/ /

Scope and extent

ال functions بتكون by default ← extern من غير ما احط extern قصادها

مش محتاج اكتبها →
declration → extern int add (int, int)
defention → int add (int, int) {
}
الفانكشن نفسها

لو عندي متغير global وعايز استعمله في فايل تاني هستخدم
← extern

```
main.c                          test.c
int counter;
// defintion                    extern int counter;
  printf (counter) → 0                // declration
  add (      )                  int add (int, int) {
  printf (counter) → 1
                                  counter ++; }
```

لو اقدر استخدمها عادي

* لما يكون ال scope ← global اقدر اعمله calling من اي مكان في البروجيكت (محتفظ بقيمته) → shared resource وده عيب

* لما يكون ال scope ← static مقدرش اعمله calling غير جوه الفانكشن (محتفظ بقيمته)

* لو متغير global عملت ليه static مقدرش استخدمه بره ال file.

usages of * :

① declration of Pointer

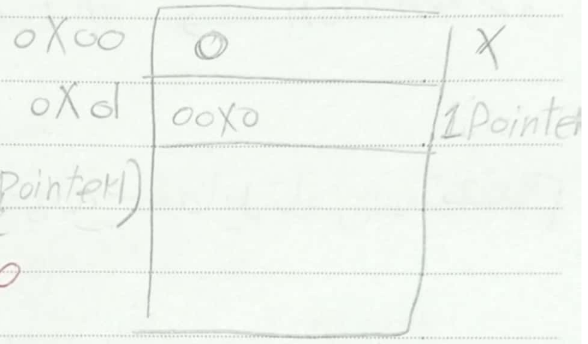
② indirect غير قيمة اللي بيشاور عليها

```
char X = 0
char *1Pointer = &X;
```

Pointer1

lcd-out (&Pointer1) → 0x01

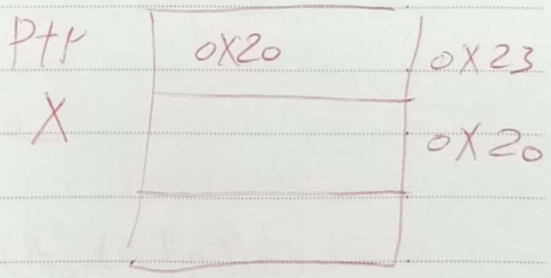
lcd-out (Pointer1) → 0x00



*1Pointer ++; → $\boxed{X=1}$

& → return address of a variable

```
char *ptr = &X
ptr ++ ;  → ptr=21
ptr -- ;  → ptr=20
```



casting

```
char X = 53;
int y = X;
cout<< X << endl; → 5 →
cout<<y; → 53
```

طبع القيمة اللي ASIC بتاعها

عبارة عن 53 وهي 5

* ال char لما بيجي يطبع بيتعامل معاه كأنه حرف فبيشوف ال ASIC بتاع الرقم

/ /

لو عايز أعمل casting

```
cout << (int) X ; ⟶ 53
Cout << (Char) y; ⟶ 5
```

Address casting

لو عايز أخلي القيمة اللي في العنوان 0X05 كلها واحد ← 0X44

```
*((Char* ) 0X05) = 0X44
```

الرقم ده عبارة عن address فأنا عند pointer من النوع char لـ address ده القيمة اللي هي 0X44

0X05 → address for one byte

Volatile:

أكتب في تعريف PortA volatile عشان أعرفه أن القيمة دي ممكن تتغير من hardware

```
#define PortA *((volatile char*)0X05)
```

```
int X = 0
  main( ){
  while(1) {
  X = PortA;
      while(X~=1){
         y++ }}
```

كده بالنسبة للـ compiler الـ X قيمتها مش هتتغير عشان مفيش ++ و أي حاجة بتغيرها جوه الكود فهيقوم شايلها من الـ loop

