"Booting Sequence"

Power on / Reset → Entry Point → Flash

Entry Point → "data sheet"

eg Flash 0x 0000 0000 to 0x FFFF FFFF

Specs → EP → 0x 1000 0000

* Inst- Life cycle → (P.d.e)

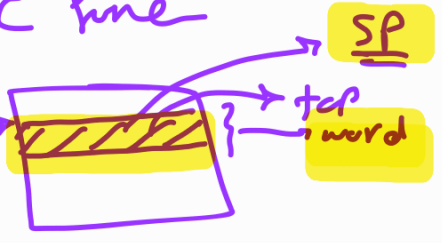
(.bin) → binary file → burn → MC → EP → Flash

① Startup Code (.s / .c)

Note

Init stack ---> C fune

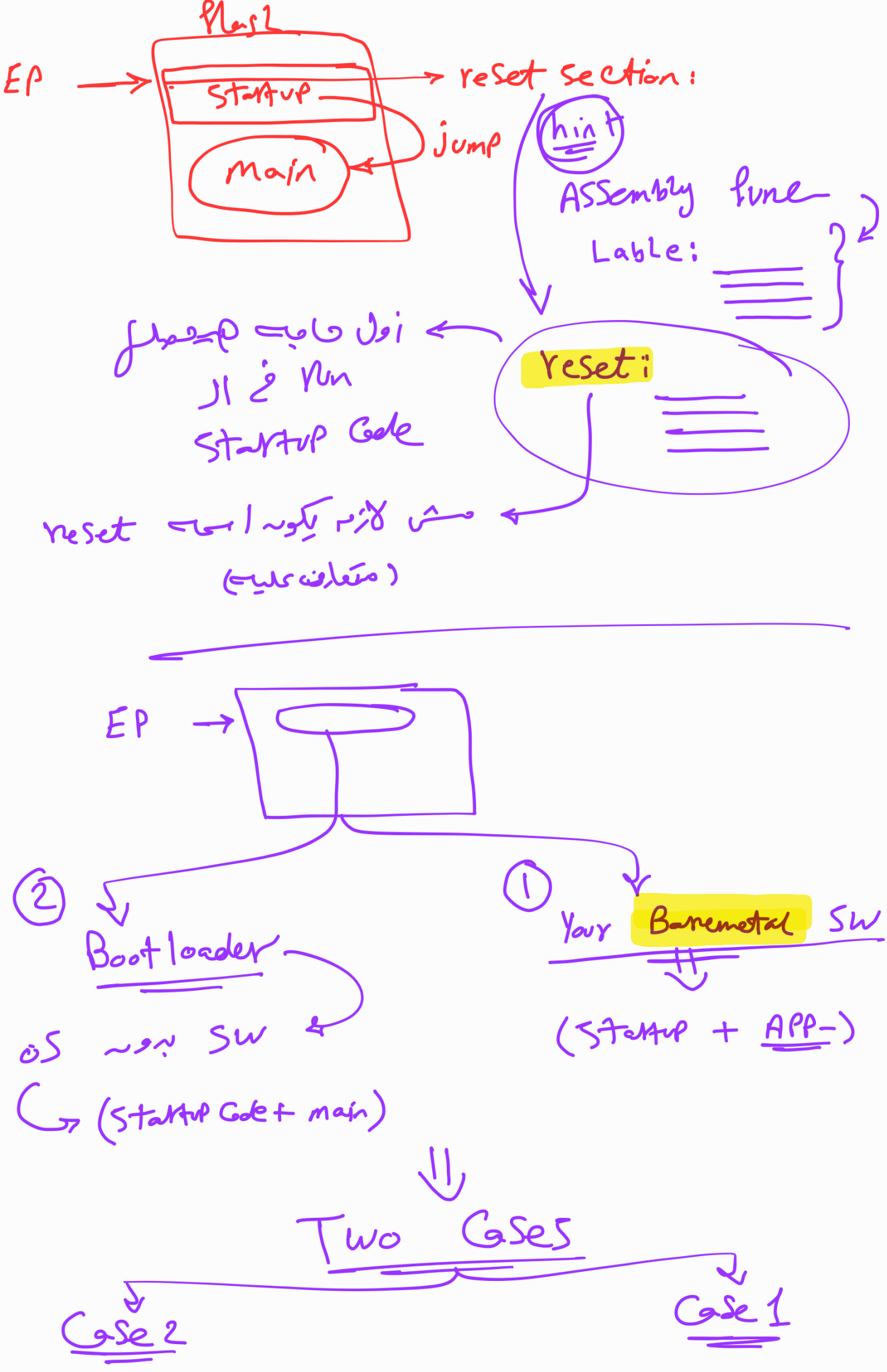
eg ARM Cortex M ⇒



② Code → (.text)

③ global & static → .data
(Init- ≠0)

④ ~ ~ → .bss
(uninit-)


Flash
EP
Startup
main
jump
reset section:
hint
ASSembly fune
Lable:
Reset:
أول حاجة هتتعمل
Run في ال
Startup Code
مش لازم يكون اسمها
reset
(متعارف عليها)
EP
2
Bootloader
OS بدون SW
(Startup Code + main)
1
Your Baremetal SW
(Startup + APP-)
Two Cases
Case 2
Case 1

Case 1 ⟹ (Linux)

developer ⟹ Entry Point ⟹ (Baremetal SW)

⇓

(Startup Code) + App (main)

PC ⟶ Entry Point → Startup Code

(Program Counter)

↳ segment

reset:

⇓

① Init stack (SP)

② Copy (.data) — Flash → RAM

③ Reserve (.bss) — RAM

⋮

④ branch / jump ⟹ main

& so on

Case 2

developer ⟹ Entry Point → (Boot loader)

ex: BootROM

Startup Code + main

reset:

⇓

Boot loader

Boot

load

& so on

load (Source → dest-)
@(Run time)

Boot loader ⇐

Init- ① ←

Periphe- ← (modules)

↳ eg SPI

ek ← load ②

Flash ⇒ OS/SW → Embedded Linux

load → RAM

(عاوزة لوجود)

Jump ③

↳ Startup → SW

branch → main

hint → Reusable memory space

Prm ⇒ two Cases

↳ (two Running modes)

Running modes

Rom mode (case 1)

- Rom
  - → Entry Point
  - Exec Code + Const data
- RAM
  - Stack + char data

RAM mode (case 2)

- RAM
  - else – Code & Stack + char data
- Rom
  - → Ent Po. Boot Loader
  - ⇒ Code image OS/SW

another loc (or) Rom

- SD card
- ether
- flash

Compar

Rom mode (✓)	RAM mode
• very simple • req smaller memory • fixed Code address • Relative small Code	• Complex • Relocatable mode • fast why? (RAM > Rom) ??

hint

→ Embedded Linux

→ RAM mode (Boot loader)

Compare

→ (Boot loader Vs Startup Code)

(Phases & Stages)

→ (Embedded Linux diploma)