What is ES?:-

Computing System ال لازم يتكون من ٣ → Processor / memory / I/O

حاجات اساسية

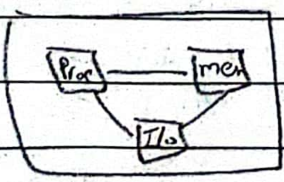
Computing system → input → Processor → output

تنقسم إلى نوعين:

- General Purpose → Computer
  - كله بيشتغل في كذا حاجة
- Specific Purpose → single Purpose
  - الاداه غرض واحد بس
  - → E.S → Constrains ثوابت
    - power
    - cost
    - speed / time
    - size
    - → ES-Challenges

How to make E.S?

- System on Board → اشتريت كل Comp لوحده
- System on chip → IC



	S.B	S.O.C ✓
Size	↑ اكبر	اقل ↓ ✓
Cost	↑ اغلى	ارخص ↓ ✓
Power Consumption	↑ بيسحب اكتر (محتاج اسلاك بتستهلك power)	بيسحب اقل ↓ ✓
Performance / exp. time	Same	≃ Same
Configurability	✓	X (Built in)

* Mezet el S.B :-
(اقدر اعدله) قدرة ال Configurability ال System modified
يعني ممكن ازود ال memory
مناسب في حالة التطوير

* What is IC? → Integrated circuit → single chip

→ VLSI → very large scale Integrated circuit → mill of transistor
عدد الخلايا بيكبر NAND
فايدته
→ size (↓)
→ functionality ↑ عالية
تقدر تعمل حاجة كبيرة

PAGE
DATE

(*) moore's law ... عدد الترانزستور بيتضاعف
doublication مع مساحة الترانزستور كل سنتين

(*) MPU → micro Processor unit

Processor
MP micro Processor
CPU
Graphics Processor unit
GPU → secondary (slave)
DSP

انابيب
مفردة
Transistor (New Processor)
Processor Master
Master → CPU
Primary Processor
ثنائية
ثلاثية
bit

