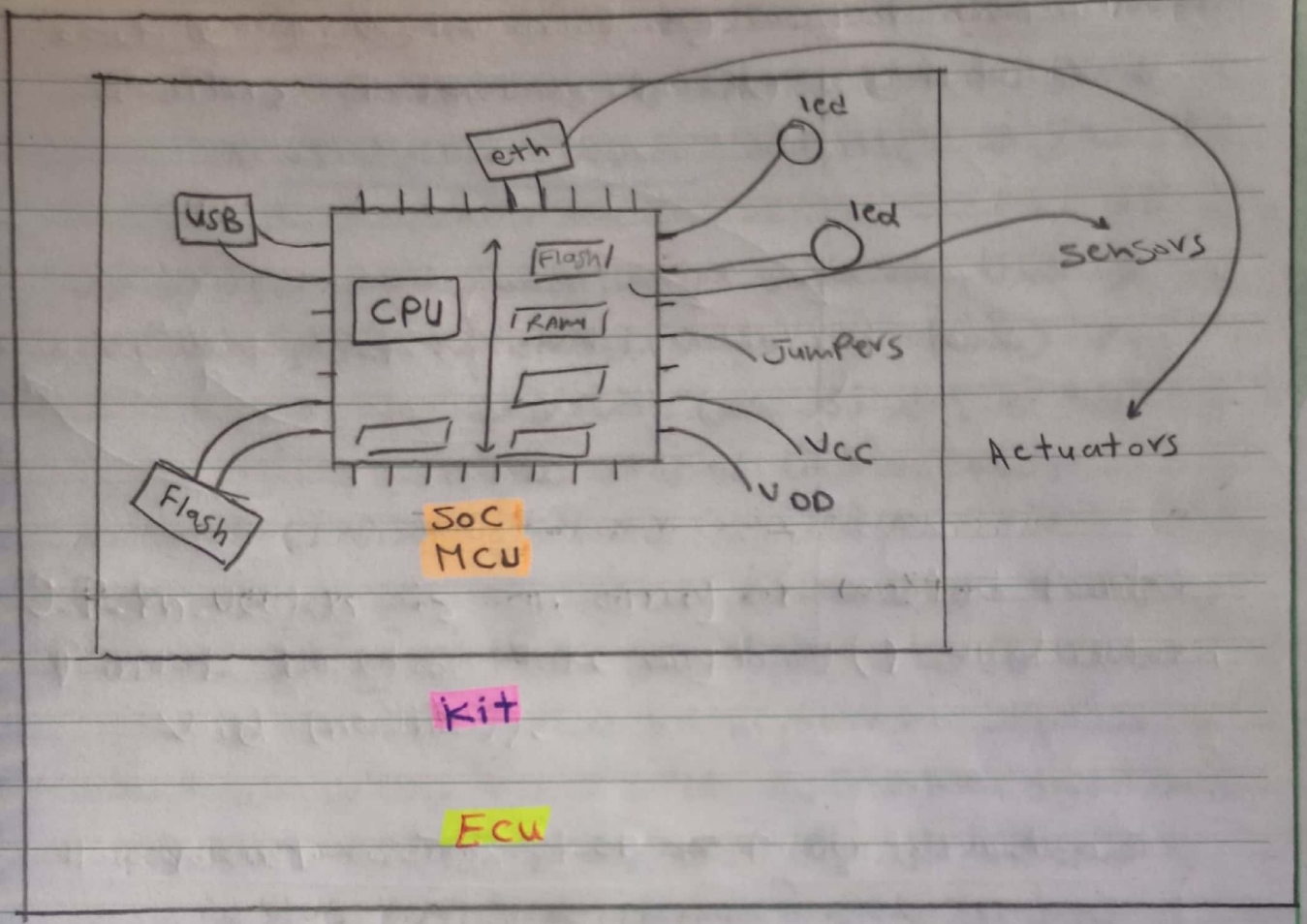


SoC ⇒ System on chip

Mcu ⇒ Micro controller unit

دى chip بتبقى موجودة على ال kit ولما نجى نبرمجها
هتختلف كل حسب نوع الى محفوظة وكل واحد ليها Trm
أو specks مختلفة عن التانية

Trm ⇒ technical referance manual

kit ⇒ دى بتحتوى على ال SoC ومتوصل بيها حاجات زى
ال led أو ethernet أو USB أو ——

Ecu ⇒ electrical control unit
دى بتبقى فى العربيات وبتتحط فى أجزاء العربية وبتتكون
من kit + sensors أو —— عشان تؤدى وظيفة معينة.

lab3 → write a baremetal SW to toggle led on PF3 which is connected with green led.

