

Gdb debugger commands

What is the gdb ??

it is a program comes with cross tool chain
debugger → is software runs on my host machine
to debug a software remotely (exist on target machine).

→ It allows the user to exercise control over the program and examine variables.

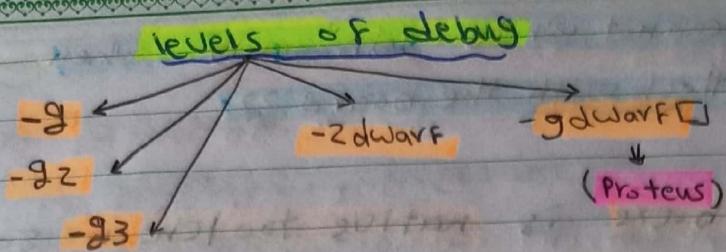
→ It allows to inspect what the program is doing at a certain point during runtime.

GNU Debugger is also called gdb

ARM JTAG interface debugger

- To include debug information in object file we use **-g** as a flag.

if we debug the elf image using gdb and the elf image has not contain debug information → then the debug is run but it writes all the program in assembly.



To run and open gdb circuit and the CPU wait commands from debugger we use

flag → "-S -S"

qemu-system-arm -M VersatilePB -m 128M -no graphic
 -s -S - kernel Learn-in-depth.elf

port number of debugger circuit from datasheet

is 1234

ip address of debugger circuit is the same ip address of my host machine and we can use local host to get ip address of my host machine.

To connect debugger circuit with debugger on host machine.

arm-none-eabi-gdb.exe learn-in-depth.elf
 ↘
 binary + info

check reading symbols after this command ✓✓

if not check → ① object files has debug info or not

② check level of debug (-g, -gZ, ---)

target remote localhost:1234

↑
 ip address ↘
 part number

Note why we cannot write "gdb" only when we attach with the board?

because gdb.exe is native toolchain so it is ~~a~~ debugger for a program run on the host machine not cross toolchain and we need cross toolchain to understand instructions and treat with ARM processor

في الواقع gdb.exe ليس من دليل المبرمجين بل هو دليل المبرمجين لـ bare metal application حيث لا يوجدnative program على نفس الـ arm processor ويعمل على arm toolchain لـ cross debugger يعني لا يوجد debugger على cross toolchain

لذلك gdb ليس ليس هو

الـ gdb هو دليل لـ debugger على board الموجود على الـ circuit board

GDB commands

l → list file يطبع ما في الملف

display /ni \$pc

This display n of instructions and point to the instruction that pc has it is address.

يكرر عرض من الكائنات بحسب الـ PC بعنوان .lines

b █ → This make BreakPoint at █
ex → b main
 لـ address is breakpoint لـ address
 is breakpoint لـ main لـ main
 سـ لـ main لـ main
 initialize stack لـ switch context
 my is breakpoint لـ my
 ss. (1) لـ my فـ main لـ

b * address_main	vv
------------------	----

si → Step instruction (assembly).

s → Step with c line

c → Continue till you meet a breakpoint then
 stop at it.

print my-var → it will print the value of my-var

watch var → it use to make a break point at var
 if var changed at anytime it will
 stop the debugger at this line.

where ⇒ get the location where we pause

info breakpoints → shows information about
 all declared breakpoints.

الموضوع :

التاريخ :

Set \$PC = 0x1

This will use to jump with PC to address of instruction and don't execute the instructions between the current instruction and the new instruction that will specify by \$PC

