل اسندت لها func  $\rightarrow$  الملف الثاني وطبع

قيمة آخر  $x$  (الـ  $x$  بـ 4) وبعدين تزور القيمة

بملف  $x = 3$  خروج بـ 7 وبعدين لما ترجع الى main

هـ  $x$  تتحول قيمة الـ  $x$  من صفر لـ 8 فلما نجي

نطبع قيمة الـ  $x$  الموجودة جوا الـ main فليطلع

$x = 8 \rightarrow$  لينا

Conclusion The visibility of the static global variable

is limited to the file in which it has declared #

## extern

- External storage is used to tell the compiler that the variable defined as extern is declared with an external linkage elsewhere in the program.

معنىه أن بنكرف الـ compiler في الـ var هو متكرر في  
مكان آخر ونهى عنه declare هنا لأن تغير نسخة  
عن الملف ده عن طريق اتنا تغير كل الملفات به بالملف  
أول فيه الـ definition يتغير var

- The variables declared as extern are not allocated any memory
  - Default initial value of external integral type is Zero otherwise null
  - We can only initialize the extern global variable.
  - An external variable can be ~~not~~ declared many times but can be initialized at only once.

→ extern int x;

لواحتنا كتبنا كدا (Dynamic) مكتنلش File مكمول دفع

$N \times$  فده هیوئیتی انه بحص definition

undefined reference طبعاً عاليزین نجعوا مع linking error line  
الآن إفتح ملف واحد بس (۱۱۰) ؟

١٥٣- ملخص درس الگوهای مکالمه

الحالاتى هنفوص كاميلين definition لا X نفس الملف سواء هنكتب المترافق

حوادث main (ورثتها) الآتنين صح

definition  $\rightarrow$  int x;

(note)

لهم نفس الاسم بس كل  
وادر متعدد كـ 1089 في ملف دينليز كـ 1089  
ومعنى كلمة external قبل أحد واحد من الاثنين  
فهي بعض الـ compilers يتحملها بين  
الاثنين variables و تستعمل على أساس فرضها  
ونامح طول الكلمة external قبل واحد يعني  
بس كل المقادير الـ كـ 1089

Compiler dependent

(note)

[File1.c]

int x=5;

[File2.c]

extern int x=7;

لكن الـ var compilation error يعني او  
الـ initialization outside definition extern مرة

واحدة فقط في ملف واحد بس ،

register:

→ the variables defined as register are allocated the memory into the CPU registers depending on the size of the memory remaining in the CPU. [compiler dependent].

بعضها لو سوين في المايموري  
general purpose registers الباقي لهما غير الموصى بهم  
auto var. لـ Var. الـ CPU register  
CPU registers في المايموري

→ we cannot use dereference [&] with registers Variables  
لـ registers لا يمكننا استخدامه معهم CPU register لا يمكننا استخدامه معهم

memory address address address

→ The access time of register variables is faster than the automatic variables.

→ The initial default value of the register local → zero

→ we can not use more than one storage specifier with the same variable.

ex

register int a = 1;

printf("%d", &a);

output → compilation error

why because we can not use [&] with register variables.

الإجابة

لـ إيجاد

هي جدول متغيرات حلو

Recursion

is a situation happens when a function **CALLS** itself **directly** or **indirectly**

Directly

```
Void F1() {
```

{

```
F1(); };
```

{

عنوان فنك الـ **func**

نفسها (دورة)

Indirectly

```
Void F2() {
```

{

```
F1(); };
```

{

```
Void F1() {
```

{

```
F2(); };
```

{

func 1st calls func 2nd then func 3rd ...  
indirect recursion **func 1st calls func 2nd**

الحالات التي تحدث فيها overflow (أو ما يسمى **stack overflow**)

عندما يكون هناك دورة في كل دورة يضيف stack

فهي تأخذ جزء من stack الذي من الممكن أن يكون كبيراً

ويؤدي إلى احتباس stack

infinite loopNormal loop

```
#include "stdio.h"
Void PrintHello()
{
    printf("hello\n");
}
Void main()
{
    while(1)
        PrintHello();
}
```