Kit → TivaC
SoC → TM4C123
processor → Cortex-M4

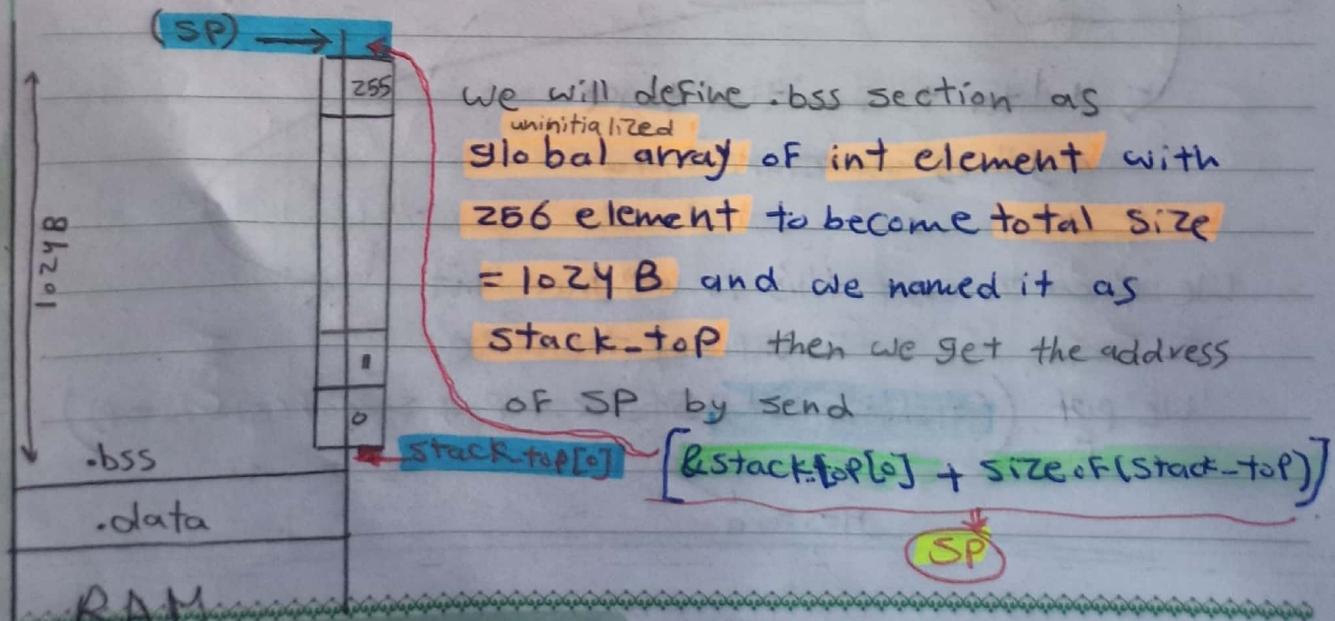
يبقى فيه نفس خاصية الـ Cortex-M3 وهي إننا لازم نحط entry point اسمه (address of SP)

in this lab:-

- stack_top should define in startup.c not define by linker script

- define the vector array by array of constant pointers to function takes () and return Void.

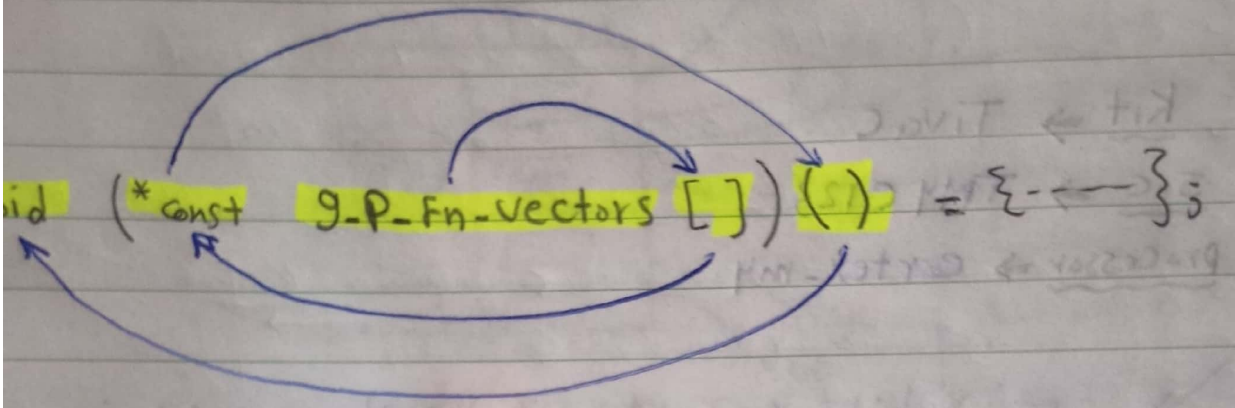
- define stack in startup with size 1024 B without using stack_top symbol at linker



التاريخ :

الموضوع :

define the vector array by array of constant pointers to functions takes() and return void.



g_p_fn_vectors is an array of constant pointers point to function take anything () and return void (no thing).

How to cast address of SP to pointer to function ??

(stack_top + sizeof(stack_top))

This address will cast to pointer to function

(void (*const)())

(stack_top + sizeof(stack_top))

pointer to function take anything and return void

We need to cast this to unsigned int to sum it with value of the size we put ((usint32_t) stack_top + sizeof(stack_top))

Mostafa

التاريخ :

الموضوع :

Debugging Mechanism:

How To debug the lab on keil ??

- open keil file
- open Project tab and choose options for target 1
- choose output and select the folder that has the executable file
- copy the name of executable and paste it in name of Executable
- open Debug and choose Simulator.
- start Debugging #

الموضوع :

التاريخ :

Debugging linux Application running on Embedded linux.

في شكل تاني في الإسلايدات

التاريخ :

الموضوع :

في الحالة اللي عندنا دي إحنا عندنا بوردة عليها linux OS كـ ورطنا ~~عليها~~ عليها gdb server وبعدين حطينا الـ application اللي إحنا كاتبينه فاحنا هنعمل connect بين الـ gdb server اللي موجود على الـ SoC ومكان الـ application والـ gdb client الموجود على الـ host machine هنعمل لهم connect عن طريق أي protocol وبالتالي إن الـ host machine موجود عليها بردو الـ application بتاعنا

مثال زي ما كنا عملنا لما جينا استخدمنا gdb debugger كنا كتبنا ← target remote localhost:1234 هنا إحنا كنا بنوصل الـ gdb server مع الـ gdb client عشان نعرف نـ debug الـ application بتاعنا اللي إحنا كاتبينه.