Mark File Tutorial

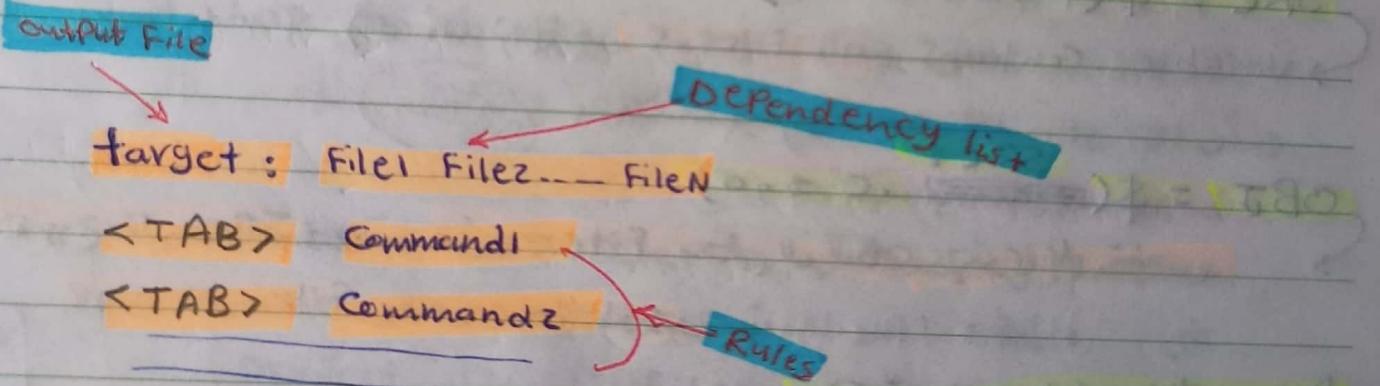
File کو کسی لئے اپنے Terminal میں دکھانے کا طریقہ

- automate the build by Makefile

لارج ناتيونال ميبل build project فيه ملف main كايزيزن نعمل
لارج فانتس لارج يرجع إن وقد أعمل build كل file لارج
ومن الآخر أوي link بينهم الموضوع ده يقدر وقتن
فالحل هو دى ننا نستخدم ال Makefile

لوي احنا كتنا عر د حلو من ال Files و احنا غيرنا من File واحد رس مثال build ف ميزه ال MakeFile ينه بيعرف ال File ده ويعرف ي build لو جده عمبيديش جمب الباقي ويركين ي link او Files كلها مع بعض دده هيوف الوقت كتنا بدل ما يتحمل .files كل او build MakeFile is

- Makefile is Incremental Build



ex -

app.c : app.c

arm-none-eabi-gcc.exe -c -I . -g -mcpu=arm926ej-s
app.c -o app.o

CC → Variable contains name of toolchain

CFLAGS → Variable contains the option passes to compiler

-g -mcpu=arm926ej-s

INCS → Variable contains options or include header files.

-I .

LIBS → Variable contains static library or other libraries we want to link them in our project.

\$< → dependencies

\$@ → target

% → Generic Rule

SRC = \$(wildcard *.c)

Variable contains all files.c (Names)

OBJ = \$(~~wildcard~~ .c = o) ←

عند إدخال OBJ على Files الذي SRC نجد File.o هو file الـ output

AS = \$(wildcard *.s)

variable contains all files.s (Names)

AS OBJ = \$(AS : .s = o)

عند إدخال Files في AS نجد File.o هو file الـ output

→ For Comments

What is the difference between

CMake and GMake

→ ordinary make

make or G Make

uses **makefiles** to describe

How To build your Code.

makefile contains targets and recipes.

د) اللي إنتي بتحاول إذلهم حاجات

МАЯ - в Море, море, море

Cmake is a higher level language with the same purpose but with a higher level of abstraction.

With Cmake you can generate ordinary Makefiles or files for other build systems.

ولكن اولاً هو الـ gmake فهو يعتمد على make ولكن اولاً هو الـ gmake فهو يعتمد على make !!

يساعد في إنشاء files Project ن_genrate

— او Code Blocks یا Visual Studio کی

Project structure files will generate هو بيعمل

(ordinary makefile) makefile ای یعنی اس دی files ای کہیں۔

اکسی بیکمل نیورڈجکٹ و بردا auto build

منہماں All Files دی پیغام اور startup, linker دی پیغام

use to automatic Full project not automatic

Build of Project only.

Lab 2 → Write Baremetal SW on ARM Cortex-M3 today
 32Bit micro controller STM32F103C8T6 chip.
 → (Toggle led).

Processor → cortex-m3

Board → STM32

Startup in Lab 2

- Define interrupt vectors
- Copy data from ROM to RAM
- Initialize bss in RAM
- Initialize stack pointer (SP)
- Create reset section and Call main().

entry point of flash is 0x0800 0000

Cortex-m processor in this processor we should put address of stack pointer (SP) at the first address of Flash memory (at the address that PC will point to when power is applied to the MCU).