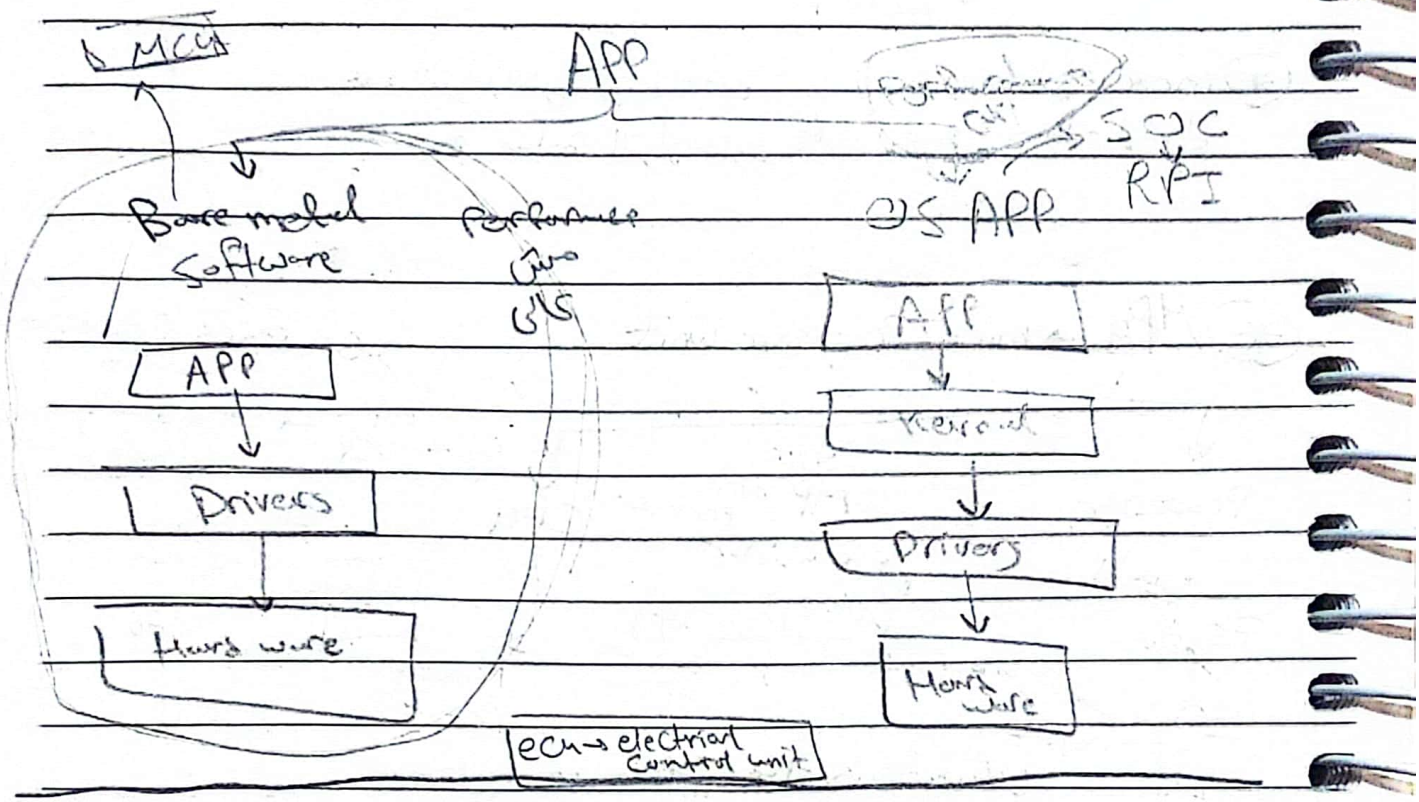
What is the difference between

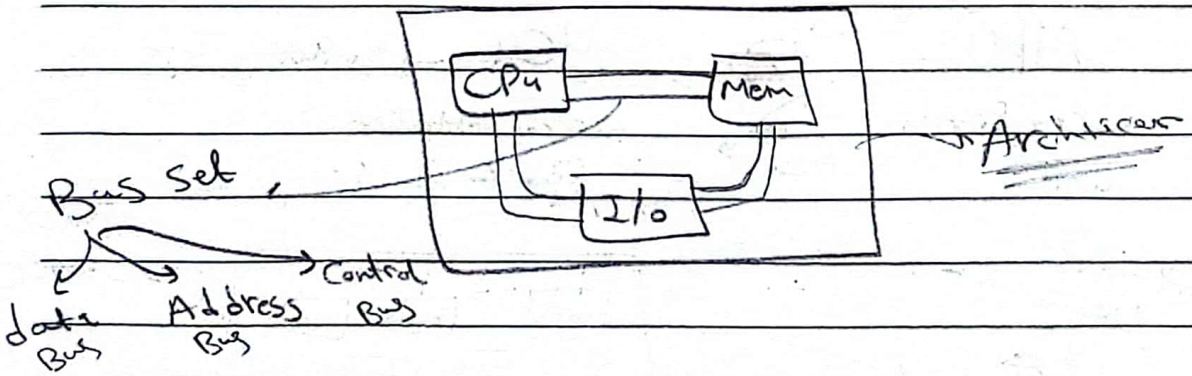
MPU جزء من MCU
micro Processor unit
MCU → Computing sys
micro controller unit

Arithmetic Logic unit
جول
ALU
CU
Reg - Files
Control unit
يقوم بالعمليات الحسابية والمنطقية

MCU (high Perf)
S.O.C
mPu
mem
I/O
GPU
DSP
draft
graiph
CPU
Secondry Processor
Primary Processor

PAGE
DATE





Processor

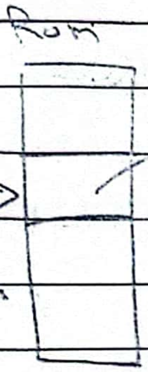
مكون من ٣ حجات اساسية

1- ALU
2- CU
3- Reg Files (Bank)

محتاجة انو يكون ٣ حجات

1- انيشيال فيتش
2- decode
3- excute

Instruction بنسميهم ال life cycle



Program Counter ⟹ اللي بيشاور على الـ code او الـ instruction اللي عليها التنفيذ

بتاخد الكود الموجود في الـ Rom ← بيبعت

وتعمله Fetch لـ الـ Code بعدها

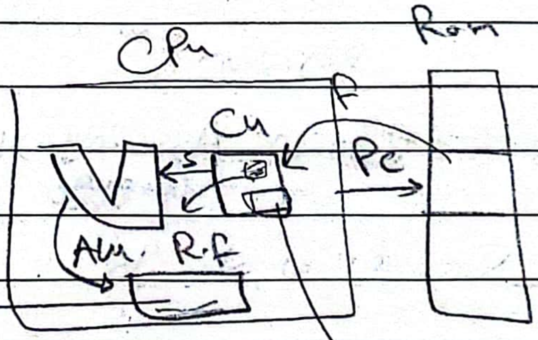
وتحطها في الـ Register → IR Instruction Register

Program Counter → File ده في الـ Register Files

بيبعت عنوان المكان Address اللي جاي اللي عليه الدور في التنفيذ

Next instruction

مش اللي بتتنفذ دلوقتي



IR ← ID - Decoder

بيروح بعدها يفهم decode بدايرة "decoder"

الـ Instruction → بيدين بيبعتها للـ Alu

علشان افهم ال Ins.

علشان اعمل Decode نعمل حاجتين

① Ins. Set

② Ins. format → شكل ال Inst

8 bit ال حاجة

000	111	01
op-code (operation) بيفسر ال Instruction	op_1	op_2

op-code	operation
000	Add

* هنرجع يتكرر تاني في R-files (Register)

Note:- ال Assembly language مختلفة من M.P للتاني
كل واحد له ال format بتاعته → (علشان كده بنستخدم ال C language)

* علشان نفهم ال Inst

- H.W → ID - Decoder
- S.W → Compiler

PAGE
DATE

Prof. كمل

ISA:- Inst. Set Architecture

RISC (SW↑)

Reduce inst. set Computing (↓) يقوم اوامر قليلة

CISC (SW↓)

Complex inst. set Computing (↑) يقوم اوامر كثير

متى نحتاج Compiler قوي

decode → I.D تفسير الـ opcode

Hard wired

Compiler قوي

(logic gates) اوامر كثير / gates ↑

I/P → L.C → O/P (سريع)

Memory mapped → (Search)

Search → set

Instructions كثير

operand

CISC

RISC

PAGE
DATE

ES - Challenges, RISC

	RISC	CISC
Size	ID ↑ اكبر Alu ↓ متقدم	ID ↓ صغير , Alu ↑ كبيرة
Cost	SW ↑ H.W ↓	SW ↓ H.W ↑
Performance من اسرع	SW H.W (*→+)	SW H.W (Search)
Power Consumption	Alu ↓ ID ↑	Alu ↑ ID ↓

(The same)
Instructions كل حسب *

(2) Register Files MP → Alu Cu (R.F)

General P.R → Register
↓
data Store
(temp.)