طب فرضنا لو إحنا عايزين نـ debug الـ linux kernel بأي حاجة بترن على الـ CPU بتاع البوردة مش بس أ debug الـ application اللي موجود فوق الـ OS لوحده ؟؟

في الحالة دي هنكون debugger circuit عشان نقدر نتحكم في الـ CPU أما الحالة اللي فاتت دي فاحنا بنتحكم في الـ gdb server وعن طريق بنقدر نعرف حاجات الـ application المحطوط على الـ OS وبالتالي في الحالة إحنا محتجناش نستخدم debugger circuit

الخلاصة لو أنا عندي بوردة محطوط عليها linux كـ OS وإحنا حطينا application عليه وعايزين نعمل debug ليه في الحالة دي إحنا مش هنحتاج debugger circuit وفي الحالة دي الـ gdb server هو اللي هيضبط الدنيا.

الموضوع : التاريخ :

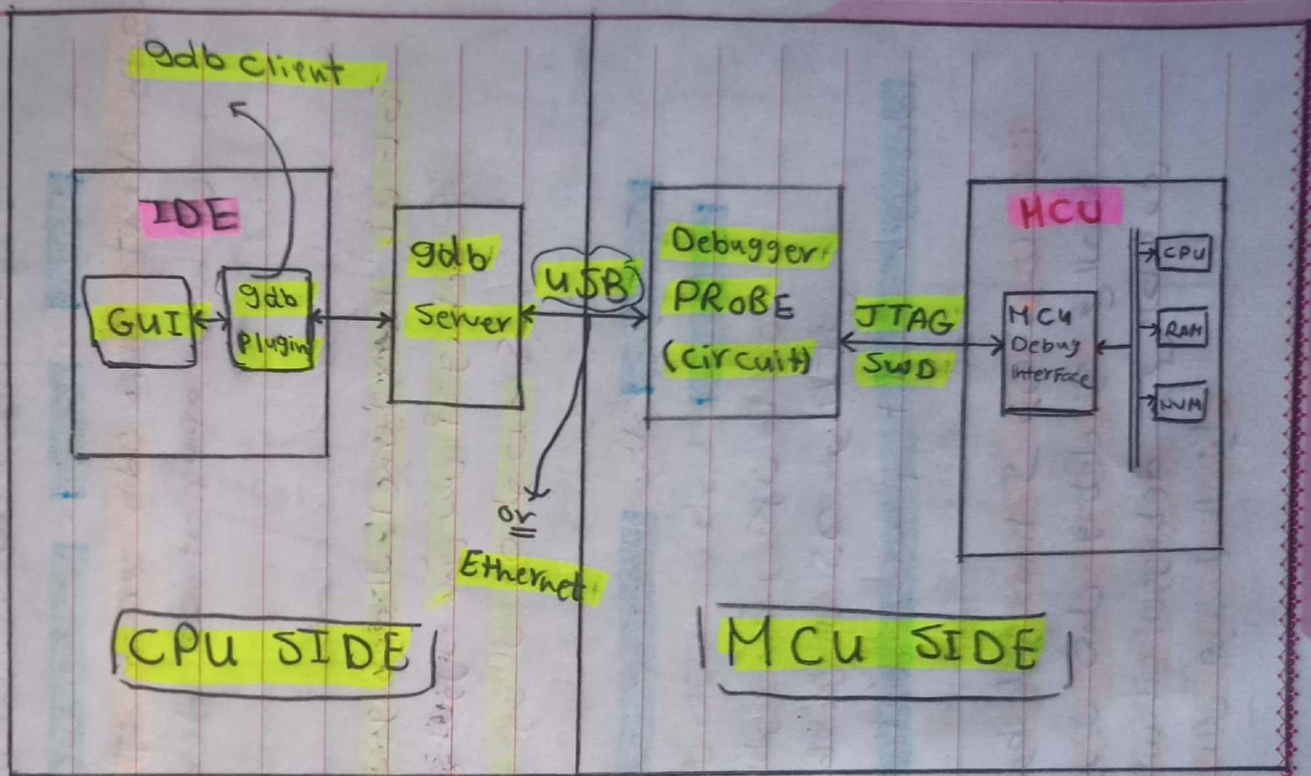
if we need to

Debug all SW which run on top of the CPU (may be the OS itself or baremetal SW or the bootloader)

in this case we need :

Debugger circuit

may named → debugger emulator
→ debugger adapter
→ debugger Probe



دا إحنا هنوصل الـ debugger circuit اللي عندنا بالـ MCU (SoC) عشان
نقدر نتحكم في الـ CPU ونتحكم في أي حاجة محطوطة على البورة
طب هنوصل عن طريق ايه ؟؟ في عندنا وصلتين يا إما
عن طريق JTAG أو SWD (Software debugging wire)
وبعد كدا هنوصل الـ debugger circuit بتاعتنا للكومبيوتر
عن طريق USB أو ethernet

الموضوع :   التاريخ :

ال debugger circuit هيتوصل بال gdb server عن طريق ال USB أو ال ethernet

طب وظيفة ال gdb server إيه ؟؟

بيحتوى على ال interface الخاص بال debugger circuit

أى debugger circuit بيبقى ليها بعض ال drivers الخاصة بيها + SW أقدر من خلاله أتحكم فى ال debugger circuit اللى هو بيبقى عندنا ال gdb server .