عندما يدخل CPU للمرة الأولى

العنوان المحفوظ في PC

هو الـ JI instruction (Jump)

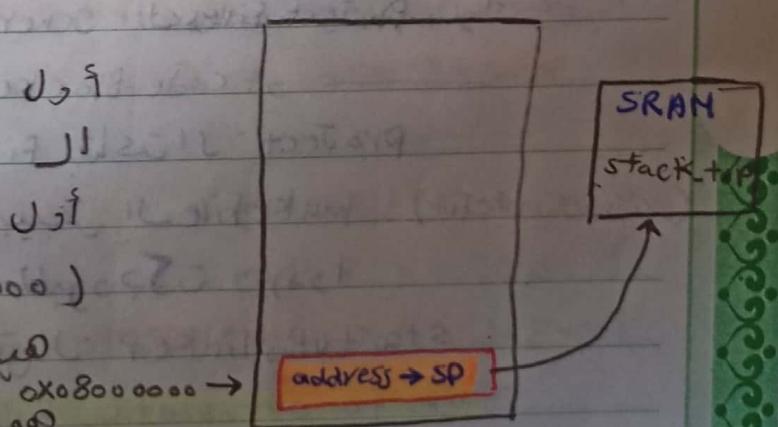
لـ 0x0800 0000 (0x0800 0000)

عنوان كبير

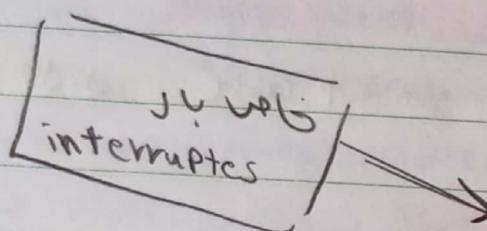
وهي address التي تذهب إليها

هي address التي تذهب إليها

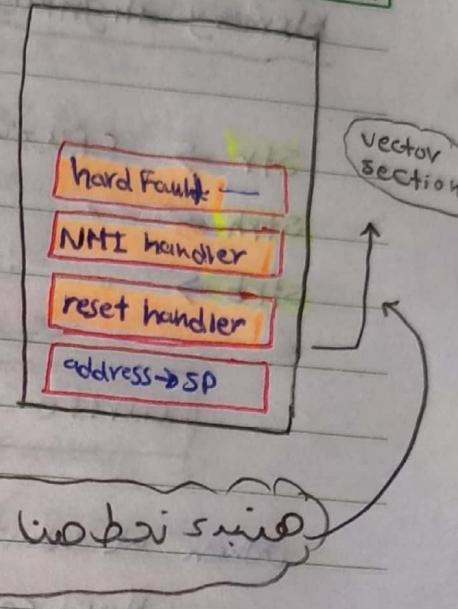
Stack pointer address هو



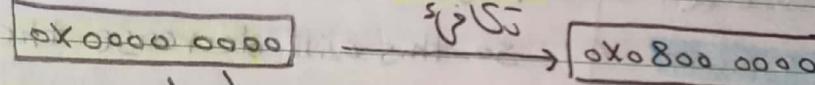
The program counter then steps to the next ~~instruction~~ address (0x0800 0004) and expects the address of the reset handler at this location.



0x0800 0004
0x0800 0000



in this processor in Vector Table



جاء 0x0000 0000 لـ 0x0800 0000
جاء 0x0000 0000 من Boot ROM

0x0000 0000 (0x0800 0000) in Vector Table
is Reserved why??

because.. we put address of (SP) in
this address.

reset section \Rightarrow 0x000 0004

NMI handler \Rightarrow 0x000 0008

hard Fault \Rightarrow 0x000 000C

وهكذا

الـ Vector table يكتسب عن طريق الـ startup

Memory alignment

Str → CPU write in memory with 4 Bytes (Word)

Strh → " " " " 2 Bytes (half-word)

Strb → " " " " 1 Byte

Write or read

when we want to write 4 Byte in memory

How can do this ??

