Address Bus

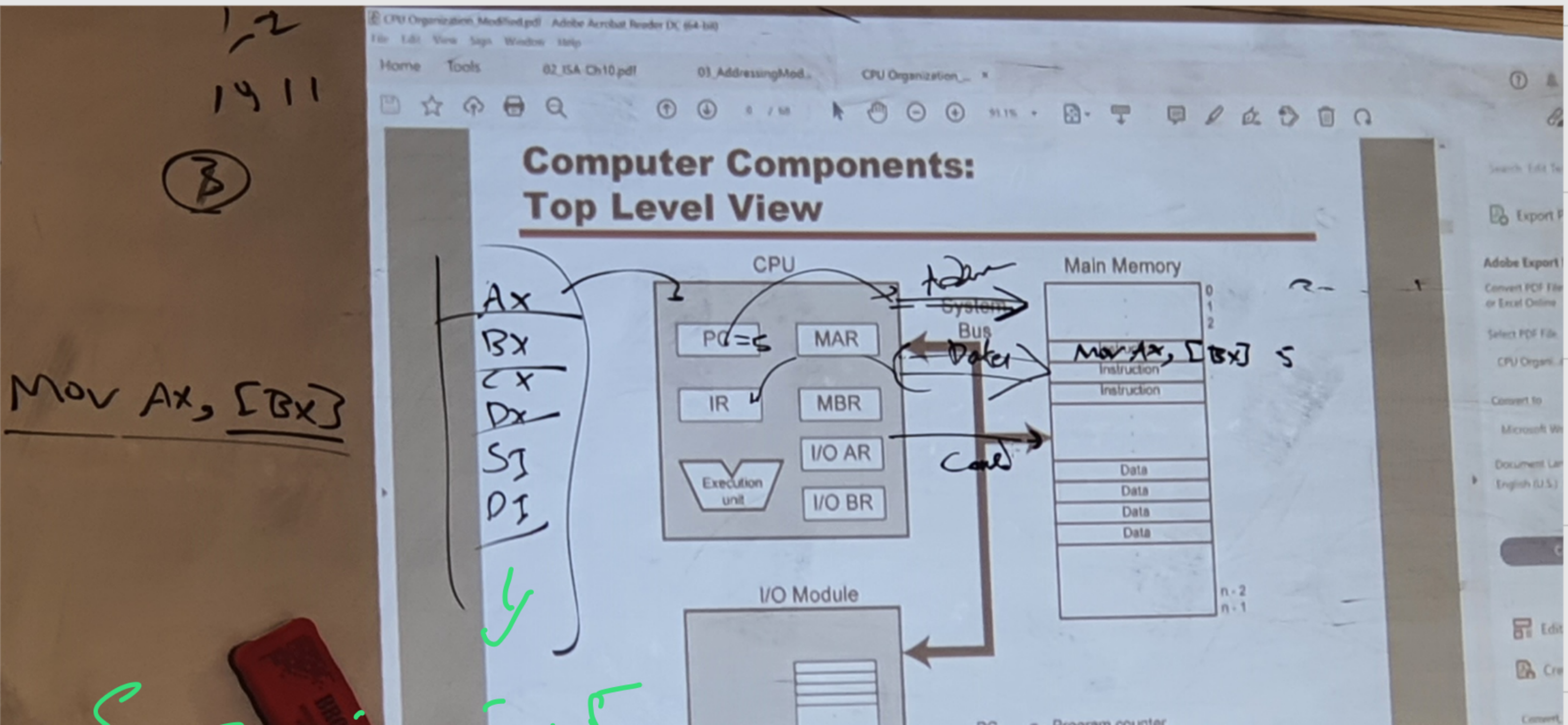
CPU → memory باتجاه واحد

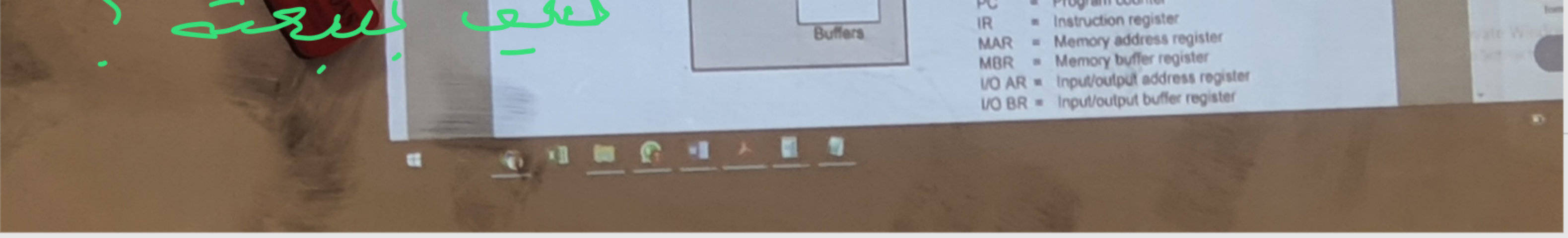
Data Bus

يتنقل الداتا بالاتجاهين

control Bus

instructions CPU → memory


Computer Components:
Top Level View



PC = Program counter
IR = Instruction register
MAR = Memory address register
MBR = Memory buffer register
I/O AR = Input/output address register
I/O BR = Input/output buffer register

16 Bit mux

MAR (memory Address Regester)

