* Motivation to Learn ARM-Cortex:

■ used in most Microcontroller today:

1] Battery power Devices ex: health monitaring Fitness Tracking.
2] Automotive Apps.
3] IoT Apps.
4] Mobile & Home Apps.
5] home & Building Automation.
6] Toys & Consumer product.
7] Ac & mobile Accessories.
8] Test & measurement Devices.

■ most of the famous MCU manufactors producing MCs based on ARM:

1] Ti ⇒ Texas instruments
"Low Power Battery Based Apps."
2] ST ⇒ High + medium + Low Performance MCu
3] Toshiba ⇒ measuring Equipments & meters.
4] NXP
5] Microchip
6] Board Communication ⇒ wireless Connectivity
⋮

■ manufactors Love ARM CortexM For:

1] it's minimal Cost.
2] minimal power.
3] minimal Silicon.
4] 32-bit processor will boost Computational Performance with the Same price of 8 & 16 bit processors
5] Ultra Low power to High performance based Apps.
6] processor is Customizable include FPU, DSP, MPU, etc.
7] very powerful & Easy to use interrupt Controller which support 255 Exceptions → 15 system Exception → 240 Interrupts
8] RTOS friendly in which it provides Some System Exceptions, Processor operational Modes, Access Level Configuration, Stack Model helps to develop secured RTOS related Apps.
9] it's Instruction set is Rich & memory Efficient, it uses Thumb instruction set (16 bit), ARM instruction set (32 bit).

Now we use Thumb 2 Instruction set [16 bit & 32-bit].
"most uses"

*ARM Cortex-M Cannot Excute ARM instruction Set, it uses Thumb2 instru. which give the Same Efficiency & Power of The 32 ARM Set but in 16 bit format.
10] ARM provides alot of documentation
11] & Reference manuals, user Guide.

Scanned with CamScanner

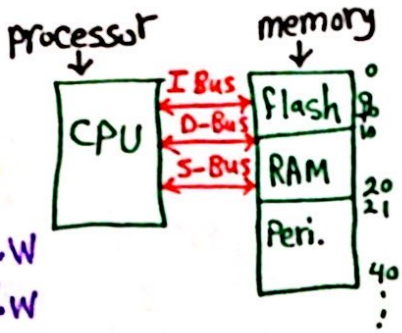
MicroController Architectures:

There are 2 Architectures:

[1] Von neuman & [2] Harvard

[1] Vanneuman

Vanneuman:



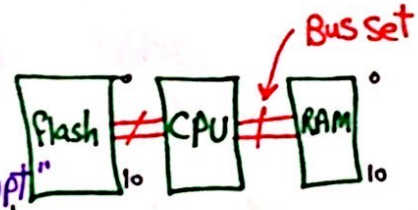
Advantages:

[1] simple in S.W
[2] // // H.W

Disadvantages:

→ no piplining [slow] & Low throughput.

[2] Harvard:



Advantages:

[1] Piplining "High throughput"
[2] Address not Incremental

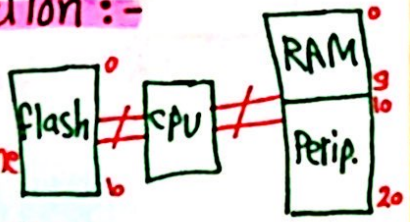
Disadvantages:

[2] Complexity in H.W & S.W

* In Harvard we have 2 Type of Connections:

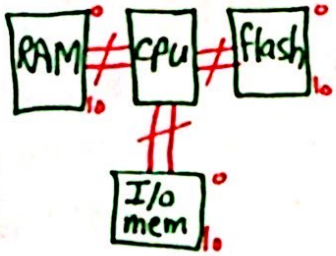
[1] memory map Connection :-

[1] Simplicity in S.W
[2] Lower Stage in pipline

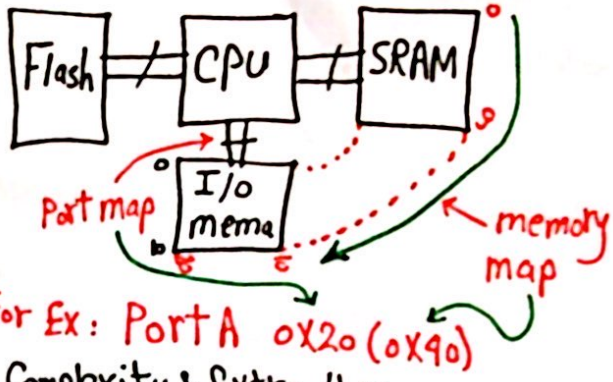


[2] Port map Connection :-

[1] Complexity in S.W & H.W
[2] Extra pipline Stage



* In AVR Architecture:



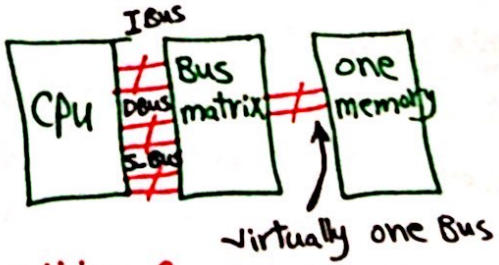
For Ex: PortA 0x20 (0x90)

* Complexity & Extra H.W

← في الـ AVR هو عبارة عن Harvard Arch و دا بيبين ميزة الـ Piplining و لكن الـ AVR وفرتلي ميزة كويسة وهي أنها أدتني 2 أختيار ما بين الـ memory map والـ port map و دا لو أنت عايز تكتب C فانت كدا لازم تستخدم memory map والطريقة دي أسرع في الـ Dev. ولكن هتفقدن جزء من الـ Piplining و هو هيفضل موجود ولكن ليس بشكل كامل لو عاين تستخدمه بشكل كامل فلازم تستخدم الـ port map وهنا هتضطر تكتب Assem. علشان تعرف توصل I/O Peripherals.

* Data Bus in AVR to Flash is 16 bit
* // // // // to RAM is 8 bit

*In ARM ARchitecture:



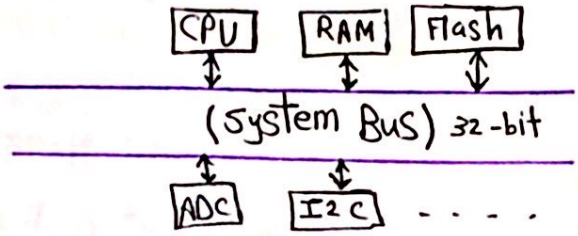
*writing C, you Can access any memory.

* Using Pipline of Harvard.

Scanned with CamScanner

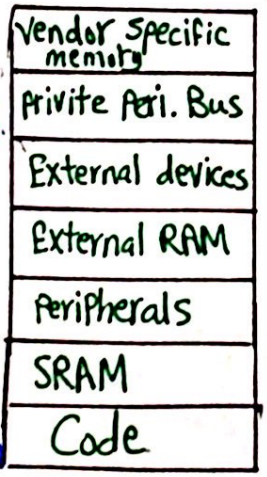
*Lec 2 ⇒ memory map

Explanation of mapping of different peripheral Registers & memories in the processor addressable region, depend on size of address bus, This is Called memory map.



This address bus is 32 bits, so it can access up to 4G Locations starting from 0 ⟶ 0XFFFFFFFF

* memory map:



→ This Regions are fixed by The processor designer, Can't be Changed.

* program Code Could be Excuted from SRAM Region.

[1] Code Region:

0.5GB ⇒ 0 ⟶ 0X1FFF FFFF

- Code memory is Connected here 512MB is very Big for most vendors, Connect as MAX 1MB
- usually for bigger Code memory than MCU internal flash, we go for External Flash.
- processor by default fetches instructions & vector Table information from This region.
- directly after Rest, you Can change this by the boot Pins of the Mc.

[2] SRAM Region:

- Next 512MB, 0X2000 0000 ⟶ 0X3FFFFFFF
- mainly for Connection of internal SRAM
- 1st 1MB of the SRAM Region is bit band Region "bitaddressable Region".

```
*(U8*)(1000)|=1<<0;
```

↳ Read, modify, write

1clk 1clk 1clk

∴ 3 Clock Cycle to write1


```
*(U8*)(2000)=1;
```

↳ 1 clock cycle.

- bit banding Can be used for bit banging "Stream of bits"

[3] Peripheral Region:

- Next 512MB → 0X40000000 → 0X5FFFFFFF
- used mostly for on-chip peripherals.
- Like SRAM, 1st 1MB of this Region is bit "banding" addressable Region.
  ** if the bit band feature included **
- Processor peripherals are not here [NVic..]
- This is an Excute never Region. ##
- Trying to Excute Code from this region will Triggering fault Exception.
- Basically this is to prevent Code injection attacks, Like SB Can transmitte Code through the peripheral & make processor Excute the Code.

[3]

Scanned with CamScanner

[4] External RAM:

* Suppose you need more RAM for your project, i.e Some Graphics related Project So you Can Connect External RAM to this Region. #Can Excute Code.

[5] External Device Region:

* 1 GB.
- Intended for External Devices and or Shared memory.