فى بعض ال gdb server بتبقى free ومن اللى بفلوس

زى free ← Open OCD

بفلوس زى ← Lauterbach debugger
دى مشهور وبتستخدمه شركات العربيات واسم ال SW الخاص بيه trace32

طب إيه الفرق مانستخدم أى واحد فيهم ؟

لا مش هينفع عشان كل gdb server بيدعم بعض البوردات المعينة وبعض ال processors وبالتالى على حسب ال Processor & Board هنختار ال gdb server اللى ينفع معانا ونعرف نستخدمه فى إننا نعمل debug للبوردة

الموضوع :   التاريخ :

الخلاصة هنا لحد دلوقتي إننا لو كانرين نشترى debugger circuit
عشان ن debug بوردة معينة فاحنا أول حاجة نشوف
هل ال debugger circuit نقدر ن debug بيها
البوردة الي معانا ولا لا طب ماشي بعد كدا ايه؟؟
نروح نشوف ال gdb server الي بي support
ال debugger circuit دى هل موجود واحد
ينفع نستخدمه لو موجود يبقى حلو ننزل
ال drivers وال Configuration Files بتوعه
ونشتغل طب لو مش موجود حاجة Free ندور
على gdb server يكون بفلوس بس ميكونش
غالى أوى وإلا هنكون اشترينا ال
debugger circuit ال الفاضي عشان مش هنعرف
نجيب ليها gdb server وبالتالى مش هنقدر نستخدمها.

في بعض الشركات بتنزل ال debugger circuit ومتلقيش أى
gdb server من ال Free يعرف يشتغل معاها فساعتها
هتضطر تشترى gdb server فتكون الشركة دى
كاملة واحدة وعارضه للبيع وبتكمل معاها IDE
عشان تعرف ت debug ويساعدك إنك تعمل كدا
بسهولة وعشان تزود سعر البيع برضو ☺

في بعض الشركات التانية بتصنع ال kit وتحط عليها ال SoC
وتحط معاها debugger circuit نقدر من خلاله ن debug
ال MCU فده هيسهل على الشخص وهيخليه ميروحش
يشترى debugger circuit و ي debug على طول
مثال ← شركة stm أو kit بتاعت Tivac

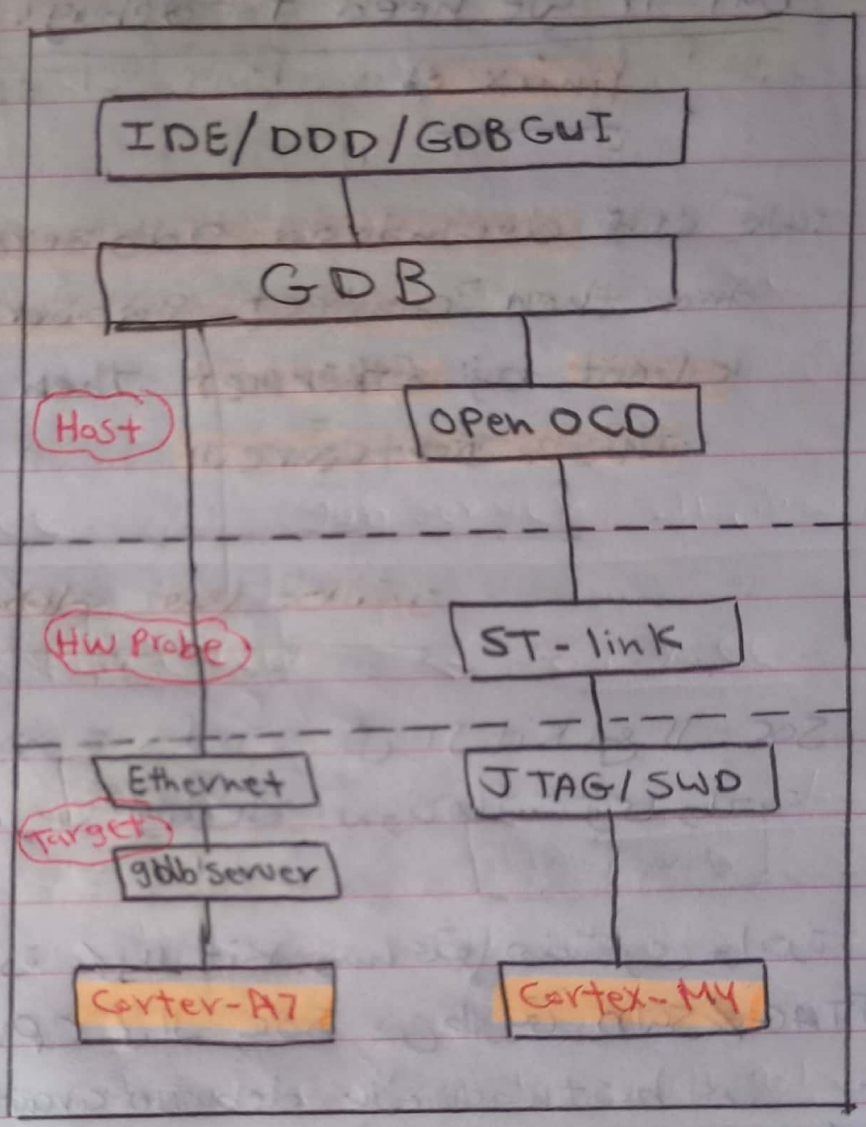
Gdb setup paths in soc which have different cluster

different CPUs

This board has two CPUs < Cortex-M4 / Cortex-A7

Cortex-A7 we put on it linux as OS and we put our application to debug it. So we put gdb server on the linux To debug the application

Cortex_M4 we can't put gdb server on it because we will need debugger circuit to debug the application on it and we will connect cortex_M4 with debugger circuit "ST-link" by JTAG or SWD wire and we put open OCD on host machine to connect it to ST_link by USB and we should download drivers of ~~of~~ ST-link on host machine to know it when we connect by USB then we open gdb client that attach with open OCD then we debug by DDD or gdb Gui (like eclipse) or author IDE. or by terminal.