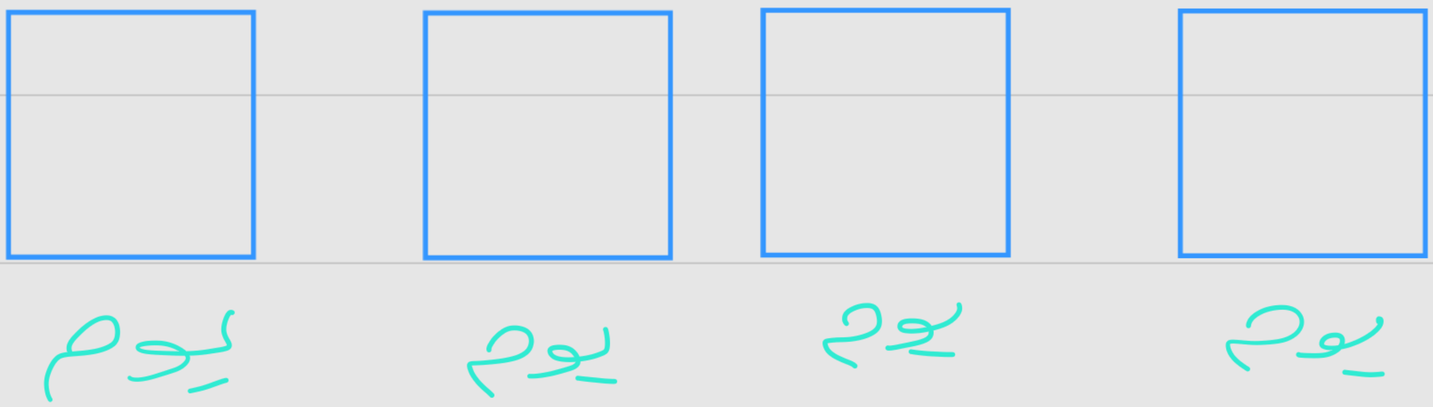
مجموعة بيتخزن فيها Address

عشان ا عملية دي بتطول لتقراه

Pipe line

نشبه اوتوماتيك لصنع سيارات؟

لو بِعرف ٤ مراحل ↙



لو كل مرحلة يوم

بخلي يلي بتخلص شغل تجيب

يلي بعدها وتبلش شغل

⇐ مراحل التصنيع هي مراحل API

ثمّ عليِّ ... ثمّ لما قابلنيِ

ثمّ عليِّ ... يا ولدي يا ولدي

instruction Mov A_x, $[B_x]$

$PC \longrightarrow MAR$

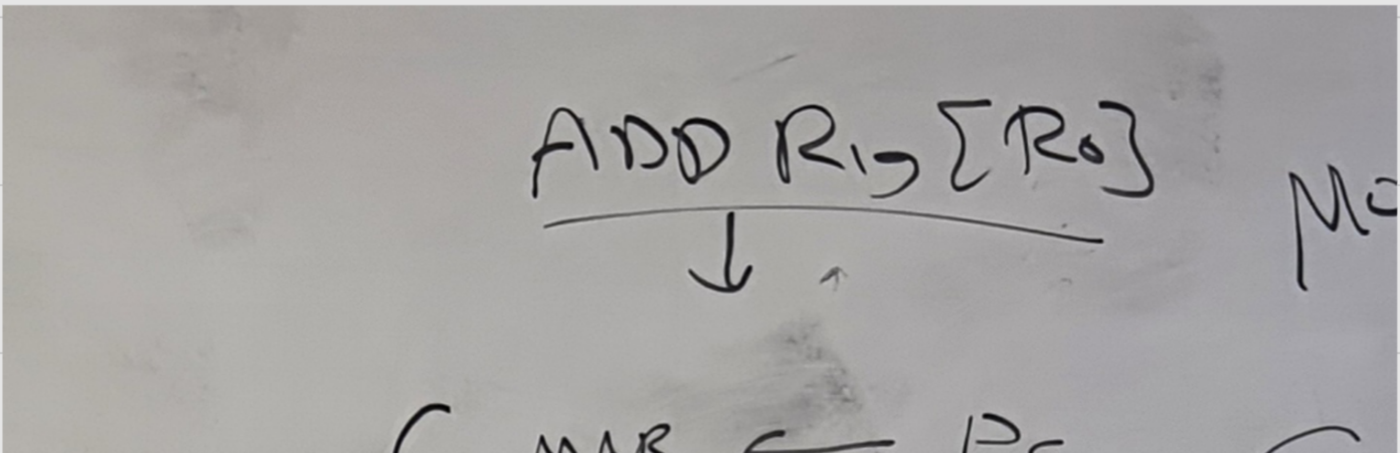
$[MAR] \longrightarrow MBR$

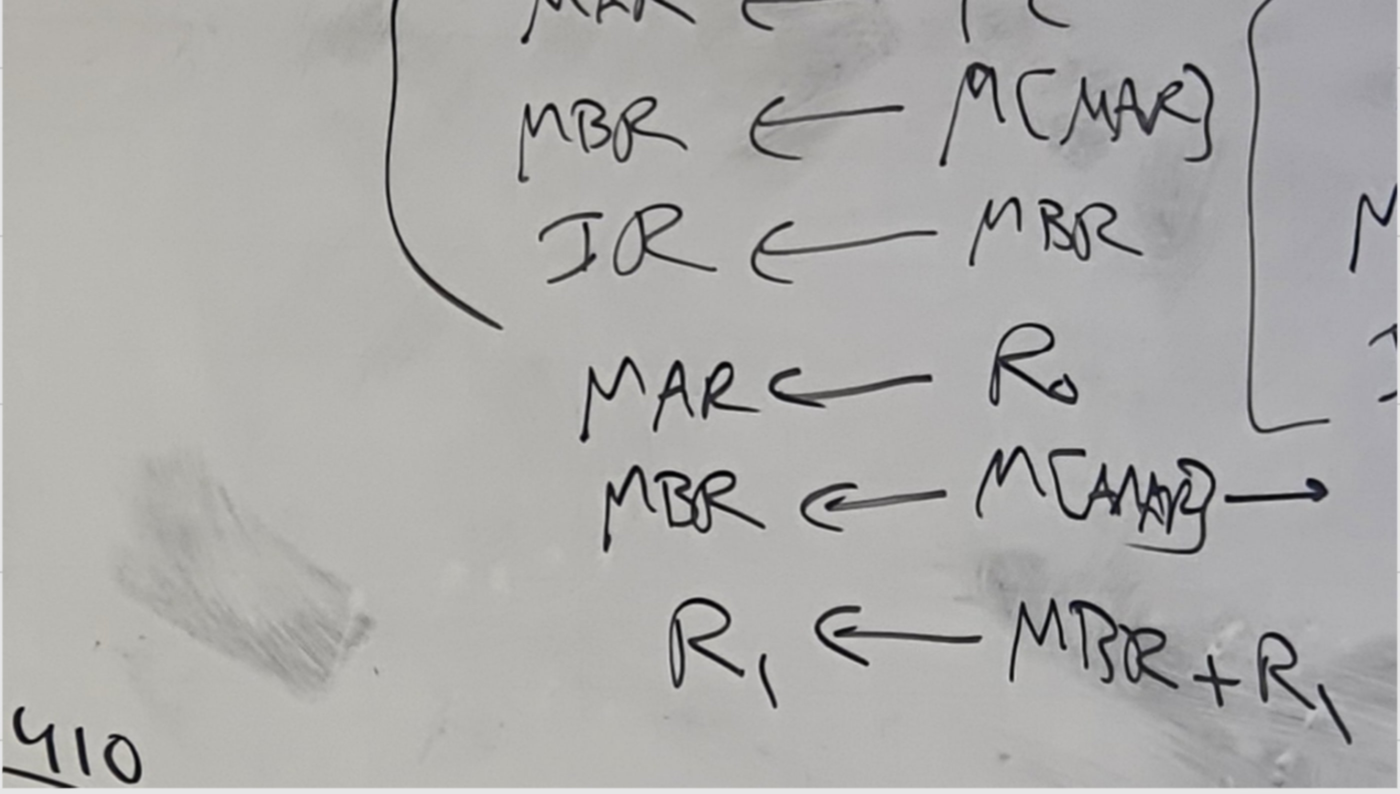
$MBR \longrightarrow IR$

