Subject: .... Date: ....

* We need variable/object to be provided once needed whatever

* لو عندي obj = static في فايل و obj تاني = extern في فايل تاني
بنفس الاسم والداتا تايب كدا هيضرب ايرور وملها يعفيد من الـ main
لان الـ static مينفعش اشاركها غير في الفايل اللي انا موجودة فيه
بس دلا بيكون عن طريق انا مش بيحصل لل obj الـ static
generate to symbol table.

* object → Provided (global)
  → needed (extern)
  → Not needed nor provided (static)

- memory sections:

Flash
0000 .vect (vector table)
.txt (My Code) startup code فيه جزء منه
.rodata (read only) (const global)
.data (intialized) global variables
.bss global variables (uninitialized)
بيجيلي عند الـ bytes كل الفاريبلز اللي مش متعرفة
وبيتكتب فيه انستركشن بالعدد ويروح للعدد كدا
دججر مساحتهم الحقيقية في الـ stack في الـ SRam.

SRaM:
.data global var inti
.bss global var uninit
stack local var & functions arguments
heap dynamic memory objects.

int arr[100] = {1, 2, 00, ...}; دا أرق لأنه أخد مني مساحة الأرأي دا اللي
هي 400 byte مرتين مرة في data في الفلاش ومرة في الـ SRam

المستفاد اياك تعرف global var تكون intialized والا
ابلع الـ rejected ♡

Subject:

Date:

تلخيص سريعة لـ الـ Linker:
1- object vervication
2- out put generation (Combination) : locator & Linker script مساعدة

شغلة مهمة بقى ان لما بحول out لملف asm فأنا بلاقى ان فى كود زيادة
اتضاف والـ addresses بقت حقيقية معدتش صفر وكمان فى حاجات
موجودة ابل Function main اللى المفروض هى بداية اى program
دا بقى هو startup code واللى

Note this:
main file

```
intmain{
    int x = sum(2,3) ;
}
```

sum file

```
int sum (int x , int y){
    return x+y
}
```

If I change to:
int x = sum(3) ;
↳ this will be built successfully !
but why ?

due to bug of not including file.h (header file)
So Linker checkless with (return type or arguments) it's lang. rules ! so It's jop of compilar

طب الحل هنا انى اعدل prototype كل مرة اعمل calling فى File
هاو حتى فى الفايل اللى انا معرفة فيه الفانكشن دى بس بدال التكرار
كدا هنوصل لعند header file انى هكتب الـ
declaration just once and include file.h in main file
in my case.

Note this : gcc -E → .i Preprocess بيوقف build
-S → Compilar only .S / .asm
-C → Compi + assymbler .o

Subject: .... Date: ....

③ linker + locator

↳ main.o + sum.o = hex file or .elif or .srec or .catt

• object verification : هل كل needed موجوده provided مره واحدة

زي obj

و ٢٥ ٢٥

مينفعش الاقي اكتر من intialization لنفس اسم ال function أو

linker error علشان ال var. intialize مرتين للفاريبل

في فايلين مختلفين لنفس البروجكت.

error disc : undifined refrence to (___)

يعني انا ملقتش الفاريبل أو الفنكشن دي كان ولا شيل منه اللي معمولهم include in project.

• every provied obj is Provided once

error disc : Multible difintion of (sum) as fun. مثلاً

Note: بيخرج فايل مع hex اسمه map بكون فيه كل تفاصيل المبوري

↳ How linker combine 2 files to out one hex File ?

main.obj	sum.obj	= File.hex
0 —	0 —	0 —
2 —	2 —	2 —
4 —	4 —	4 —
6 —		②+6 —
		8 —
		10 —
		12 —

لازم locator دخط الانستركشنز في الاماكن الصح فعليا جوا الرام او الفلاش ودا بيتم بمساعدة linker script عنده معلومات عن كل حاجه في الميموري عنده بداية ونهاية الرام وكل تفاصيل في الرام هل هو read بس ولا ريد رايت ولا الاتنين وهكذا.

Note This:

دي Sum الـ check اللي بيتعمل موجودة في فايل تاني غير الـ main وده دور الـ Linker.

sum | fun. | 2 | needed

Sum (Symbol table)

obj name	type	virtual add	Athority
sum	fun.	0	provided
x	int	2	provided
y	int	6	provided

3] assembler: out → main.o / obj
& Sum.o / obj } relocatable_files

* obj file (binary code)

في جوا الفايل دا حاجة كدا معناها ان الـ IDE اللي انا شغالة عليه او الـ tool اللي بيعمل حاجة اسمها dumping يعني obj ⇄ assymbly

ELF (excutable and Linking format)

Note: LR (Linker register) responsible for Couting the instruction storage in memory (act as dinamic memory allocation)

1- preprocessor: (text replacement)
• ↳ #difine, #if, #elif, #includ, #else, #prgma
↳ دى تبع ال compile مش preprocessor
output : intermidiate file (main.i)
• check preprocessor errors.

(Code generation stage)
2- Compilar main.i → main.asm / s / a (based on arch)
• check syntax
• Convert C file to assembly file.