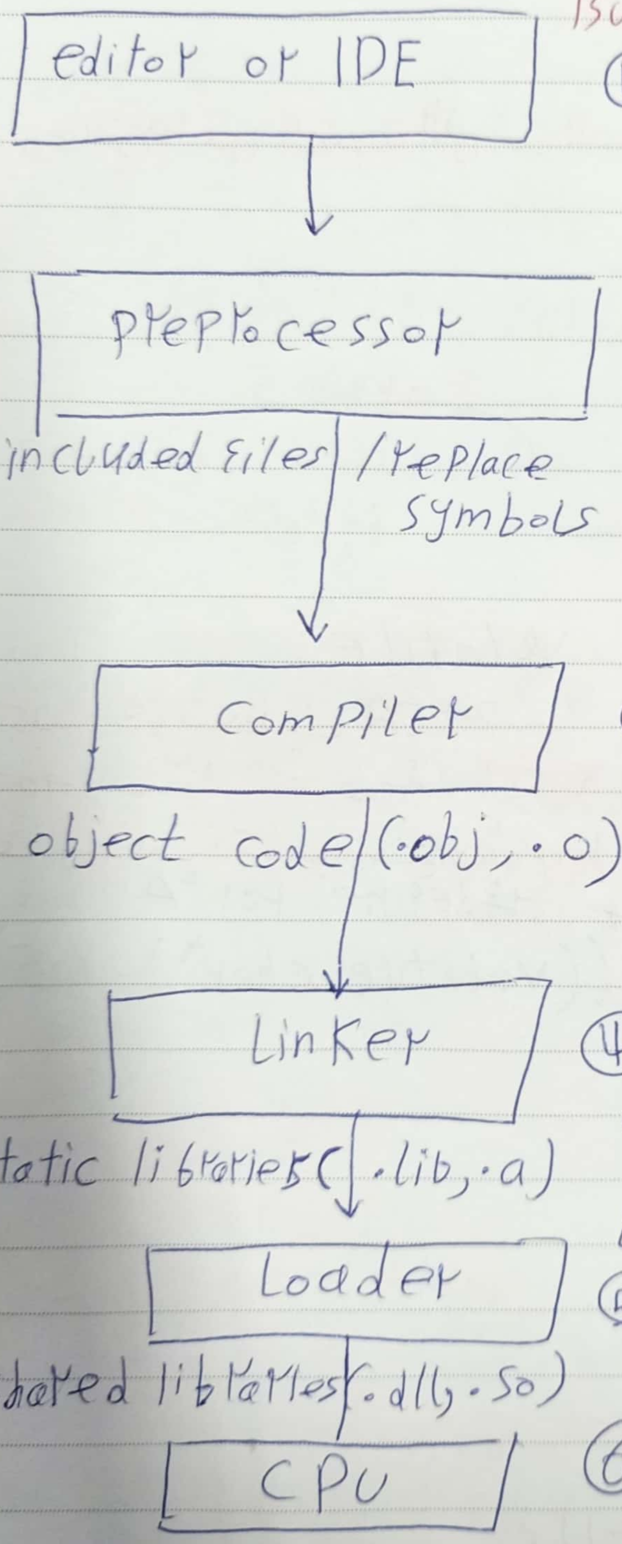
```
  main () {
≠ X = PortA
  while(1){ (whileX~=1) {
        y++; }
       }
```

error لأنها ممكن تتغير

Compilation Process in C:

تبدأ من أول ما أكتب Source file لحدما يطلع الـ execute code اللى

بيتنفذ من ① لحد ④ ← Bulid



editor or IDE
preprocessor
included files / replace symbols
Compiler
object code (.obj, .o)
Linker
static libraries (.lib, .a)
Loader
shared libraries (.dll, .so)
CPU

① write source code file.c / file.h

② - وظيفته يشيل #include ويحط مكانه الـ code اللى جوه الفايل اللى بعمله include
- بيشيل كل #define ويحط القيم بتاعتها فى الكود

③ - وظيفته ياخد الكود بتاعى ويطلعها assemble فايل

④ - فى حالة كان فانكشنز أنا بعملها calling وعملها declaration صح فهو يعلم عشانها ويدور عليها فى الفايلات لو ملقهاش يعنى كده مفيش definition ليها فيدينى linker error

→ executable (.exe) code

⑤ ⑥ → Run

*while writing multi-file Project each file is assembled individually into object files linker combines these object files to form final executable

$$\begin{tikzcd}
a.s \arrow[r] & \boxed{\text{assembler}} \arrow[r] & a.o \arrow[rd] & & \\
b.s \arrow[r] & \boxed{\text{assembler}} \arrow[r] & b.o \arrow[r] & \boxed{\text{linker}} \arrow[r] & abc \\
c.s \arrow[r] & \boxed{\text{assembler}} \arrow[r] & c.o \arrow[ru] & &
\end{tikzcd}$$

محتاجه الكود يبقى الـ runtime بتاعه ديما موجود ويكون ROM عشان ميتغيرش (.txt)

محتاجة الداتا أقدر أعملها write / read (.data)

يقسم الكود بتاعي لـ 2 sections → .txt , .data
ده من assembler

لو عندي كل file يبقى فيه .data / وسكشن .txt عايزة اجمع كل سكاشن .data مع بعض وسكاشن .txt كلها مع بعض وده هيحصل في Linker script file وظيفته يحدد إنها جزء من كل ملف هياخد يعني انا ممكن مخدش الفايل كله ؟ او لو كان عندي كذا فانكشن ومعملتش calling غير لاتنين فأنا مش هدخل غير الاتنين

بس كده لو احط الداتا في .data اللي هي هتبقى في RAM و أفصل الكهرباء وارجع تاني هلاقي القيم اللي عندي معدتش موجودة !!!