Excute Never Region

[6] Privite Peripheral bus Region:

- This Region includ [NVic, SysTick, SCB, ..]
- Excute Never region. (XN)

■ Bus protocols & Bus Interface:

ARM provides [AMBA]: advanced Mc Bus Architecture, standard for on-chip Comm.

• AMBA Supports Several bus protocols:

1- AHB Lite [AMBA High Performance Bus].
2- APB [AMBA Peripheral Bus].

* on chip Communication means Communication between Processor & memory & peripherals.

⇒ AHB & APB: "Bus Protocols"

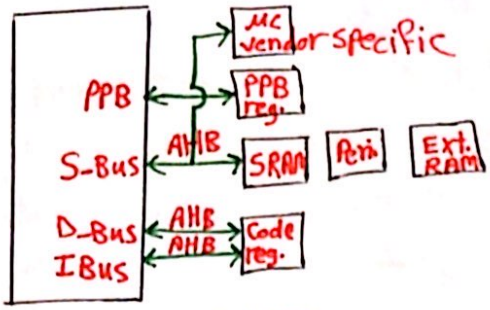
1- AHB Lite is mainly used for the main Bus interfaces

2- APB is used for PPB Access & Some on-chip peripherals, using an AHB -APB Bridge

3- AHB Lite bus majorly used for High Speed Communication with peripherals that need high operation speed

4- APB is used for Low Speed Comm. Compared to AHB, most peripherals that don't require Hi-perf. are Connected to [redacted] APB.

■ Processor gives out 4 bus interfaces:



3 AHB buses & 1 APB.

2 buses interfaces for the Code region

1] I-Bus: used for instruction fetch & Vector table Read

2] D-Bus: used for data Access in The Code Region [Like Const]

So fetching Data & Instruction Can be done at The Same Time

3] S-Bus: vari. bus on chip peripherals & memories.

4] PPB-Bus: used for Core peripherals

Scanned with CamScanner

Processor Vs processor Core

#Inside the Core:

1-ALU
2- Decode & Excute Circuit [CU]
3 - Register file
4- Pipline Engine

M3& M4 only have One Core.

#feature of Cortex-Mx processors:

1- Operational Modes
2- Different Access Level
3- Register Set of the processor [Core Register].
4- banked Stack Design [stack Memory handling]
5- Exceptions& Exceptions Handling.
6- Interrupt handling
7- Bus Interfaces & Bus Matrix
8- Memory Architecture [Bit Banding, memory map, ...]
9- Endianness
10- Aligned & UnAligned Data transfer.
11- Boot Loader & IAP Programming

[1] Operational Mode of the Processor [M0, M3, M4].

There are 2 Modes:

[1] Thread Mode [AKA User Mode].
[2] Handler Mode.

[1] All your Code will run [Excute] under Thread Mode of the Processor

[2] All the Exception handlers [ISRs] will run under "handler Mode", either it was system Exception or Peripher Interrupt.

[3] processor always starts with Thread Mode.

[4] Whenever ISR starts, Core Chang it's mode to handler mode.

[5] In handler mode, you have full Control over all the processor, system Level, Register, Interrupt Config., Control Registers.

[6] In Thread mode, you may have Control or you may not have Control depending on access level.

Scanned with CamScanner

[2] Access Levels :-

[1] Processor offer 2 access levels:
- [a] Privilaged access Level (PAL)
- [b] non privilaged access Level [NPAL]

[2] if your Code is running with PAL then your Code have full access to all the processor resources, restricted registers, System Level registers which are protected by the processor it self!!

[3] if your Code is running with NPAL, then your Code may not have access to Some of the restricted registers of the processor.

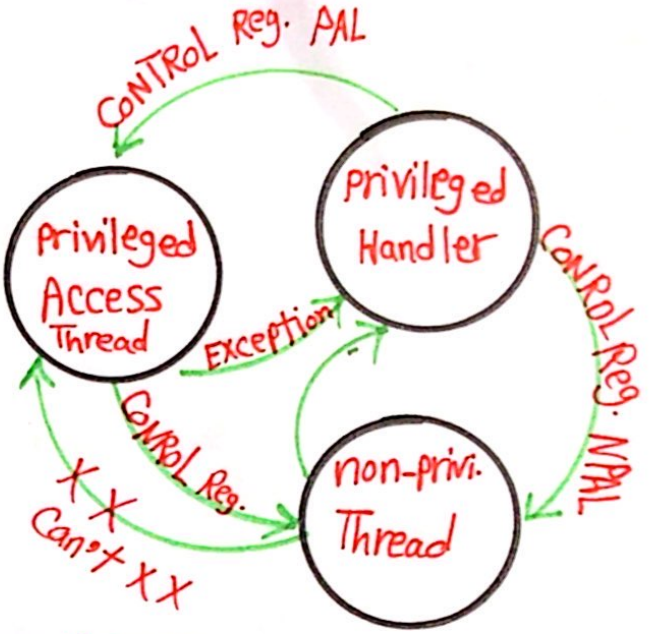
[4] By default, your Code will run in PAL

[5] when processor in thread mode it's possible to move processor to NPAL then it's not possible to get it back to PAL, unless processor Changes to handler Mode.