الموضوع : التاريخ :

in case of cortex-A7 we can debug it by using st-link as we do with cortex-M4

but if we need to debug SW put on linux :

We will download gdbserver on cortex-A7 and then connect gdbserver with gdb client by Ethernet then we can debug this Software.

ال circuit debugger فيها حالتين

(1) إنها تكون موجودة جاهزة على ال kit مع ال SoC وبالتالى نقدر ن debug بدون مانشترى واحده

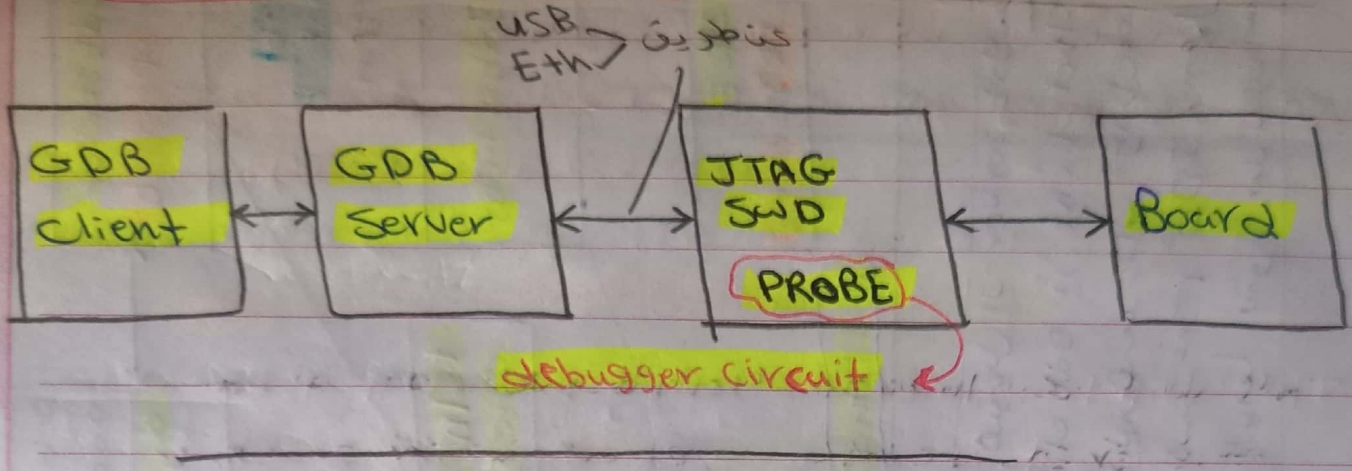
(2) إنها مش موجودة على ال kit وساعتها هنشترى واحدة ونوصلها بال cpu فى ال SoC عن طريق JTAG أو SWD ونوصل ال debugger circuit بال hostmachine عن طريق USB أو ethernet وساعتها هتكون واخده بال gdb server اللى هيكون إحنا منزلينه ومنزلين drivers عشان تتعرف على ال debugger circuit وال gdb server ده ممكن يكون free أو بفلوس على حسب نوع ال board اللى معانا هنشترى واحد يكون بيدعمها وبعد كدا ال gdb server هنوصله بال gdb client ويكدين نبدئ نـ debug بـ DDD أو GDBGUI زى eclipse أو ممكن عن طريق ال terminal أو أى IDE