$B_x \longrightarrow MAR$

$[MAR] \longrightarrow MBR$

$MBR \longrightarrow A_x$





$MAR \leftarrow PC$

$MBR \leftarrow M[MAR]$

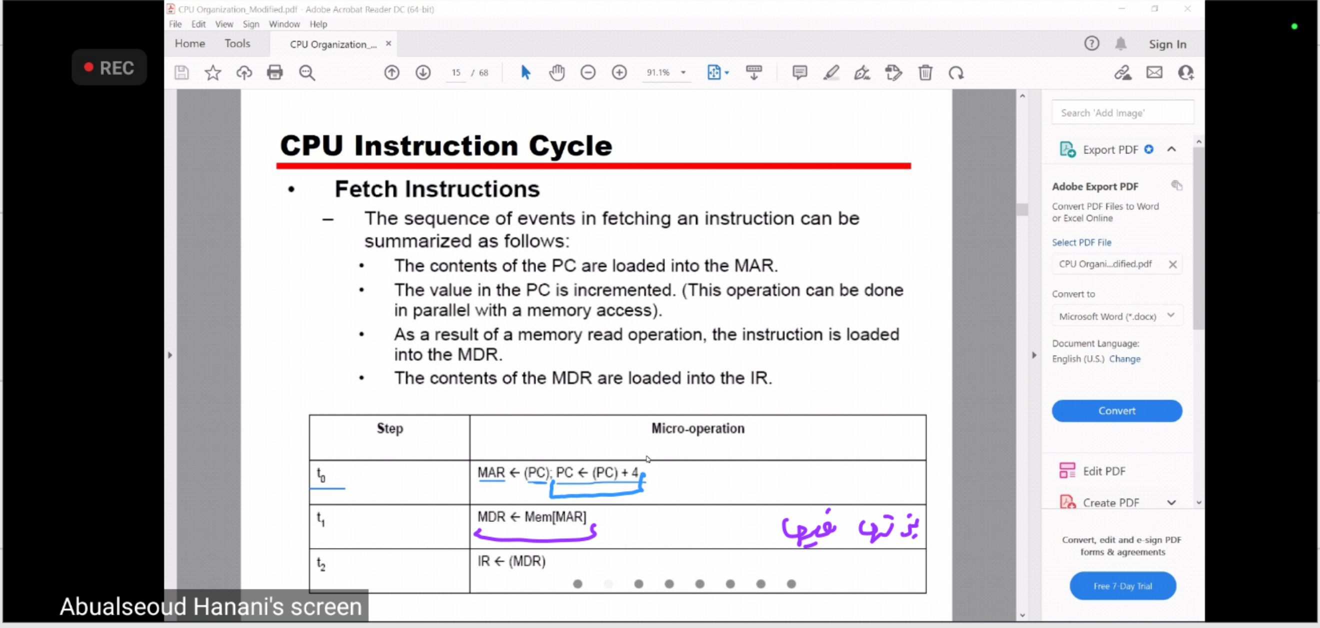
$IR \leftarrow MBR$

$MAR \leftarrow R_0$

$MBR \leftarrow M[MAR] \rightarrow$

$R_1 \leftarrow MBR + R_1$

410



CPU Instruction Cycle

- **Fetch Instructions**
  - The sequence of events in fetching an instruction can be summarized as follows:
    - The contents of the PC are loaded into the MAR.