Specific P/R
↓
(Specific Purpose)

Special function Register:

① PC → next inst.
Program Counter

② SP → Stack Pointer → اخر حاجة في الStack
طريقة تخزين الdata في الmemory
(Stack عكس memoryالـ التخزين)

③ ACC (Accumulator) → بيتخزن الResult مؤقتا (لا يوجد دايما) (زمان)

④ IR → Inst-Reg (بيتخزن في الIns الي جايلك fitch من الmemory)

⑤ PSW → Process Status word
→ flags (Pointerزي ال)
اخر عملية حصلت على الAlu

* وممكن لا يظهر الflag في حالة الJump→Inc
X+1

PAGE
DATE

* Memory :-

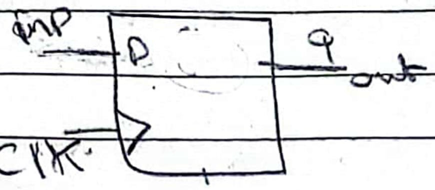
8 bit = 1 byte

0 ... 1

* Acess time ↓ → ↑ سرعة الـ memory

(↓ Read - write ↑)

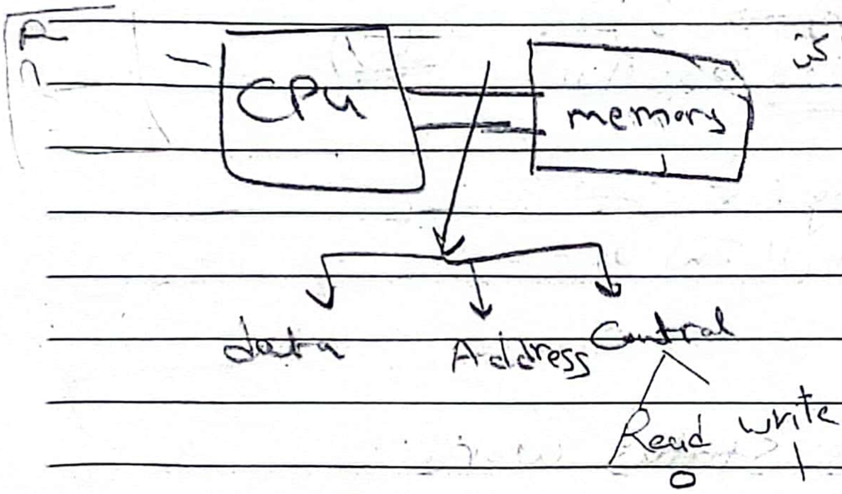
→ Basic memory element → (flip - flop) (D)



bits: n → 2^n Address lines

n أماكن

2^n عدد الأماكن يوصلها



ما يميز الـ memory

1. Capacity (size) كام Bit
2. Speed (Acess time)
3. organization

→ Address line لو ... يبقى

PAGE
DATE

Memory انواع

- Volatile
  - → Ram
  - متطيرة
  - لا تحتفظ بال data لو شلت ال Power منها
- non-Volatile
  - → Rom
- Hybrid
  - mix

RAM اسرع من ال ROM

① Volatile → RAM (Random Acess memory)
(RWM)

RAM

- SRAM
  - → static
- DRAM → بدائية
  - → Dynamic
  - مصنوعة من Capacitor
  - و simple Mosfet
  - مشكلتة بيفرغ بعد وقت

كدة ال CPU هيدخل متأخر

② CPU → mem

① Ref → mem
→ Priority

كدة هيزود ال Access time

Speed ↓

* Based on Capacitor
  - مفرغ ↓ ⓪
  - مشحون ↓ ①

للعلاج بحتاج Refreshment Circuit

PAGE
DATE

DRAM:- Advantages

1. Simple Hardware
2. low cost Per bit
3. high density (size↑)
4. low Power Cons. (Mosfet) Power

E-S

⇒ SRAM (Static) faster اسرع

↳ Based on transistors

bit → flip flop

min → 6 transistor

(high cost per bit) مشكلة *

Refreshment circuit

* Cache memory

يتبع الكاش

(Parallel memory)

E.S complexity	SRAM	DRAM
Size	↓ اصغر	↑ اكبر
Cost	↑	↓
Performance	↑ اسرع	↓ بطيئة
Power consumption	↓	↑ عشان بيحتاج Refresh circuit

lec 2:-

Non-Volatile → (RoM) Read only memory (Program memory
بتشيل ال Code

→ Data access time RoM ↑
↓ سرعة
ابطأ من ال RAM

RoM → Based on FG-Mosfet

floating Gate → مبين مادتين عازلتين

→ Programming state (0) (−)

→ erasing state (1) (+) الشحنة

PAGE
DATE

الشحنة (FG)

سالبة → (0) Programming state

موجبة → (1) erasing state

بالنسبة
RoM → (MP) Processor

ابدا
RAM + Battery

RoM
- Mask Programable RoM
- P-RoM
- EP-RoM

1) Mask Programmable RoM
once اي
محرق ال Code مش هتقدر تغيره حتى لو نزلتة
→ OTP → one time Program → زي اطفال