We can use (Strb)

and we will write

4 instruction from it

to write 4 Byte

and this will increase

Code size and increase
the execution time of

Program so the program

Will be slow

We can use (Str) to write

4 Byte in one instruction

so this will decrease the

size of code so the program

will execute faster.

ex struct data

{

char data1;

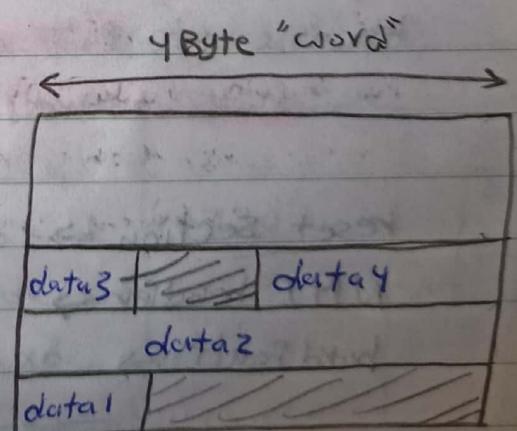
int data2;

char data3;

short data4;

}

Strb
3B Padding
Str
Strb
1B Padding
Strh



پاکستانی عمل ارکلاؤ ده کیان اماں نجیس نوزخ
ال sections memory میں ال تخلیقا

Memory alignment ہر ایسی

الكود يحقق أقصى سرعة و الأداء

الكلام ده بيكتب داخل ار linker script

طیب هست که همین لایه کانال sections مخصوصاً

no alignment address \Rightarrow alignment address is

Map File ھنریف عن طریق ادا

address is باین section کی گھر میں Map File اور
کام address is باین کے لئے

Thumb instructions

Cortex-m Processor has  16-bit instruction
32-bit instruction.

Command To open it

• thumb_func

(note)

خیلی اور startup و ہفتا پنکھن کی section

(-) يفهمون لأننا نحن أسلمة بيردز.

ویکون Alma Cline (underline)

Section 31 and Variables

ex 2 - reset:

حل اول stratus code رفع نیکتپ بار ۲

اگر یعنی اس میں پریسیس کے پڑھنے والے نہیں تو اس کو Arm cortex میں Processors کہا جاتا ہے۔

J1 is SP J1 address how works as processor J1

CPU میں اکرول ہے اور Processor جیسے entry point

لینکیں SP II address دیکھ رکھ لے define کیون میا۔

د بالتألي هيقدرا ر CPU لفهم الكود بار ٢ فاصنامه مكنز زکت

العنوان IP هو العنوان IP للـ Startup Network

نحو ال address المخطوطة في قسم ال SP

طبلو اول Processor مکنس کند سہ یا تبا بسط

new Job entry Point J1 is SP J1 address

اکتب ہار چ بردوا ہے

assembly startup دستگاه را آغاز کنید

عکسات اول و APG ۳ در لاما کیمپنکل هیچ عمل مستویه جاچان من

poz, RAM u ROM uis. dulu u copy u jelas

مان نا bss . قی RAM و بسیار بزرگ باشند.

وکدین یعنی ال SP و بکدین یعنی N Jump

لكل دار startup من قراره assembly المكتوب بالـ

main N Jump is the time limit in N days is

ساعتها نظر نفوم الكود المكتوب على الـ C فيها

1

Write Startup.c

- as (cortex-m3) can initialize SP with the first 4 Bytes so we can write startup by C code.
- CPY data section from Flash to SRAM
- initialize .bss section by zero in SRAM
- write vector handler functions

عسان انكرى شوية كود وتخيل لهم سكتن مدين

assembly خاص بالـ

• Section .name

→ Code of section

the array arr و صيغة

int arr[] -- attribute((section(".name")))) = {1,2,3};

-- attribute --((section(".name"))))

• الكلمة ده جاي من الـ data sheet اخame bar

standard C مع

Compiler N option هي انتا ندى --attribute--

بس كل واحد ينزل على #pragma C

cross-toolchain

Ex Using `vectors` as attribute (`((section(".vectors")))`)

ان Compiler option یعنی اسے دو همراه ہے جو Vectors (array of array) کا سامنہ میں ملے تو اسے اسے اسکرپٹ کرنے کا امکان فراہم کرتے ہیں۔

Functional attribute, weak and alias

Weak: the function that defined with weak address → can be replaced by another definition and linker don't make any error.