Compilar consist of frontend - backend - optimizer (optional)
• tokenizer: recognize what is written (tokens: keywords - operators -)
• symbolic tree: get the syntax checked.
• temp generator assymbly (generate temperatory main-asmfile)
(front end stage)

* backend - stage
1- target generator out main.asm
أهمية انى عندى 2 asm files هى انى اطلع front واحد بقضيه الكومبيلر بتاعى
وا generate File بقى فى الباك اند خاص بالمايكرو اللى هستعمل
ال tool chain دا

2- Memory allocation
* optimizer: بيساعد انى اجيب الكود بأقل ميمورى وتيمر
لكن برضو ممكن حاجات تخلى output مختلف (بيمسح ال Regester allocation =>) (بيشيل ال dead code لو)
2- loop unrolling لو عندى لوب فلوب بعمل الحاجة اللى جواه متكررة بدل ميلوب
يعنى مش دايما بيكون احسن حل
* symbol table genarator: out symbol table.
ال code اللى بعد return
* register allocation (بخزن ال اللى بيتعمل read كتير)
* loop unrolling
Note: بنستخدم قيمة ال Regester allocation
volatile علشان ال optimizer ميعملش كده ... stack ... ده الا optimizer ... للقيمة ..

Subject:

Date:

Notes in

How to write Code c from scratch without ide's ?

- Tool chain (types, difinion, building process
- make file - linker script - startup Code
- Booting sequence - Boot loader - memory layout (load / runtime location)
- gdb debugger (Commands - Curcuit)

Bare metal app

OS app

Em. with

* toolchain:

↳ preprocessor + assymbler + linker + debugger + Libraries (Cross com.)

+ helper programmes (Binary utilities "obj copy - objdump - readelf.)

↳ types

(1) Native in PC output runs on PC.

(2) Cross Compile code in PC Runs output in target (AVR - ARM)

systems: (in toolchain)

↳ Build system (source code of toolchain)

↳ host sys. (gcc cross compiler)

↳ target (out file.out)

* CMD gcc:

- gcc main.c → a.exe
- gcc main.c -o main.exe → main.exe
- gcc main.c -I -> I/.. (Path)
- gcc -wall main.c → enable warning
- gcc -werr main.c → enable errors.

* Pass options using file

↳ gcc main.c @ options file

- gcc --help

Subject: (strinize operator)

Date:

[5] stringification & Concatination (taken - Pasting operator):

= " "

↳ #define printf(x) printf(#x)
printf("x") ل printf(x) يحول مثلا ل
String يحول ل

↳ Concatination : #define Conc(x,y) x##y
↳ port driver بنستخدمه في

NoTe This: #define max(a,b) a>b?a:b
int main(){
int x=9, y=7, z;
z=max(x,y) * 2
↳ z=a>b? a:b * 2
↳ Then z=9

If #define max(a,b) (a>b?a:b)
↖ High precedence

Then Z= 9*2=18

Note this #Line 20 الخطأ اللي بعدها , هتبدأ من 20

⇒ # pragma (compiler directive)
↳ optmize (" ", off) : No optimize
↳ once = replace file guard "بتقراها مرة واحدة بس"
↳ startup [priority] 64 > 65
↳ exit [priority]
↳ optional [64:255]
} ? بردو مثال هنا مفهموش

* predefined macros :

__DATE__ , __File__

[3] Conditional preprocessor directives :

* #if #elif #else #endif دي وظيفة ال preprocessor

[1] we can comment code by this way it will never go to compile

↳ (the unnecessary elif _ else _ if _)

If	# If
compiler	preprocessor
check variables	check numbers not variables

usage :

① comment "Multiline" #if 0

! I am mariam I'm tired

#endif

② Configration to driver

* #Ifdef , #ifndef لو انا معرف الماكرو دا ال آه ال def لا ndef

لو معرفه هوصل كدا لو لا #else

* #If defined , #elif defined (Multi defined check Cond.)

ex : #if (defined x) && (defined y)

#endif

④ #error & #warning directives

↓ stop compile ↓ only warning

↳ used with #if not if

Subject : ....

Date : ....

⑤ what is the diffrence Between typedef / macro

* typedef	# define
by Compilar	by Preprocessor
used with types	used with values
• Limited to give symbolic names to types	• # define ptr struct stud *
	ptr x, y ; → struct variable. ↓ struct pointer

• typedef structstud *ptr ;

↳* ptr m, n " m, n are pointers From type struct stud"

[3] Function Like macro (parameterized macro)

Ex # define add (x, y) x+y

↳ Note this what If I do #define add (x,y) x+y

If I had int z = add (3, 4)

replacement ⇒ (x, y) (3, 4) which later is error

* increase Code size which affect memory size and no type return check but time (speed)

↳ in embedded we use macro in file BIT-MATH.h

[3] (...) means anything else

بمعنى ان اى حاجة مكان (...) بتتخزن جوه (__VA_ARGS__)

Ex # define Mariam(...) printf(__ ~~VA~~ ARGS __)

```
int main( ) {                VA
    Mariam( Hello World );
}
```

[4] # define Fun (a, ...) printf(__VA_ARGS__)

```
main( ) {
    Fun (x, "x = %d "); ≡ printf ( "x = %d ", x );
}
```

Subject: Date:

Notes in preprocessing :

① #define → obj Like macro → #define obj replacement list
→ #define obj(...) replacement
→ Function like macro → #define obj(parameter,...) replace

① object like macro

#define name value (السرع من var.)

Note: ① not recomended لأنه بطىء ; بالنسبة للذاكرة

replacement with value فقط

② 1. #define x 10
2. #define x 20 > will give warning

!!! to avoid the waring Put #undefine x before the new x val

ex
#define x 10
int y = x + 10 ; # y = 20
#define x 20
int z = x + 10 ; # z = 30

③ (.c) file No. = (.i) files No.

④ what is the diffrence between enum and #define

enum	#define
. txt replacement in compilation stage	. txt replacement in preprocessor stage
. must be constant	. Can have value Float / Sentence ...
. Space = int size (4 byte)	. No memory space
. Can use in switch	. Can't be used in switch "due to float"
. diffrent names & values	. one name = value