(مصنع) هينزل ال Code

Bios Chip
Basic input output system → في موجودة في الجهاز محطوطة على ال mother board

2) P-RoM → Programmable RoM
→ OTP
(user) هينزل Code

PAGE
DATE

③ EPROM → Erasable Programable Rom
↓
(UV) ← مش بسهولة

Adv → ① Erasable
② Non-Volatile

dis → ① Noise, Radiation → Corruption (بيتأثر)

⇒ Hybrid :- (الحلو من)
→ RAM → Read + write
+
→ ROM → Non-Volatile

Hybrid Adv

E²PROM

- electrical Erasable programable ROM
- ① electrical (بيتمسح بالكهربا)
- ② Endurance ↑ → 100k (تقدر تتحمل) Read and erase cycles
- ③ Byte-Access
- ④ high Cost Per bit

E² Prom → internal / external

Flash

أسرع من الـ E²PROM

- Acess sector (faster) by sector (Block Access)
- Endurance (10k)
- low cost Per bit

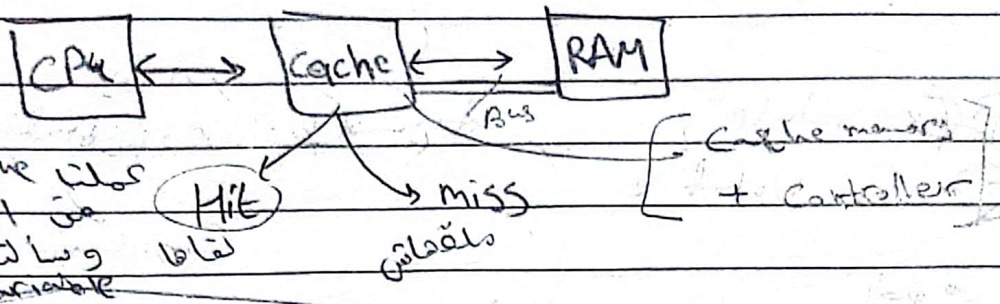
NV-RAM

- SRAM + Battery
- SRAM + E²P-ROM + small Battery

قبل ما البطارية الصغيرة بتفصل الـ SRAM بتنقل الداتا للـ E²Prom
ولما تشتغل تاني الـ E² بترجعها

PAGE
DATE

Cache memory → SRAM → high cost per bit
static

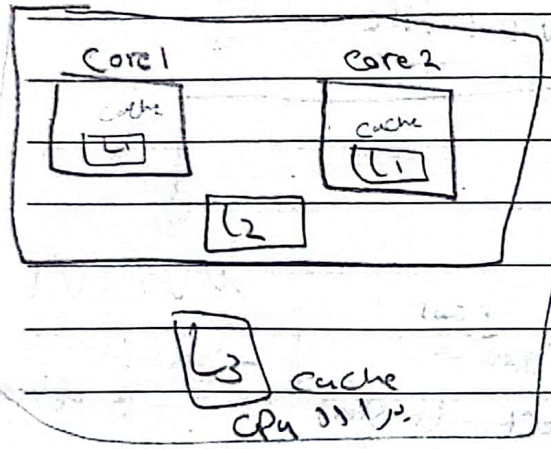


Cache كمان
Address من
وسائت كل
Variable

لقاط

ملقطش

MPU



faster L1 → level 1 cache
L2 → level 2 cache
slower L3 → level 3 cache

*Cache Coherance
او حصل اى
تغيير فى الداتا
بيسموا بيها Controller

* ممكن اعدل فى Cache من غير ما اعدل فى الـ Ram

الـ CPU اسرع حاجة وده يخزن كل
مرة فى الـ Rigester و الـ R
الـ data و نقاط من Address
ما بتوصل بسرعة

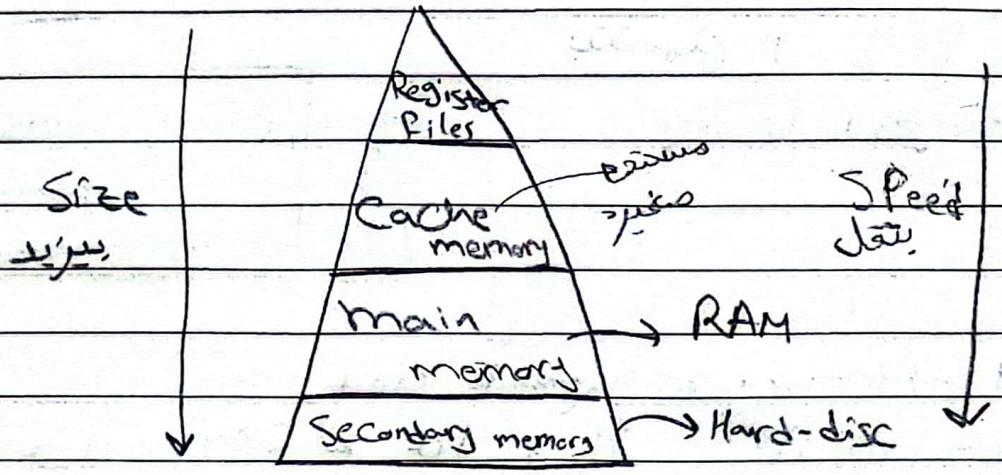
مش محتاج اعمل
Address لكل
Address bus

PAGE
DATE

⇐ ثانى اسرع حاجة ال Cache memory اسرع من ال RAM

من سرعة عشان في جوة memory ال Core

⇐ كل ما نبعد عن MP ال Core ⇒ بنبقى ابطأ



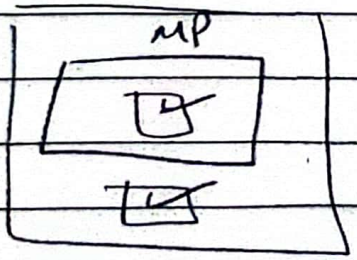
* Hit ratio :- $\frac{\text{no. Hits}}{\text{total}}$ (Hit + miss)