ہتھاں لو اے user جو اے main.c کرنے function وختن اس تو main.c کیا کرنا جو اے H-Fault-Handler start up.c میں موجود ہے اس کا purpose یہ ہے کہ جو اے linker میں ایکھاں اے function ہے اس کو اس کے symbol کے ساتھ linking error کرنا جو اے تانی.

الـ weak يعنط بعـاـلـ طبـ المـوضـوعـ دـهـ اـنـحـلـ إـزاـيـ (٢٩) مـوجـودـ منـ الـأـوـلـ

weak The weak attribute causes the declaration to be emitted as a weak symbol rather than a global

it is useful in defining library functions
that can be overridden in user code, so
it can also be used with non-function
declarations.

يبحث الخلاصة لو إلهايننا File Two وجوا كل لفافاً
linker معرفين اسمها main فنستار ال معرفين Func
ميفريش error سنان الآيتين نفس الاسم
فعونعمل واحدة من الآيتين weak بديهيان
الآتانية تحمل override على ال definition
بتاحها وسايتعو ال errors لفافاً

alias ("target") The alias attribute causes the declaration to be emitted as an alias for author symbol which must be specified.

--- attribute --- طریقہ is Compiler نے بندی option ← alias
alias فیہ اتنا نکرو ہے اور symbol یہ کوئی نہیں
وہ اسکی اچھی ممکن تجھے کل another symbol
الیas وہ تو ہے symbols کا
کہاں میٹاں weakAPI دو دوسرے symbol
نکریں یا override کرنے والے
روزی خدا من ال alias

`Void NMI-Handler(Void) __attribute__((weak, alias ("name")));`

خ باللغة C++ كل ال vectors الى واحد بين نفس ال addresses
لو كينا نشوف ان addresses المترافقين بينهم واحد
بما فيهم صنلا فربما كلهم واحد بينهم نفس address
وال alias واحد وال alias (weak alias) override
address المترافقين (weak alias) (override)
ال alias مختلف عن address

السبع في ذلك استخدمنا الـ alias

۱) انتان خلی کل ارHandlers پستاوورا علی مکان واحد
پس ماکل واحد پستاوورا علی مکان رنکوز بکتب
نه کل واحد.

حل اور Handler بیکری مکان ٹائپز

جوا ال مسحطةوا جوا . Handlers ال خزم ال Vector Section
هذا تبين انه حسب (Interrupt vector table) الموجود جوا
ال Microcontroller ال درکن كتابة ال اقدر اكتسبهم Functions جوا

Vector section based on interrupt vector table
that depend on micro controller.

How To Copy data section and Create .bss Section

II Virtual memory layout

يمكن نرسم كـ 1 ونشوّه خطوط إيه أول حاجة

في الـ Flash (إلا اللي بعد) وإنما اللي فيه خط

في الـ SRAM يعني اللي هنا فتحتمل إيه (١)

في ساق الـ Flash (.text) في الخط ١

إذ (.vectors) ، إل (.text) وهم خطوط إل (.vectors) الأول مساز

أول SP إل address ، وبدر مانحة إل (.text)