Recursive loop

```
#include "stdio.h"
Void PrintHello()
{
    printf("hello\n");
    PrintHello();
}
Void main()
{
    PrintHello();
}
```

func 1st

Stack overflow due to loop

المقدمة في البرمجة

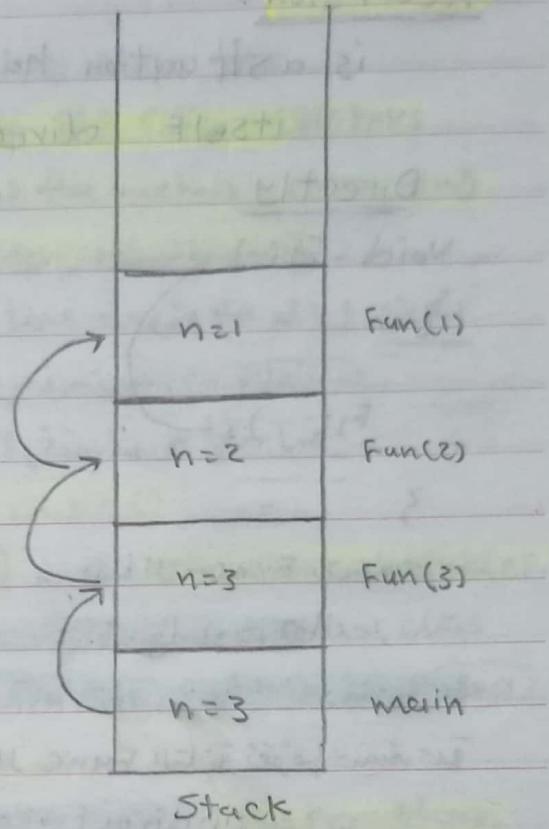
Example

```

int Fun(int n)
{
    if(n == 1)
        return 1;
    else
        return 1 + Fun(n-1);
}

int main()
{
    int n=3;
    printf("%d", Fun(n));
    return 0;
}

```



حال n=3 دلالة Fun() نcall دلالة main

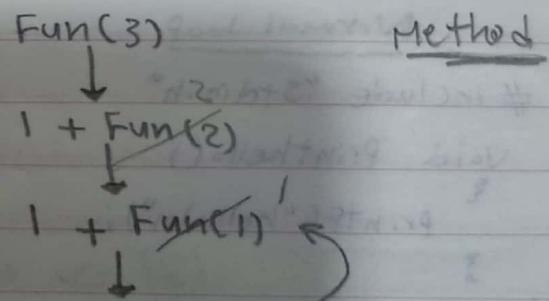
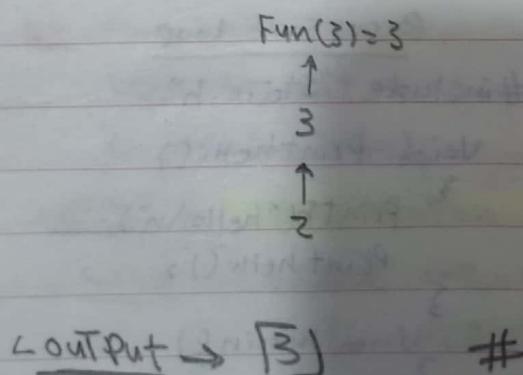
و n>1 False  $\rightarrow$  عبء شرط if متحقق Fun()  $\rightarrow$  دلالة main

يعنى دلالة Fun() تابع دلالة Fun(1)  $\rightarrow$  دلالة main

ار n م تكون بـ 2 و هي دلالة بعد دلالة if

تحقق ويصل بـ 1 وبدر دلالة return

ونكتها ونحذف دلالة Fun() من stack



What is the output?

```
#include "stdio.h"
int Fun(int n)
{
    if (n == 0)
        return 1;
    else
        return 7 + Fun(n-2);
}
```