عدد المرات اللى ممكن اعمل Hit فيها

FPU = Floating Point unit

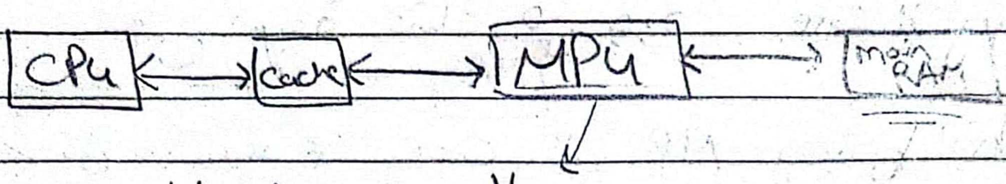
* وظيفتها - عشان ارقام float وعايزة اعمل عملية حسابية

MC



Performance بتزود

*MPU :- memory Protection unit



ال Hardware circuit بتاعتي

يتقسم ال RAM لأجزاء

يتقسمها Ranges

RTOS → نظام تشغيل

*MMU :- Memory managment unit

لو ال RAM = 1G و ال APP بتاعي 8G

فال MMU بطريقة virtual بيفرد مساحة

منها يعني انا هشغل 1G

---

*Adress Bus → uni-direction طالع من ال MP

*Data Bus → Bi-direction

*Control Bus → 4 lines (Active low)

- $\overline{MWTC}$ → RAM
- $\overline{MRDC}$ → ROM, RAM
- $\overline{IOWC}$ → output Device
- $\overline{IORC}$ → Input Device

PAGE
DATE

CPU — mem

Bus-set (data, Address, control)

Architecture

Von-Neumen → PC → مقدرش اكلم الاتنين At the same time

one memory-System

Harvard → MC

RAM — MP — ROM

RAM 0–255 | ROM 0–255

MP (Fde) RAM 0–200 | ROM 201–500 | I/O 501–1000

Address كل حاجة ليها *

الخاص بيها

راكي بيفضل انا بتكلم مين

S.W (Software)

RAM

Assembly instructions → Load/Store

ROM

Assembly instructions → Read/Write

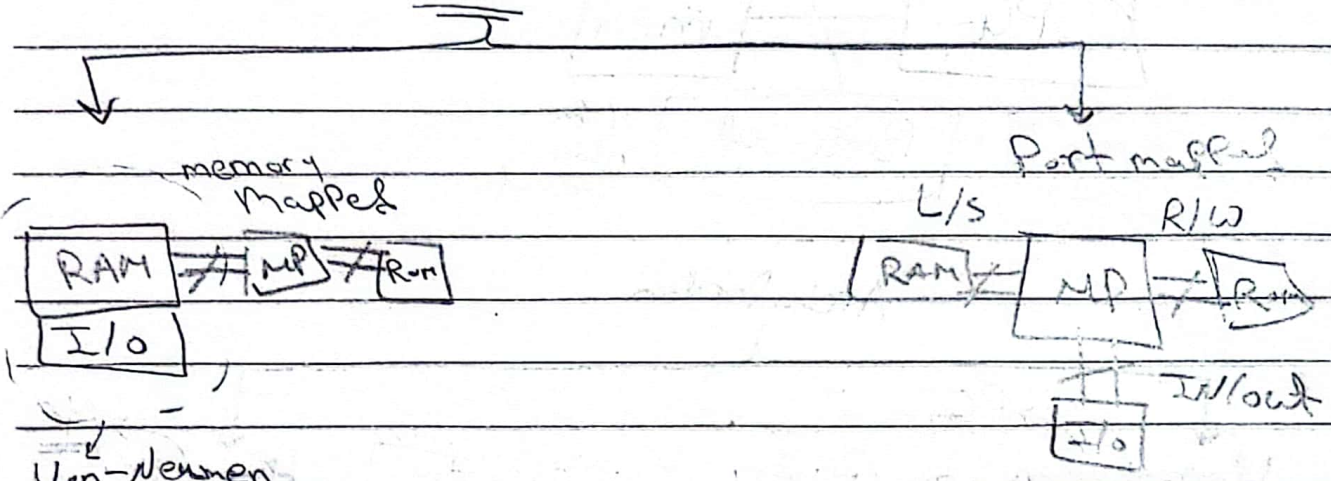
C-language

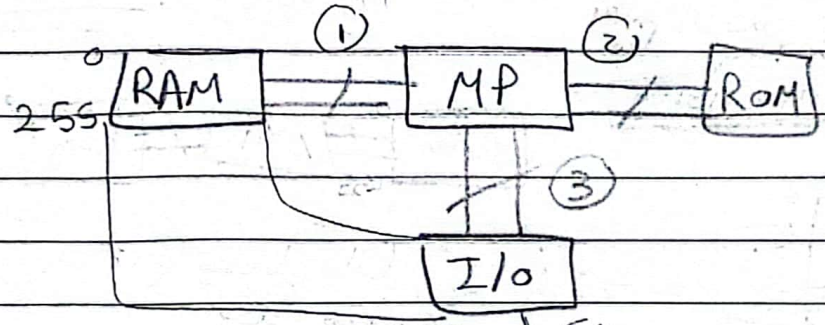
Compiler ✕

محولة

PAGE
DATE

I/o (Harvard)