    - The value in the PC is incremented. (This operation can be done in parallel with a memory access).
    - As a result of a memory read operation, the instruction is loaded into the MDR.
    - The contents of the MDR are loaded into the IR.

Step	Micro-operation
t_0	MAR ← (PC); PC ← (PC) + 4
t_1	MDR ← Mem[MAR] بترتيب فيها
t_2	IR ← (MDR)

بعطي كل طول ال instruction انتفيش

مساحة cell للتخزين [illegible]

instruction => 32 bit

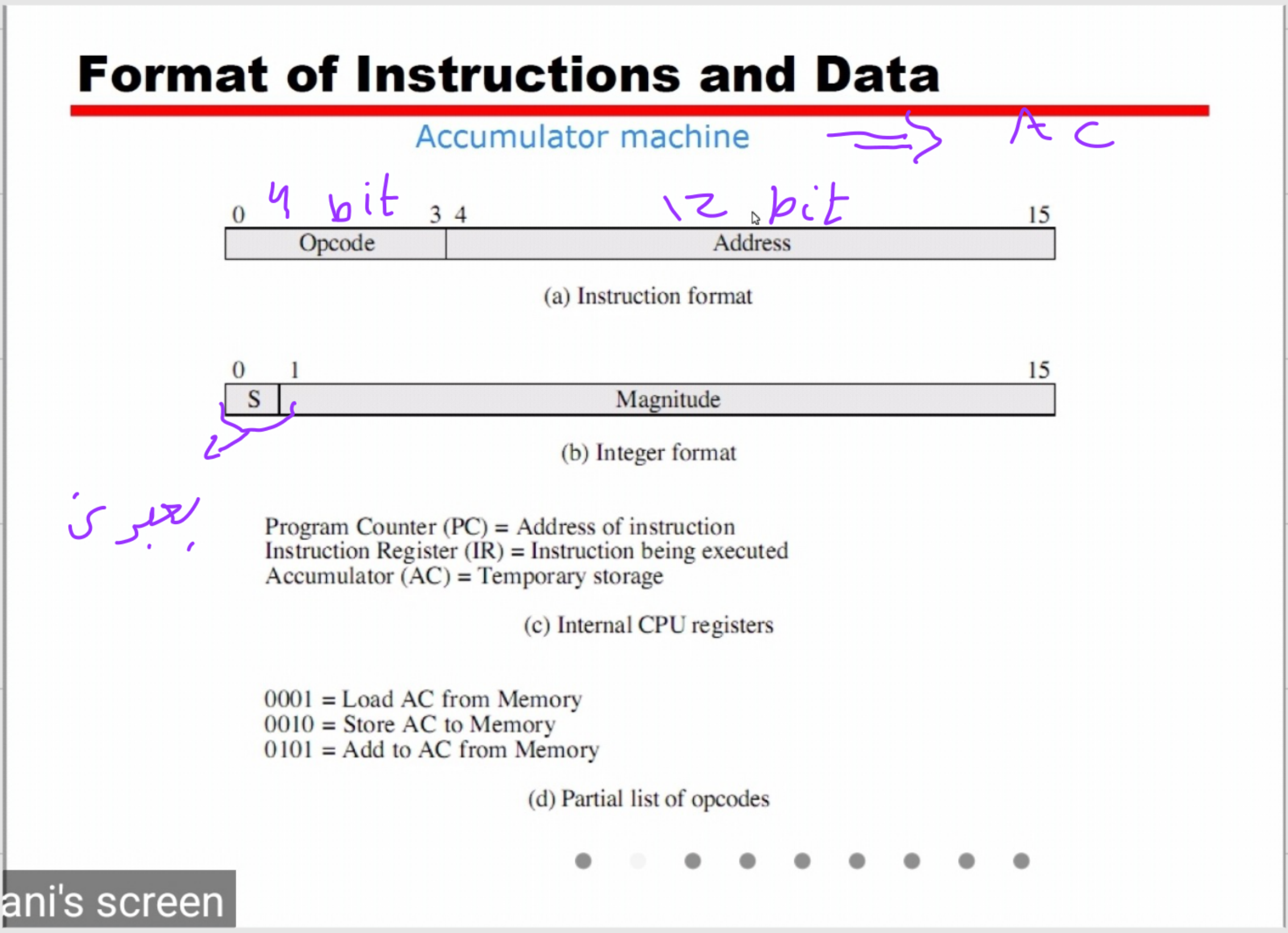
increment by 4

Execute : ADD A_x , [B_x]