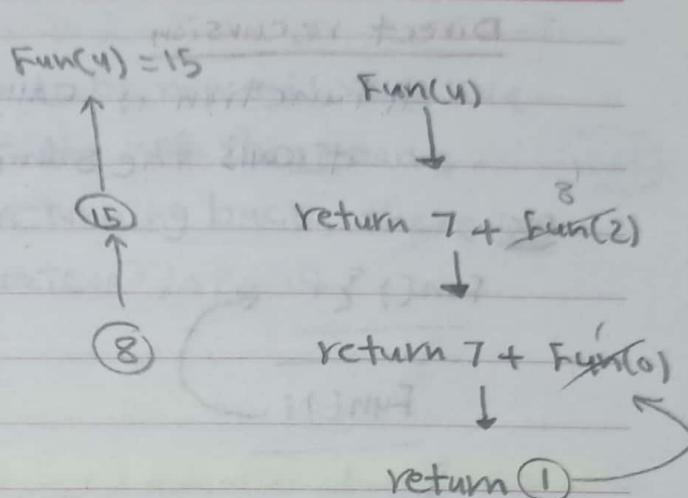
```
int main()
```

```
{
```

```
    printf("%d", Fun(4));
}
```

```
return 0;
```

```
}
```



| Output = 15 | #

## TYPES OF RECURSION

① Direct recursion

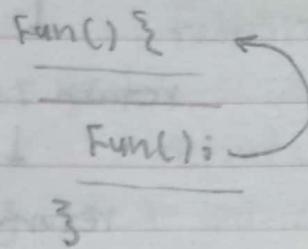
② Indirect recursion

③ Tail recursion

④ Non-tail recursion

Direct recursion

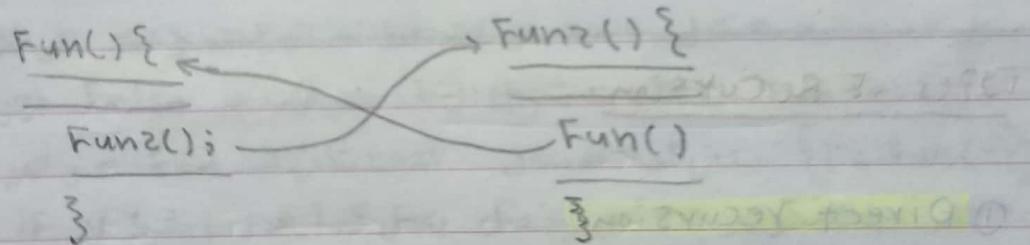
A Function is called direct recursive if it calls the same function again.

Ex

دالة فنكى تدعى `Fun()` ←

Indirect recursion

A Function (let say `Fun`) is called indirect recursive if it calls another function (let say `Fun2`) and then `Fun2` calls `Fun` directly or indirectly.

ExTail recursion

A recursive function is said to be tail recursive if the recursive call is the last thing done by the function. There is no need to keep record of the previous state.

الموضوع :

التاريخ

### Non-tail recursion

A recursive Function is said to be non-tail recursive if the recursive call is not the last thing done by the function. After returning back, there is something left to evaluate.

## Functions in Depth (ARM "case study")

(note) void MyPrint(void)

3

printf("Hello");

3

٤١. System call يدخل stack لـ Function do

عند دخول stack يدخل stack return address نحو

٤٢. فونكشن

يستخدم return address بعدها الـ func

ما هو المخرج من func return address

٤٣. برقم الـ func

(note)

### Format in ARM

•  $\rightarrow$  BL label (Branch with link)

BL  $F_1 \longrightarrow F_1 : \cdot \cdot \cdot \rightarrow Bx LR$  (Branch indirect)