[6] Handler mode Code Excution is always with PAL.

[7] use the CONTROL register of the processor if you want to switch between the access levels.

[8] In Handler Mode you Can Config. whether it will return on PAL or NPAL.



Benefits:

[1] When a user program goes wrong, it will not be able to Corrupt Control Registers, most Common use in RTOS [Kernel & Tasks]
↓ PAL ↓ NPAL

[2] MPU Can block Some regions from program user.

Scanned with CamScanner

[3] Core Registers :

[1] R0 → R12 (GPRs)

[2] all Core Registers are 32bits Size

[3] R13 is Called SP (stack pointer), use to track stack memory.

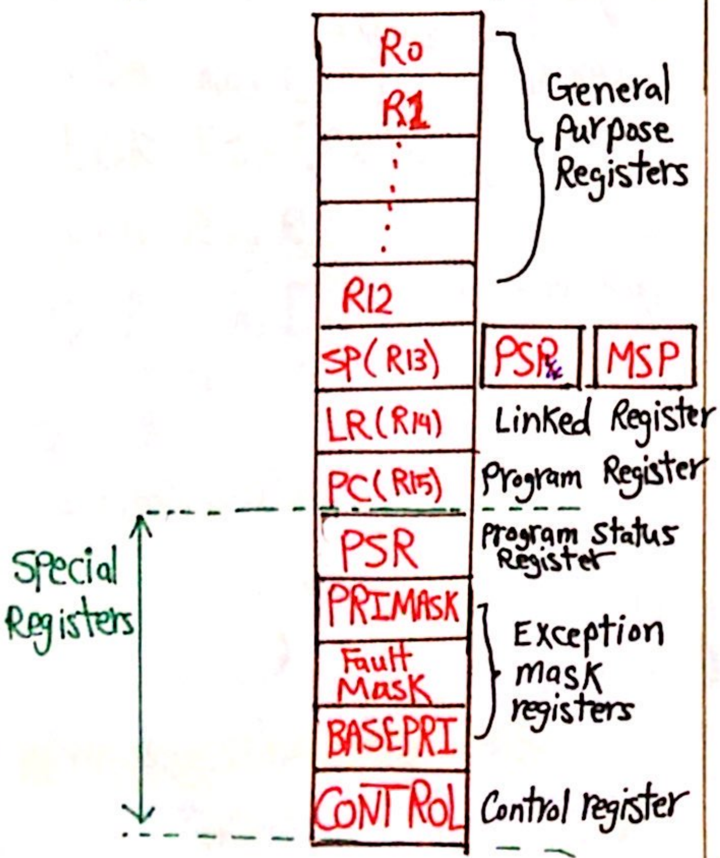
*PSP ⇒ process / program stack pointer

*MSP ⇒ main stack pointer.

#Banked versions of SP.

[4] R14 is Called LR (Linked Register), used to save the return info from ISR, Subroutine, function call.

[5] program Counter (PC), Contains Current program address, Bit[0] of this register is Loaded into T-bit & must be 1.



bL : branch with Link
update PC & LR

bX : branch Indirect
PC = LR

■ Special Registers :

*5 Registers [PSR, PRIMASK, FAULTMASK, BASEPRI, CONTROL]

[1] PSR : Program Status Register, hold status of Current Excution of the program this Register is Collection of 3 different registers.

[1] APSR : APP. Prog. Status Register (flags)

[2] IPSR : Interrupt prog. Status Register, Containe ISR number.

[3] EPSR : Execution program Status Reg.

■ branching Instruction :

bX [Address] 0b 0 1 0 0 0 1 0 .... [1] (bit0)

0b 0 1 0 0 0 1 0 .... [0]

[1] PC = address

[2] get bit0 off address, give it T-bit [EPSR].
0 : ARM SET
1 : Thumb SET

* if (T) bit of the EPSR is set to (1), processor thinks that the Instruction which is about to Excute is from (Thumb) Instruction Set.

* if T-bit is (0), processor thinks that The Next Instruction which about to Excute is from (ARM) Instruction Set.

* ARM Cortex-Mx Processors only Support Thumb set, so T-bit most be always 1 if it be Comes (0) it results in processor fault. #

pc bit(0) & T bit are physically Connected so most Instruction that update PC will update T-bit, bX, bLX