عشان الكلم الـ I/o معايا طريقتين

Bus-set ١ كلم

١ → C-language

Load/Store

اسرع

٢ كلم

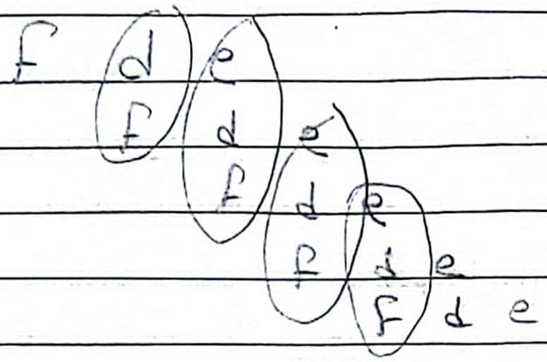
Bus-set ③

Assembly

IN/out

→ **Pipe line** → اسرع بدل ما استنى ال fde الاول يخلص

في كل Cycle
بعمل بدل
f d e



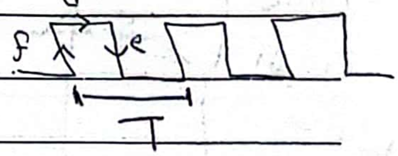
* Von-ne → Can't Support Pipe line → لانه مينفعش اعمل f للداتا و store at the same time
* Harvard → support Pipe line

* Risc → Can support Pipe line (one clock cycle)
* cisc → Can't support Pipe line

Inst. → Assembly ← MIPs → million instruction Per sec
Inst.
↓
f → d → e
⇓ depend on
Clock → Square wave

ME → n bit → Data bus

$f = \frac{1}{T}$



Risc → 1 inst → 1 cycle

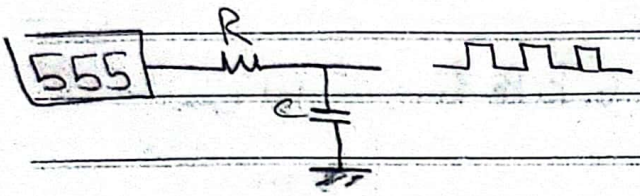
If f = 8 MHZ → 8 MIPs

Clock System

PAGE
DATE

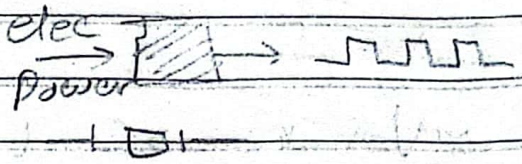
Electrical

RC-oscillator



Mechanical

materials



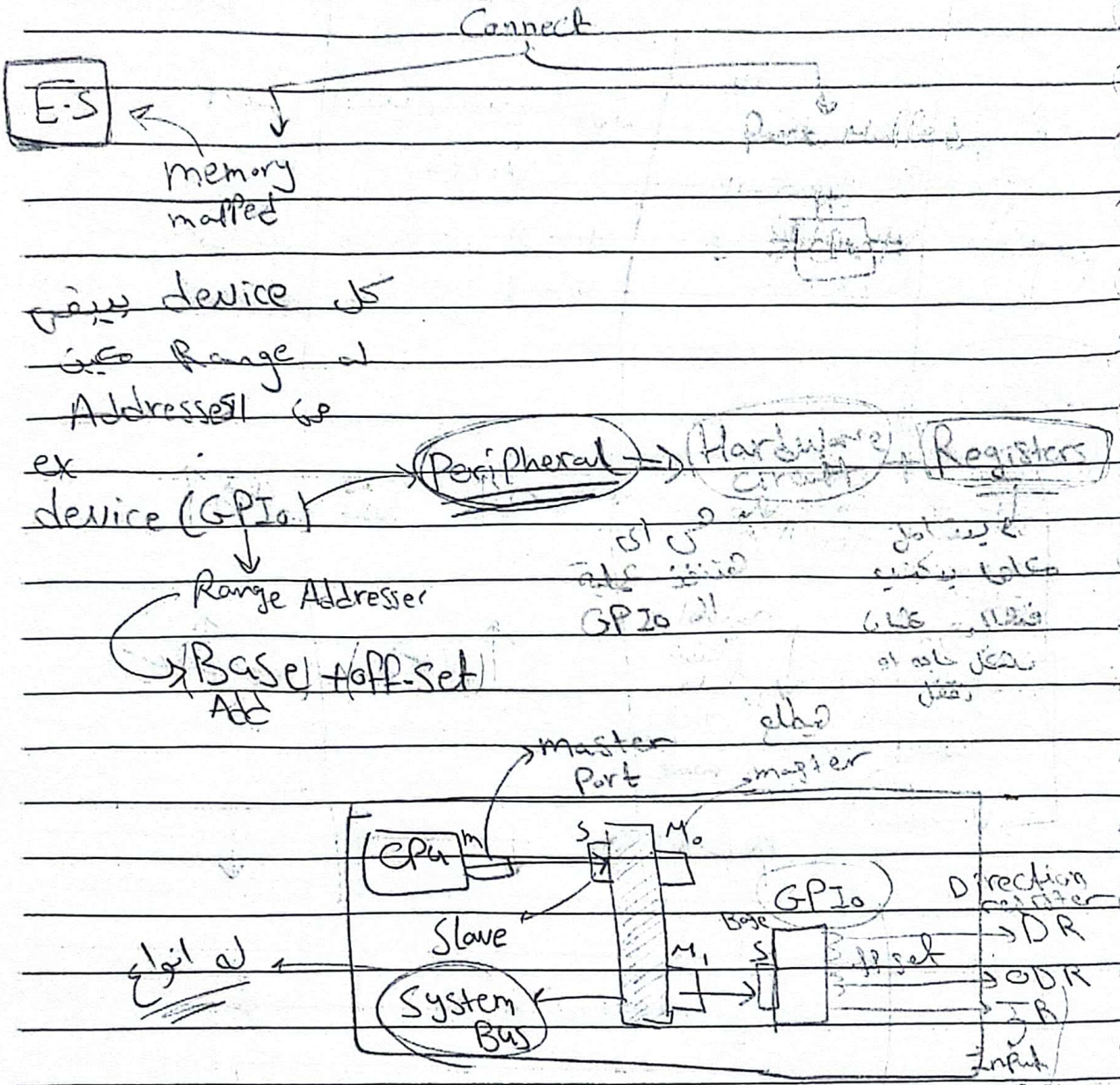
① Ceramic Resonator
② Crystal oscillator