•  $\leftarrow$   
•  $\leftarrow$   
 $Bx LR$

أول ما ندخل Function Call يتم حفظ

LR return address في الـ

(program count) PC وقيمة تغير متزنة في الـ

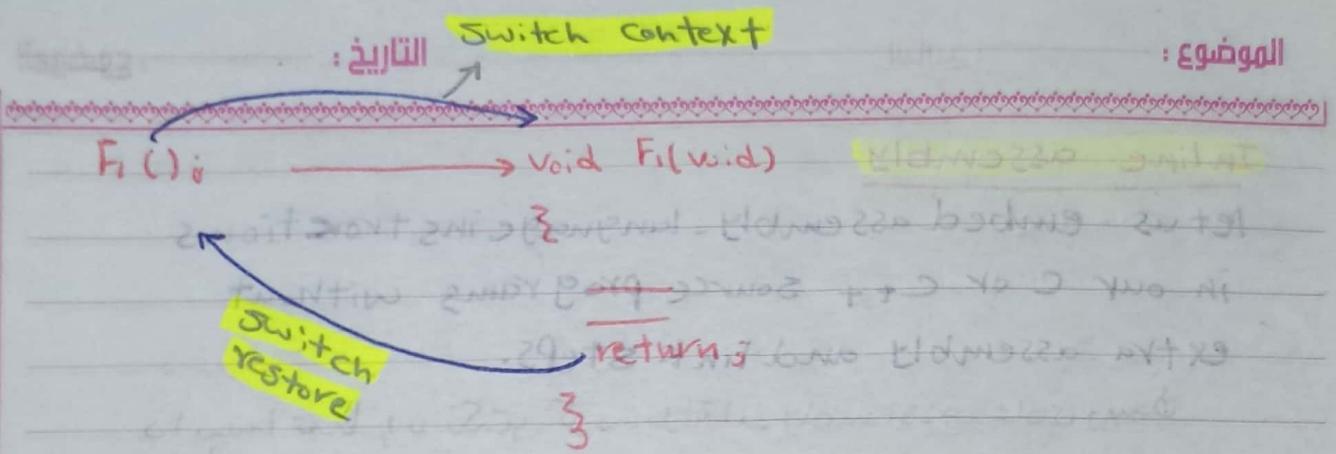
address address PC وتحل محلها PC هتغير العددة المضمنة في الـ

(address of label) FUNC أولاً سطر هي تنفذ في الـ

main كل سطور الـ FUNC صدور نرجو الـ

هنا هنا خلاص الـ PC يبدأ بروح

return address نـ



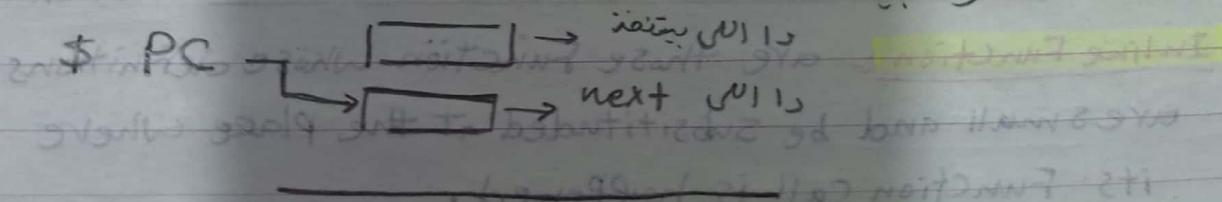
Switch Context → Function Call

الوقت اللي يعمر من أول استدعاء الدالة

Calling time  $\leftarrow$  ~~way for the birds' return to~~

switch Restore → Function exit

note) program count register هو寄存器 processor next instruction address (PC) وهو يحتوي على الـ address of instruction بعدها يدخلها processor



## Inline assembly

let us embed assembly-language instructions in our C or C++ source programs without extra assembly and link steps.

دابيوكس اسمايل كود بلقاير كود اسمايل

كود اسمايل ماسون ماين كود اسمايل

How to write it? ← extra notation

How to write it?

Using GCC:

...asm (" " int");

using VC++:

...asm { };

another shape:

asm { };

Inline Function: are those function whose definitions are small and be substituted at the place where its Function call is happened.

define it inanci C:

... inline int example (int x) { };

define it in C99 or C11:

inline int example (int x) { };

call يع داري، ياتلوكس Normal Func -> نسخة في < ياتلوكس  
مبني على حفظ LR N بدلوكس و لا يحاجب الموارد دلوكس.  
يتزداد او يتقلص ياتلوكس كل ما يكتب في code size < ضرورة  
call ياتلوكس definition اولاً