t_1 MAR ← B_x

t_1 MBR ← Mem[MAR]

t_2 A_x ← A_x + MBR



Format of Instructions and Data

Accumulator machine ⇒ AC

4 bit 12 bit

0	3 4	15
Opcode	Address	

(a) Instruction format

0	1	15
S	Magnitude	

(b) Integer format

+ve
-ve
يعبر

Program Counter (PC) = Address of instruction
Instruction Register (IR) = Instruction being executed
Accumulator (AC) = Temporary storage

(c) Internal CPU registers

0001 = Load AC from Memory
0010 = Store AC to Memory
0101 = Add to AC from Memory

(d) Partial list of opcodes

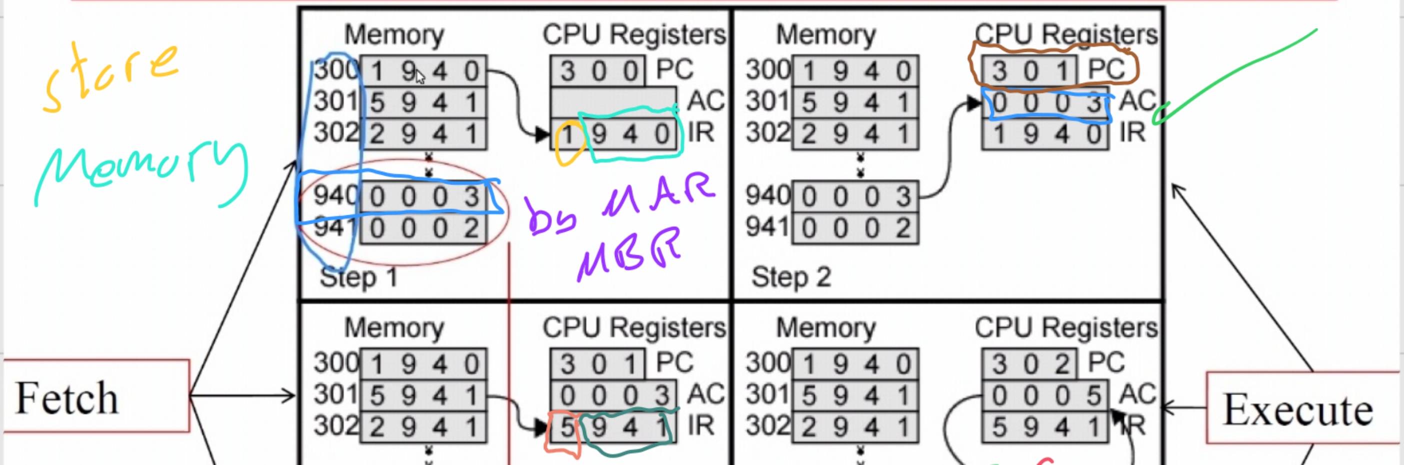
ani's screen

call فنفس كلشي لانه 1 زائد

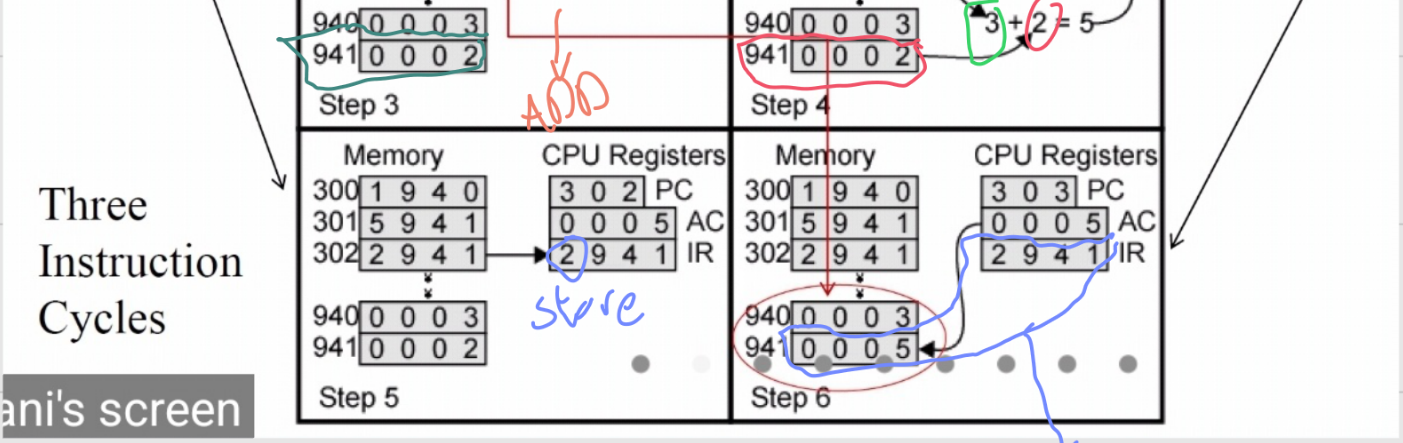
Example of Program Execution

store
Memory

by MAR
MBR



Three Instruction Cycles



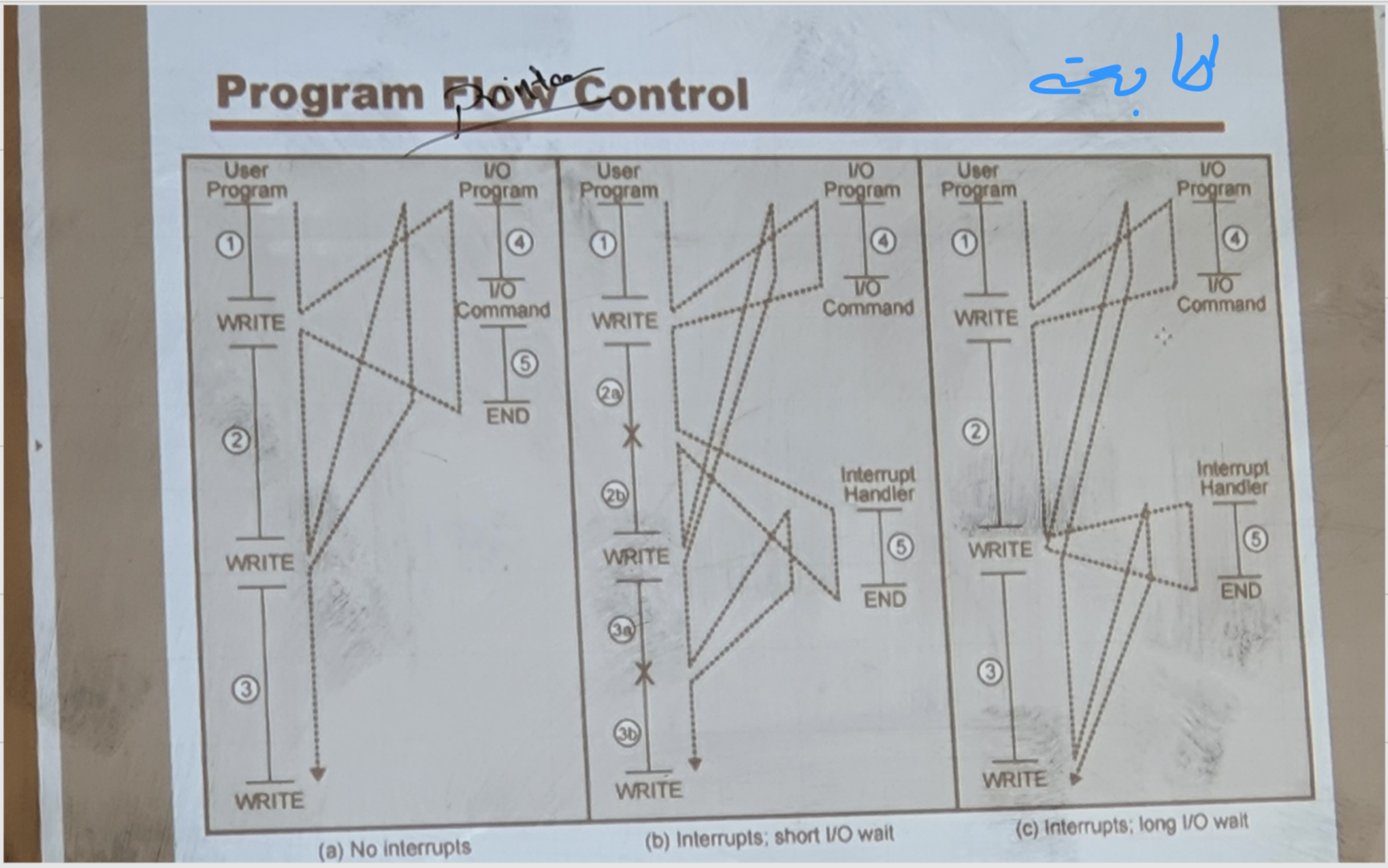
941 [illegible]