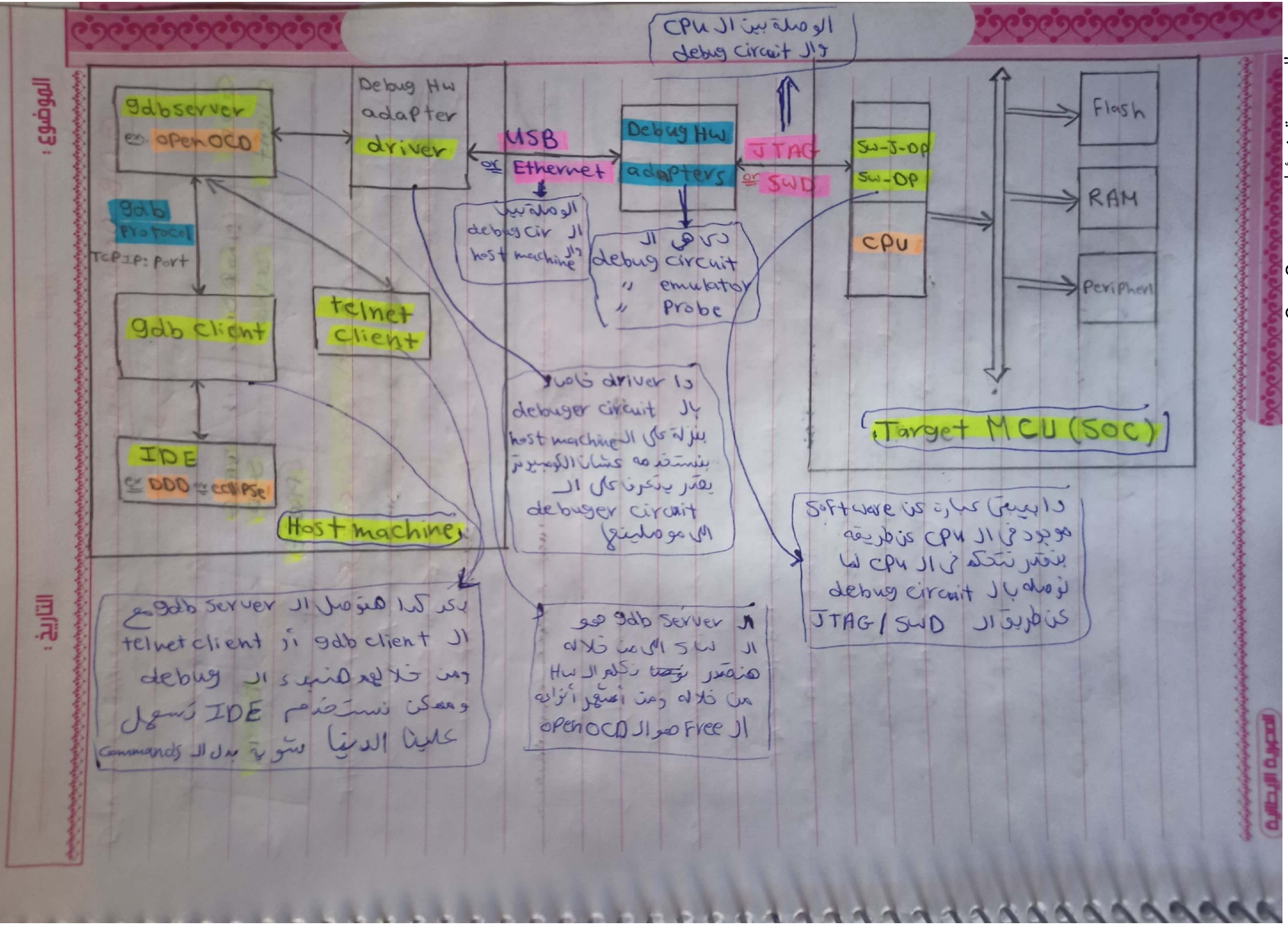
وبس كدا يا سيدى .

التاريخ :

الموضوع :

GDB Debugging chain





التاريخ :

الموضوع :

gdb server to gdb client or telnet client

Telnet to openOCD

telnet localhost 4444

initialize the init script — init

reset the board — reset

Halt the cpu (وقفه) — halt

load an image on Flash — load_image <elf image>

Finish up — exit

GDB client to openOCD

أى أمر عايزين نكتبه من الـ gdb client لـ الـ gdb server

لازم نكتب قبله كلمة monitor

لو عايزين نـ load الـ executable اللى إحنا بعتينه مع الـ arm-none-eabi-gdb

بدل الموجود على البوردة هنستخدم أمر ← load

لو عايز أعمل restart للبوردة بعد ما حرقت الكود الجديد

عليها هنستخدم أمر ←

monitor reset init

نقدر بقا نستخدم الـ Commands اللى اتعلمناها قبل كدا مع

الـ gdb client ونكمل زى ما إحنا عايزين.

التاريخ :

الموضوع :

Dynamic Allocation:

Static Memory Allocation :-

ex: `int x[3] = {7,3,2};` (local variable)

This will translate to assembly and tell the compiler to reserve a place in the stack with size ⇒ 3 * sizeof(int) = 3*4 = 12Byte and this place will be Fixed.

7	[0]
3	[1]
2	[2]

Memory used For this is (Fixed)
this means we know size of memory and it is calculated before runtime.

ex: `int x[3] = {7,3,2};` (global variable)

This global variable will save in .data section if it is initialized like as inour examble and if it is not initialize it will save at .bss section and it will initialize with zero by startup.

if it is local it will translate to assembly and this will save in .text section and it will execute at runtime but the memory is still Fixed and calculated before run time.

So in our example the memory is calculated before run time and it is fixed.

(.data) save in Flash and copied to RAM at run time.

(.bss) save in .bss by startup.

Memory Allocation (static)

الخلاصة :

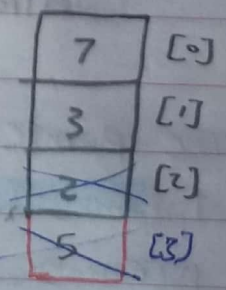
- is calculated at Compiling
- is fixed

Why static allocation is fixed??

because if we reserve 12B for .data section we can't increase size of Memory of it to 20B at runtime and we can't decrease it as so on.

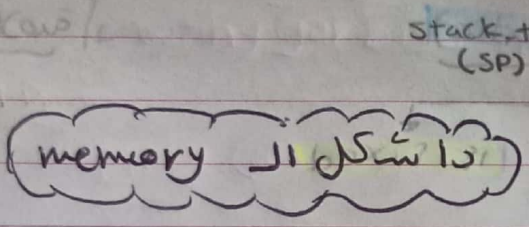
مش هينفع بسبب إن .data مكانها اتحجز في مرحلة الـ Compilling والـ linker ضبط مكانها والـ startup عملها copy من الـ Flash لـ RAM والـ size بتاعها ثابت مقدرش أغيره يعني مقدرش أكبرها بإضافة حاجة أو أصغرها بأن أمسح منها حاجة.