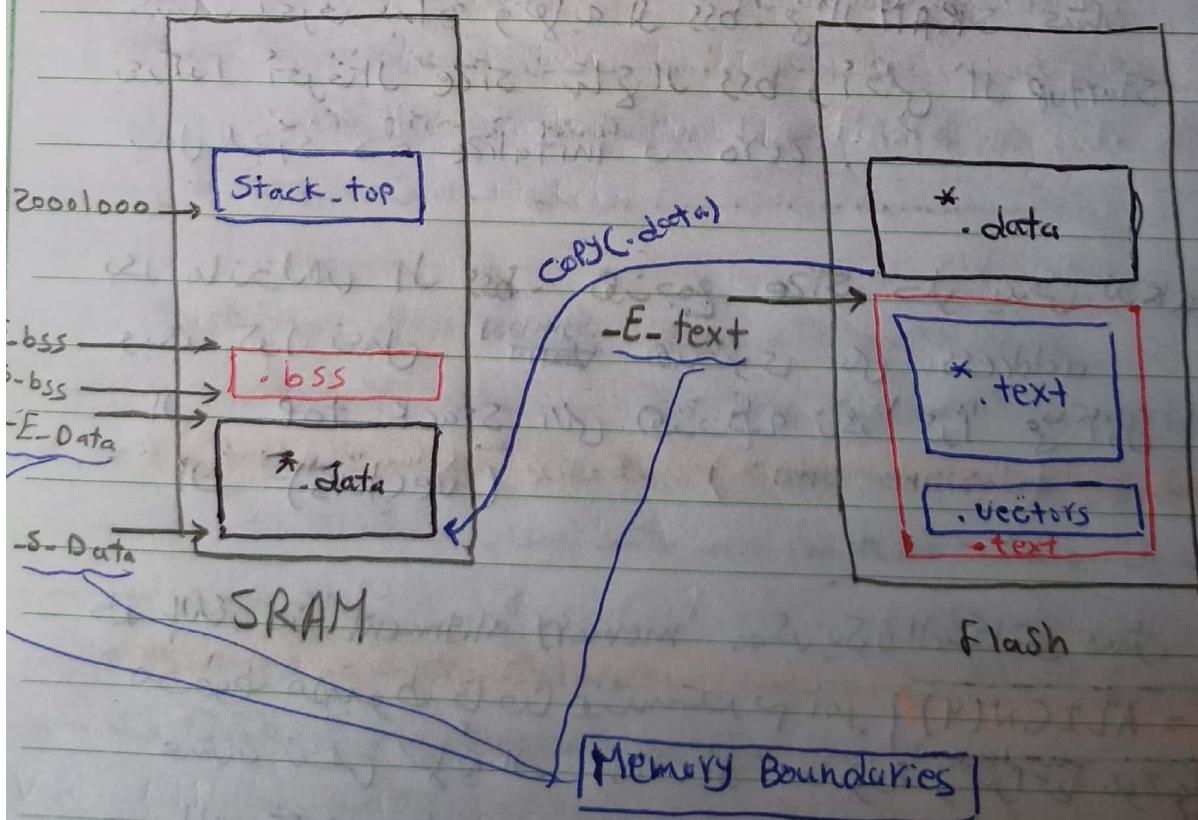
أول (.data) وبدر مانحة إل (.data) .copy (اللي موجود في SRAM)

باتي الـ bss . دلخوا مكانت في الـ SRAM عشان ينفع

خليل إل startup يكلم في بحفل وبدر

كـ ١ نسيي مساحة الـ stack وبعديه نسجل

الـ vectors (.data) أول إل stack-top إل



لما نكتب بستة فقرات امل هو ام (.) location counter عيّان نعرن ام memory boundaries وندر خبر ام اية . memory كل عيّان section نهاية كل

هذا يذكر ببداية ونهاية ام symbol ولمس variable و ام symbol لنفس تعريف ال variable بس الاختلاف ام ام symbol بدل على address و ام variable address و ام variable value و ام variable بحسب ال value .

لما نكتب symbols مثلاً في بداية ونهاية ال data . عيّان اعرف ام size ام ام size ام size startup > SRAM او Flash من ال copy على بداية ام data . ونهايته)

هذا ينطبق على symbols في بداية ونهاية ال bss . وهذا عيّان اعرف ببداية ونهاية ال bss . في ام SRAM او عيّان bss اخر فار size متى ال bss و اخلي ام startup . zero ب initialize ب يكمل لجزء ده

بعد ما نخلص ام bss . هنرجع size حلو يبص للstack address عيّان ~~symbol~~ يحتوى عيّان stack-top كدا غير اول (.vectors) ال

خد بالع ممكن يكون السكانين مثلاً memory alignment مخطوط ممبوط خاصنا هنستخدم أمر [] = ALIGN(4) عيّان تخلص نهاية ال section ممنوعة و مراعية هو ضرورة ال memory alignment

COPY .data From Flash to RAM

(١) .data symbols الخاتمة ببداية ونهاية ال

File میں اسے module extern اسی طرح

linker script ॥

extern unsigned int -S-data;

" " " -E-data;

2) calculate the size of Data Section.