	RC	Ceramic Resonator	crystal oscillator
Cost	↓ (أقل Cost)	~ in between	↑ (أكلى Cost)
Accuracy	↓	~ in between	↑ (أ)
Settling time (الوقت اللي بياخده عشان الـ clock تستقر)	↑ (بياخد وقت كتير)	~ in between	↓ (وقت أقل) (أ)

	RC	Ceramic Resonator	Crystal oscillator
ده اللي هيحدد انا هستخدم ايه ↓	فا ممكن استخدم ال RC في Vib. APP.	اللي ميستحملوش Vibration مناعة ضد ال Vib. هما بيشتغلوا بال Vib.	اكتر معندهمش لانهم
Noise Immunity مناعة → Temp	مفيش مناعة ↓ ضد ال Temp	(✓) ↑ مناعة عالية	(✓) ↑
→ EMI Electric Magnetic Interference (wave)	↓ المقاومة هتتأثر دايماً بال electro magnetic wave	↑ (✓)	↑ (✓)
→ Vibration	↑ (✓)	↓	↓

PAGE
DATE

lec3:-



PAGE
DATE

SPECS → TRM

→ memory map →

GPIO → Base Address → EN → Site of Perif

+Base

GPIO → +DR → 0x... + Base

offset = ODR 0x14 + Base

- IR → 0x? + Base

EX:-

SW → LED ~ يأخذ حالة High على GPIO pin

① Pointer → DR يشاور على

direction Register

1 (o/p)

high

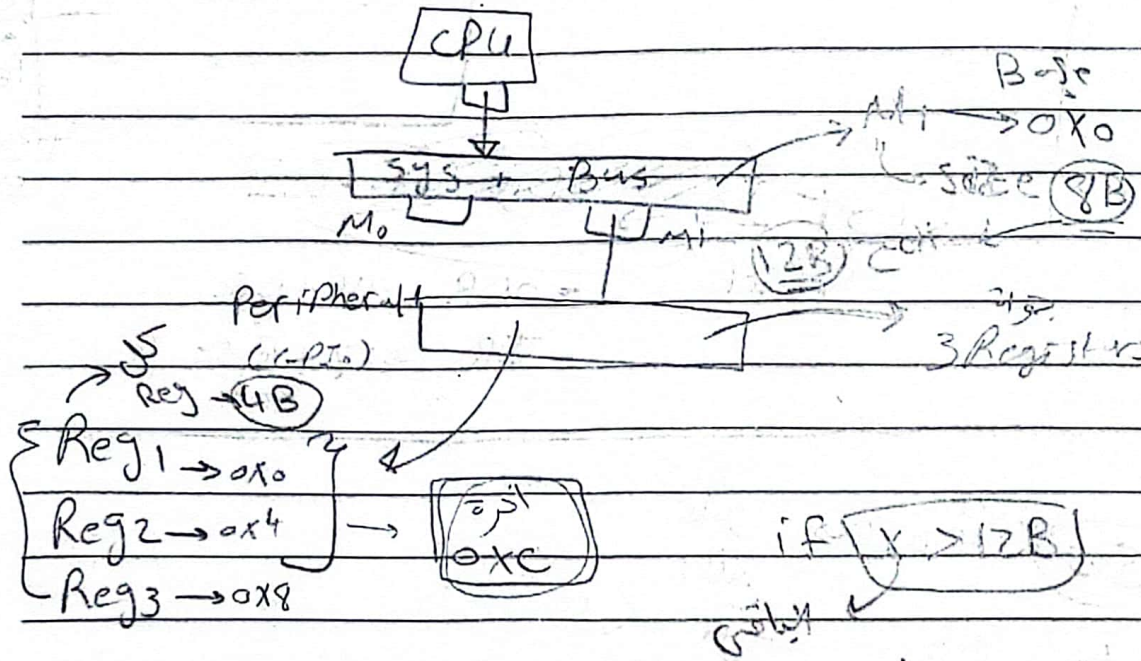
(Active high)