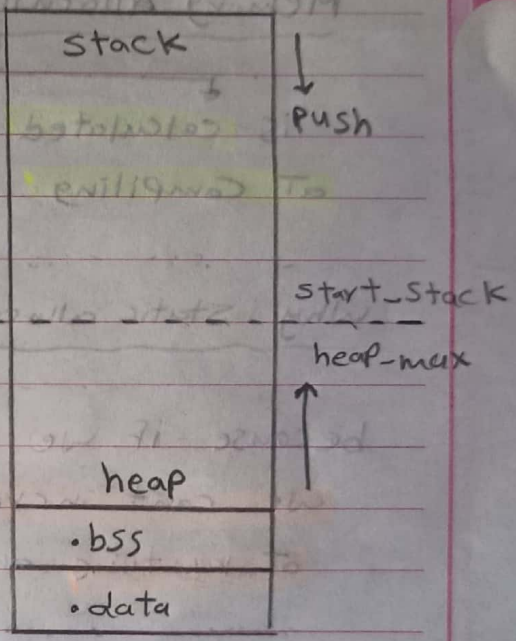
- we can't add new element "[3]" at run time.
- we can't remove any element "[2]" at run time.



Dynamic Memory Allocation :-

in it we can reserve a place at runtime on heap and after we Finish using this place we can release it by free.



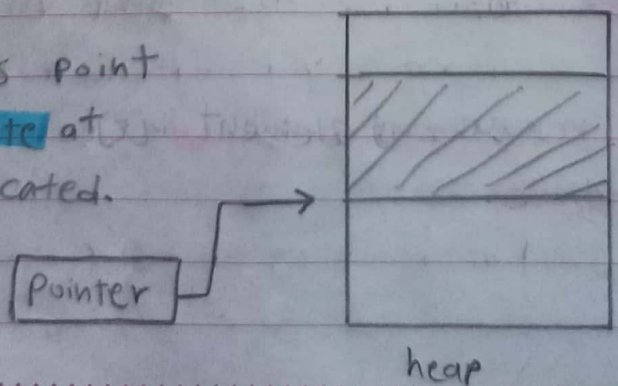


خد بالاً إننا لو محددناش maximum لل heap فهو لو إننا فعلاً نحجز dynamic at runtime كتير جداً هيكمل ويدخل على ال stack وممكن ساعتها يحرق data من ال stack فلازم نحدد heap_max ولو كدا ساعتها يرجع لينا Null.

- we put limit For the heap and if we skip it then it returns Null.

The only way to access the dynamically memory allocated is through pointer.

The pointer is point to first byte at memory allocated.



Comparison:

Static Allocation	Dynamic Allocation
performed at static or compile time.	performed at dynamic or run time.
assigned to stack	assigned to heap.
size must be know at compile time.	size may be unknown at compile time.
First in last out. (LIFO) (last in first out)	No particular order of assignment.
It is best if required size of memory known in advance.	it is best if we don't have idea about how much memory required.

Memory allocation not exist in Standard C

They are in "stdlib.h"

malloc() ⇒ allocate requested size of bytes and returns a pointer to first byte.

calloc() ⇒ allocate space for an array, initializes to zero and the retuh pointer to memory.

free() ⇒ deallocate the previous allocated space.

realloc() ⇒ change the size of previously allocated space.

Malloc() :-

reserves a block of memory of specified size and return a void pointer (which can be casted into pointer of any form) to first byte of memory allocated. The memory allocated is initialized with garbage value.

Syntax :-

```
ptr = (cast_type *) malloc(byte_size);
```

ex ⇒ `ptr = (int *) malloc(100* sizeof(int));`

Calloc() :-

it works as malloc() but there is two difference between them :-

1. calloc() take two parameters one for number of element second for size of each element.
2. calloc() initialize the memory allocated by zero not garabage.

Syntax :-

```
ptr = (cast_type *) calloc (n, element_size);
```

ex :-

```
ptr = (float *) calloc(25, sizeof(float));
```

Free() :

use to release space . (free the memory).

Syntax : free (ptr) ;

ex ⇒ free (ptr) ;

after we free the memory the pointer will be dangling pointer because it is point to deleted memory and we should make pointer point to Null after free memory.

realloc() :

it ~~you~~ is use to change size of memory previously allocated.

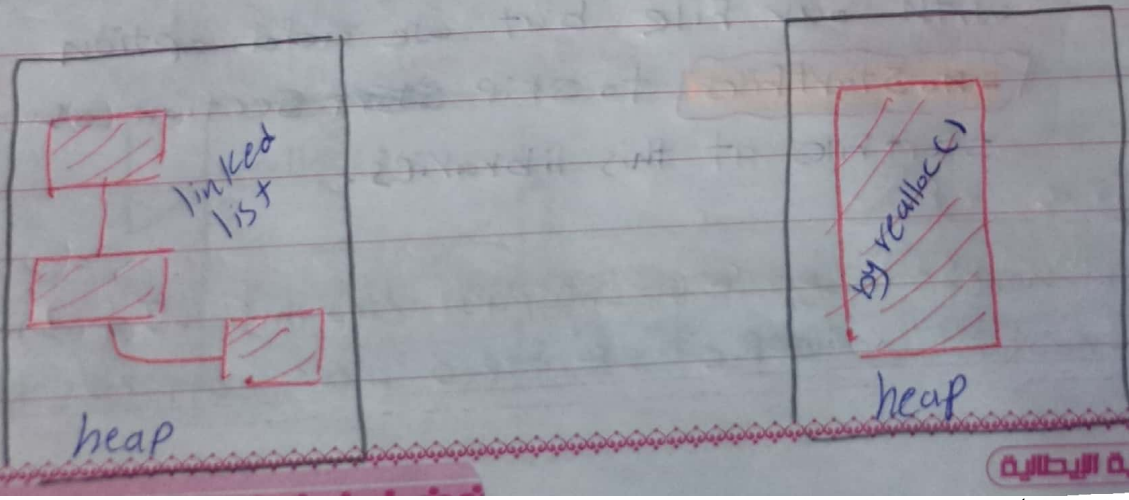
يعني مثلا لو كنا حاجزين ١٠٠ بايت وكايزين نزودهم لـ ١٥٠ بايت فساعتها هنستخدم realloc() في الدوار ده.