/ /

الحل إنى أعمل copy للداتا اللى موجودة جوه .data
من RAM واحطها فى Flash

وعولها Power up ← startup code will copy data from flash to RAM

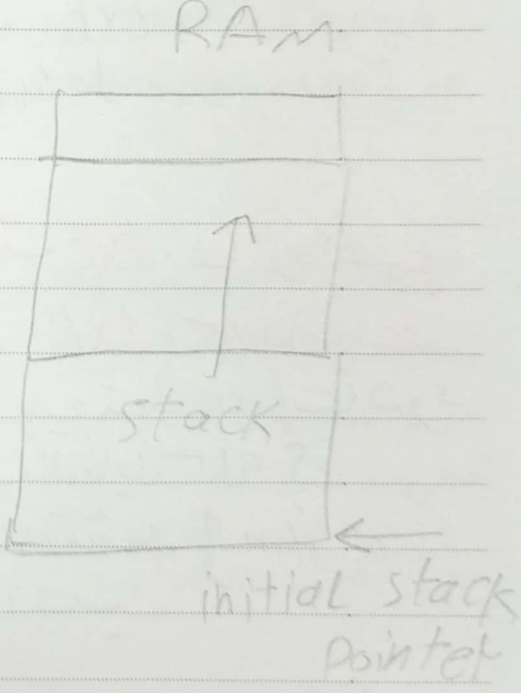
.data section has two address → load address → flash
→ runtime address → RAM

* load address → مش هروح لي غير من start up

① stack → local (auto) variables

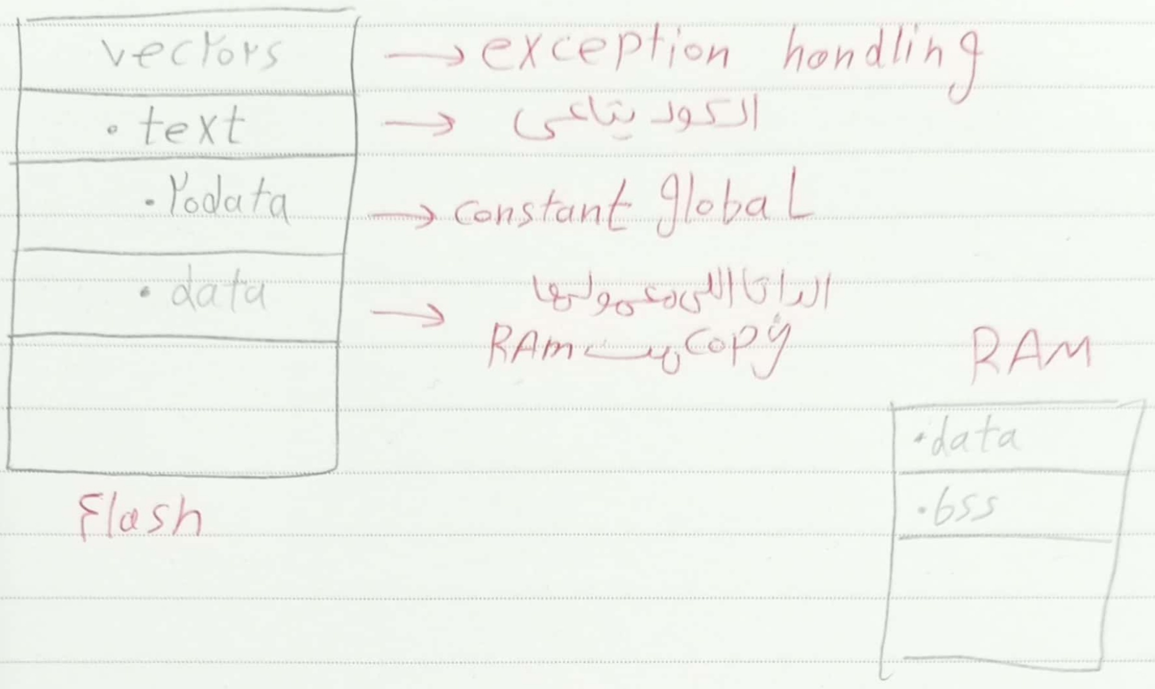
setup → we must initialized stack pointer to the bottom of the stack.

Ldr SP, =0XA4000000



② .bss global unintilized variable بيبقى فيه

أو ال variable اللى بصفر هتكون موجودة فى RAM ومش محتاج أعملها copy من flash عشان معروف إن المكان ده .bss كل ال variables اللى فيه قيمتها بصفر



startup code:

1. exception vectors
2. code to copy .data from flash to RAM
3. code to zero out .bss
4. code to setup the stack Pointer
5. branch to main