(Base + offset)

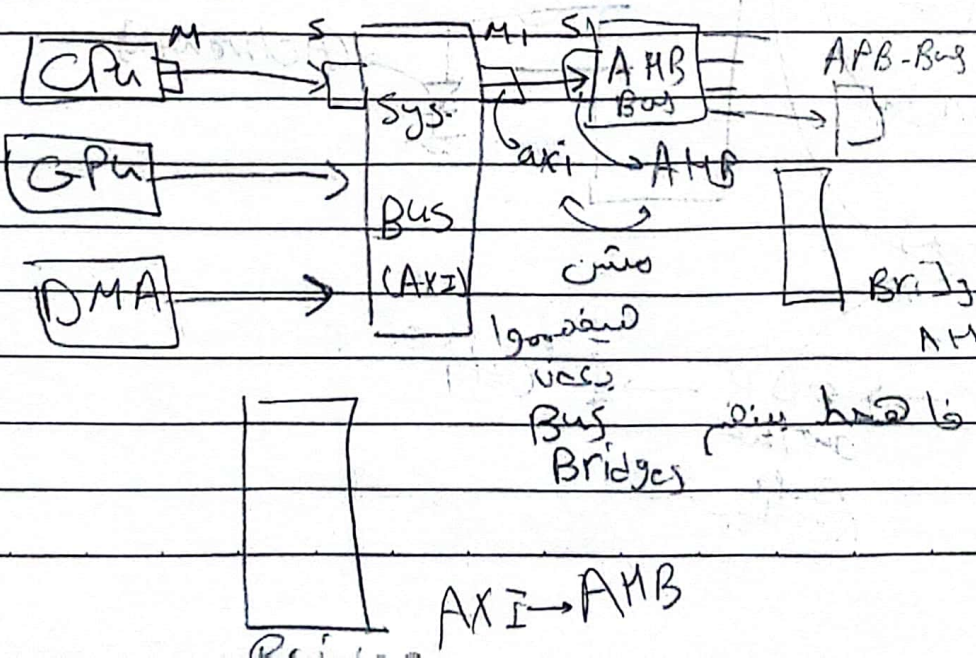
② Pointer → ODR → 1 (H)

Out Put Data Register

Transaction :→ Address
بيانات → data
→ Size



* Bus Bridges :-



PAGE
DATE

[1] Using numeric memory Adress directly

```
#define GPIO_ODR  (Base + offset)

void main()
{
    *((volatile uint32_t*) GPIO_ODR) = (1<<16);
}
```

تحول Pointer بيشاور على ال GPIO_ODR

Volatile

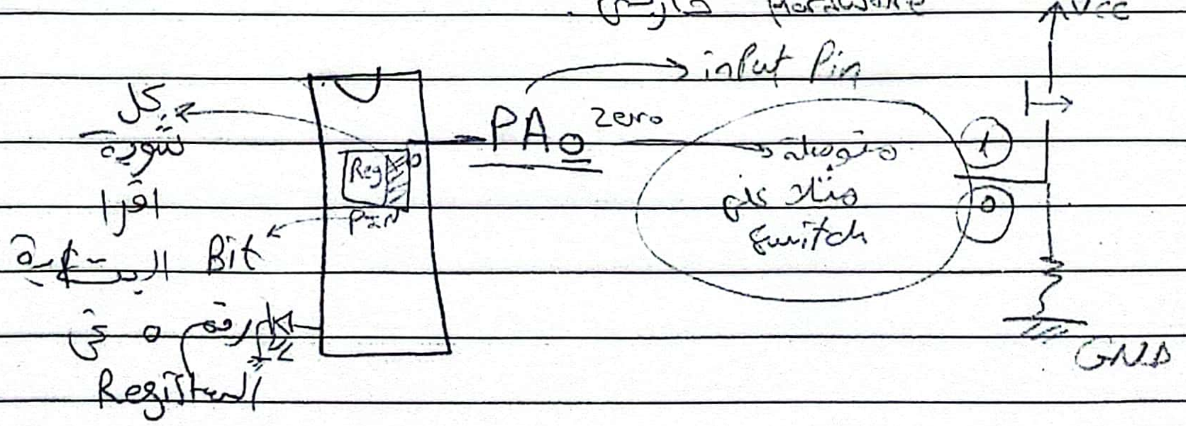
نوع ال Pointer

بيشاور

dereference operator

* بيستخدم كلمة Volatile في ال Register :- امتى
لما ال Pin ممكن يحصل عليها تغير من Hardware خارجي

بيمنع optimization



Pointer

Base + offset

PAGE
DATE

#define GPIo_oDR (Volatile uint32*)

casting address to Pointer

```
Void main()
{
    *GPIo_oDR |=(1<<15)
}
```

[3] #define GPIo_oDR (*(Volatile uint32*))

```
GPIo_oDR |=(1<<15);
```

PAGE
DATE

Hardware Port

two types

master ↔ slave

initiate ... Transaction

Protocol type

Axi | AHB | APB | CHI

digital design

Transaction → Write → (Addr, Data, Size)

Transaction → Read (Addr, Data, Size)