`unsigned int DATA_size = (unsigned char*)&_E_data - (unsigned char*)_S_data;`

Variables (مُعَوِّضات) are symbols used in formulas (أمثلة) (مُعادلات).

يعنی بيدلوا على address وتحتها Value

مکنیکیں یعنی نتائج کیجوں کے ایک ایسا دادا۔

(byte)

~~Dear Jesus living char*~~ casting ~~you~~ Jesus ←

میں اسی لیست کا جزو ہے جو اسی طبقہ کے مطابق (byte) ہے

ويمكن ان memory م تكون قائله بـ (٤) وبالنهاي لومكنتس

قابلہ ناتھا سماں یعنی ممکن نہ خل یعنی حاجان Section تانی

③ define pointer to start of source data

and pointer to start of destination data

• `unsigned char *p_src = (unsigned char *) &-E-text;`

.text section دل مکانی data بدل کنید.

• `unsigned char *p_dst = (unsigned char *) &-5_data;`

لأن الـ 8-5-data لهن الـ RAM دا هيكون واحد أول address في الـ RAM

ما اسما فناييكتين الرخوار ده في ال linker script

4) do loop with data size to copy - data
From Flash to RAM.

For (int i=0 ; i < DATA_SIZE ; i++)
{ }

$$\ast((\text{unsigned char } *) P_dst++) = \ast((\text{unsigned char } *) P_src++);$$

3

Value of P_dst but it casting to (char*)
to write byte by byte then we
increment P_dst to next byte

Value of P_src but it casting to
(char*) to read byte by byte
then we increment P_src to
the next byte.

value loop will be 1.5 MSU Jiaxin

locate .bss in SRAM and initialize it with zero

calculate .bss size

unsigned int BSS_Size =

$$(\text{unsigned char } *) E_bss - (\text{unsigned char } *) S_bss;$$

Pointer to start of destination bss section

unsigned char *P_dst = (unsigned char *) S_bss;

loop with bss size to initialize this array of memory with zero.

```
for (int i=0 ; i < BSS-size ; i++) {
    *(unsigned char *) P-dist++ = (unsigned char) 0;
}
```

after that we jump to main() function
in startup.c

```
main();
```

sections ال بحث هو linker script نوع آن

size على ال location counter (.) (.)

حل يبقى لل stack وبعد كل سجل قيمة

ال (.) خواص symbol stack-top

دكتور ال (.) موجة خط خريطة

ال vectors بدل ما في هنا حافظ رقم عشوائي

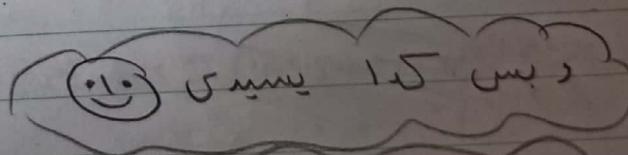
من دعائنا.

important note

لما زجت نسبات ال sections في linker script ونجز تشكيل map file من خواص المقالة ساسا

لئو کیا lines uninitialized global var قالیمفر (چنان اینہ هیئت خزن خی (bss) بس باختنا لو جینا سوچنا بدایا و نفع) لئو ال bss هنلا جمی پانہر زکی بعض طب فین ال Var ایں باختنا عر قناء هائیوسٹ لیہ خی ال bss لیں

في الحالة دي الـ `va7` اعملاً `global uninitialized` اتـخـزـنـخـي الـ `(COMMON)` و بالـتاـليـاـ هـذـاـ لو حـطـبـنـاـ سـكـسـنـاـ الـ `COMMON` مع الـ `bss` دـلـيـلـاـ بتـكـبـرـ الـ `linker` سـاـكـفـعـ هـيـمـهـافـ او عـزـزـدـ بـتـاعـ الـ `bss` `uninitialized global vars` هـتـكـتـلـفـ عن نـفـاـيـهـ طـبـ دـاـخـلـاـ فـرـضـنـاـ هـذـاـ مـئـ كـاـيـزـرـينـ `linker` الـ `Common Section` مع الـ `bss` دـلـيـلـاـ هـنـجـمـلـ اـيوـجـكـوـ هـنـجـمـلـ اـيوـجـكـوـ



د) اسکب خیانتا نلایم) بدایه و غاییه ال رسما زی بھمن