Syntax : ptr = realloc (ptr, new_size);

↳ This is pointer to reallocated memory

في حالة الـ realloc()

الـ memory اللي بتتحجز بتبقى عبارة عن block واحد ورا بعضه على عكس الـ linked list ممكن كل element فيها يكون في مكان مختلف



What happens if we try to use dynamic memory allocation in Embedded C ??

if we use (lab2 in lesson3) To practice init.

if we use malloc() in app.c without including stdlib.h what happens??
this will give us error because it is not know declaration about malloc() or Free.

if we include stdlib.h What happens??
the declaration error will deleted because stdlib.h has prototype "declaration" of malloc() but linker error will apear because we don't have definition of malloc() and it's definition exists on stdlib.o and if we know the location of it we can pass it to linker by:- -L <path> arm-none-eabi.stdlib.o
path of stdlib.o
but we can't know path of stdlib.o So we will use arm-none-eabi-gcc.exe because it can link standard libraries with our file but we add option -nostartfiles to skip ~~start~~ section of startFiles at this libraries.

ex:

```
arm-none-eabi-gcc.exe -mcpu=cortex-m3 -I. -mthumb
-nostartfiles app.c startup.c -o output.elf
```

The error of malloc() will disappear but a new linker error will appear.

undefined reference to "_sbrk"

The function "_sbrk" This dependent on ←

linux : GNU C Library (glibc)

windows: Microsoft C Runtime library (msvcrt)

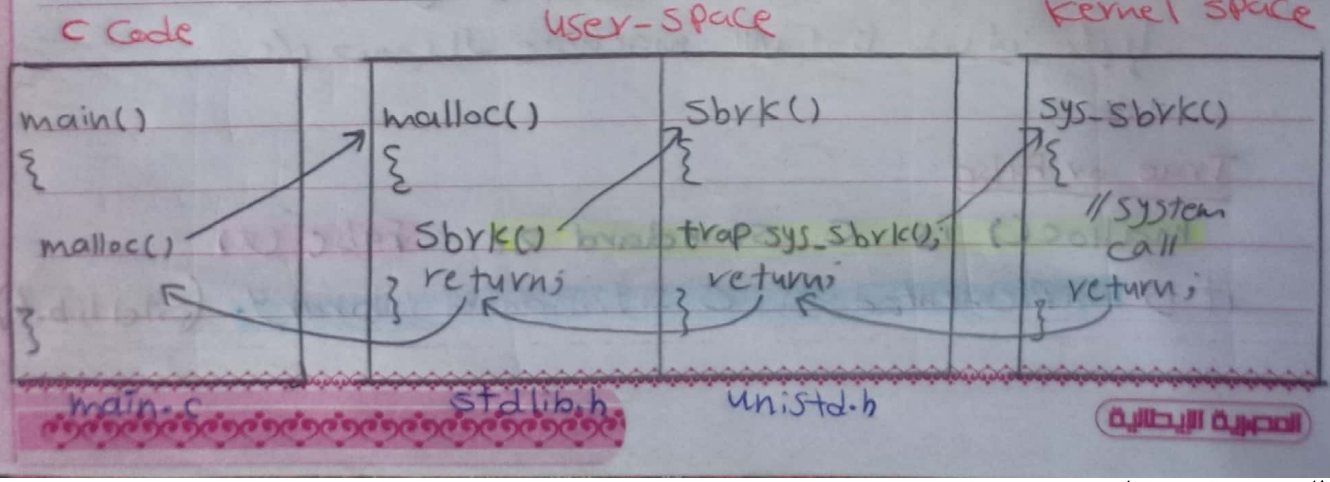
يعتمد على أن الـ func دي معتمدة على libraries موجودة في الـ OS بتاعتك زي linux في (glibc) و windows في (msvcrt)

C Standard library : These are no built-in facilites They use to preform such common operation as input/output, memory managment, string like as ⇒ Printf, Scanf, malloc

Under an OS many of the basic facilities require system calls to be implemented ⇒

printf in linux ⇒ use write system call to send out string

printf in windows ⇒ it calls write console API.



Why we can't use printf or scanf in Embedded ??

because if we use printF in main Function it will go to stdio.h and check its declaration then go to stdio.o who has its definition and this will call some function like _write() and those Function is call Comsol on ~~Wind~~ windows to print but there is no windows or Linux to display "in Embedded"

The implementations of specific standard library are platform dependent :-

- Linux → GNU C Library (glibc)
- Windows → Microsoft C runtime library (msvcrt)

الكلام ده نفس الوضع مع ال malloc (dynamic memory)

بحيث إننا لو استخدمنا malloc() عند ناف ال main فالكلام ده هيبقى على ال stdlib.o وده هيتنده جواها على sbrk- ودى جواها هيتنده على حاجة جوا ال windows أو ال linux على حاجات تسمى access للبرنامج بتاعنا على ال heap الخاص بال windows أو ال linux على حسب ال machine اللى إحنا شغالين عليها.

True or False

malloc() is C standard ?? False (x)

it is implemented in C Standard library. (stdlib.h)