MBR length = Data Buses

MAR length = Address length

Program ~~Flow~~ Pointer Control

لما بعت



(a) No interrupts

(b) Interrupts; short I/O wait

(c) Interrupts; long I/O wait

Inteupt ⇒ is a function

↳ ISR

بيكون فيِ Address للفنكشن هاض

int 0

int 1

عناوين كل

int 2

Functions

Adresses

⋮

int n

$\boxed{\text{int } 60} \rightarrow$ interupt مش هخزن قيمة

⇐ بقدر نسويه لغيرها بدنا

Ex: $\frac{x}{0} \xRightarrow{\text{system}}$ let it realize into

---

الكيبات (input) ⇐ عبارة عن

interupt

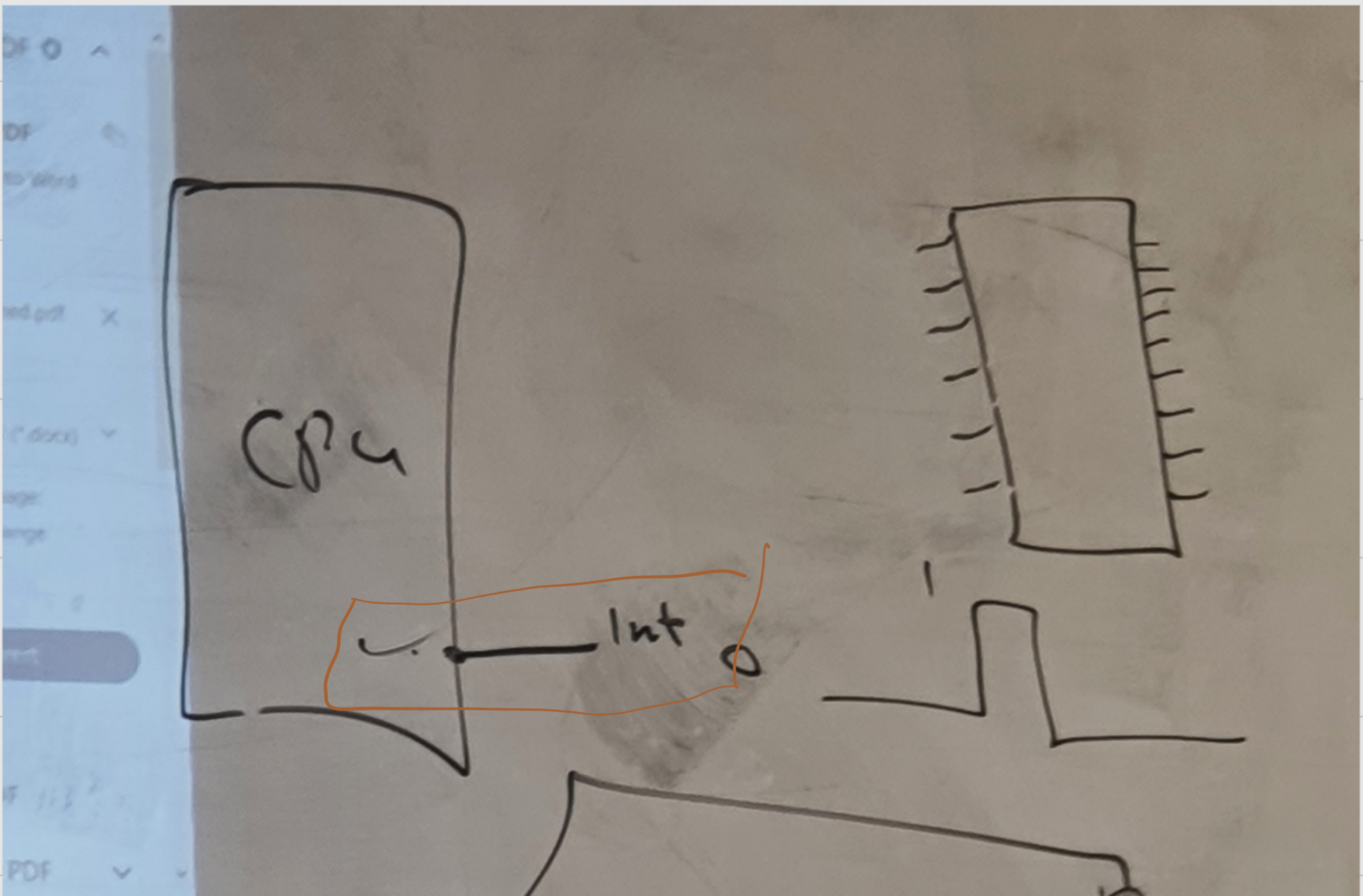
[illegible]

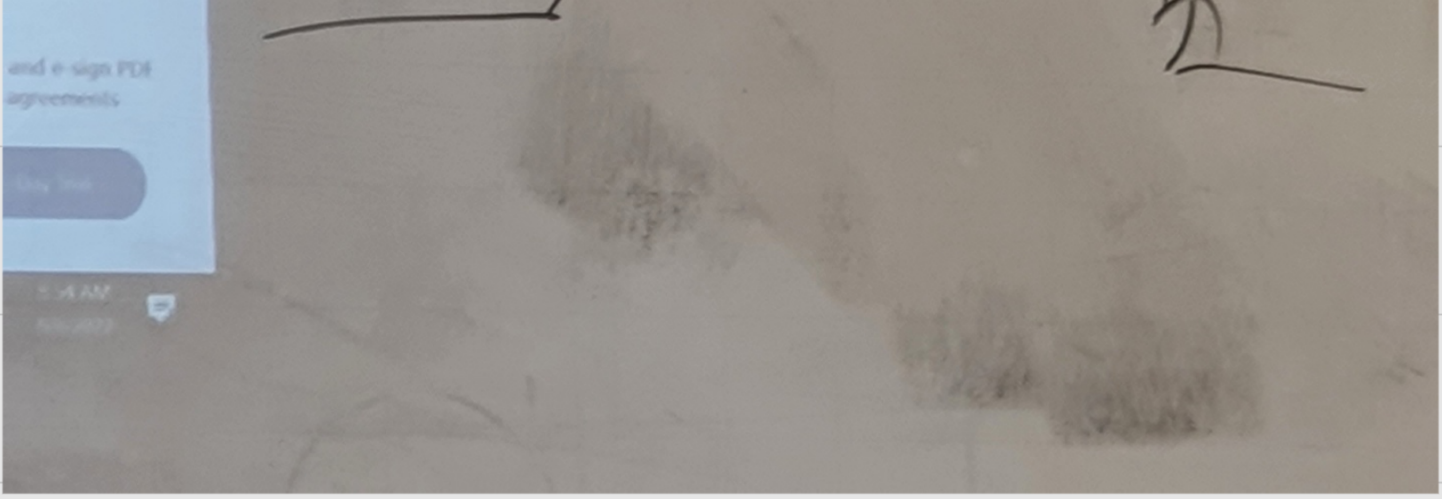
الاشياء الي تعمل interupts تكون اشياء الكل ممكن يتعامل معها

two kind:

Exception = اثناء احطها ب

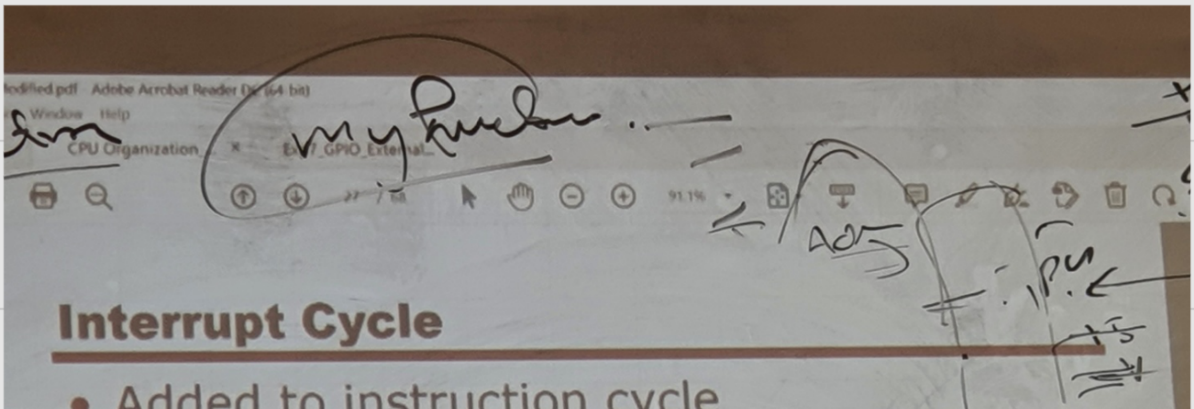
I/O كما تكيبس





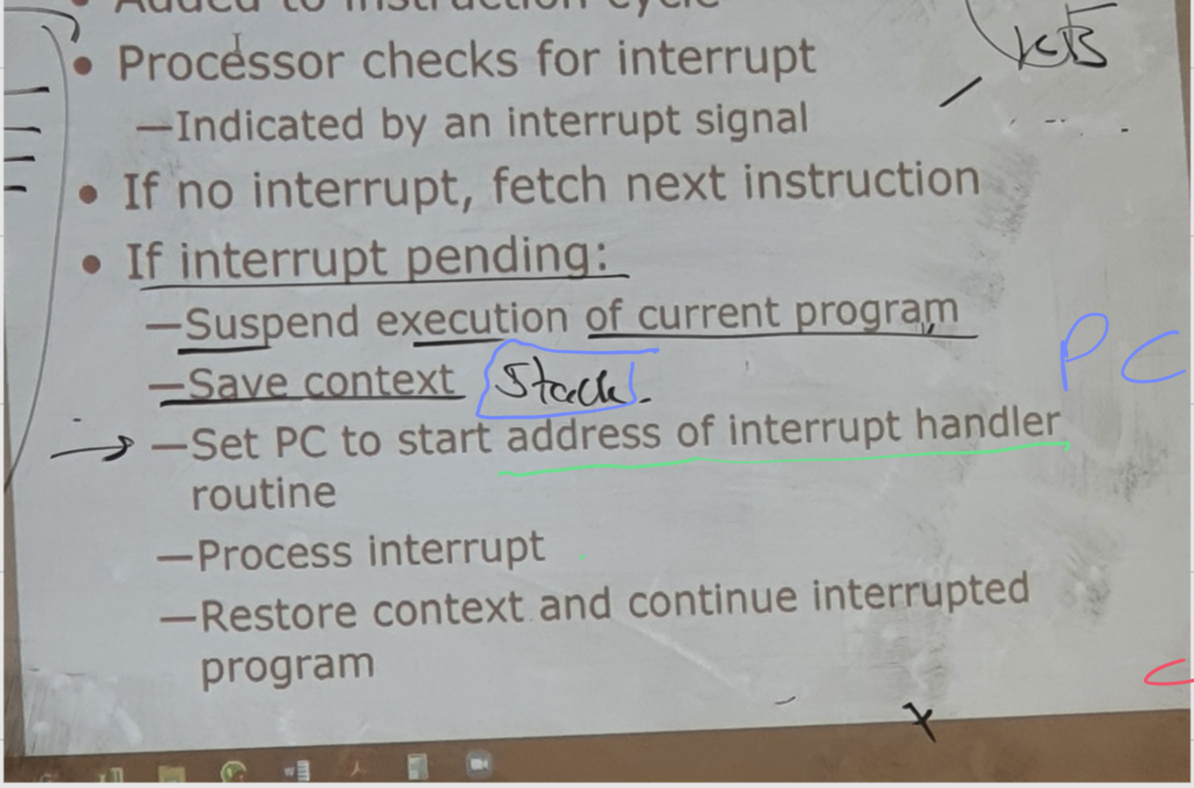
كل ما يخلص instruction
يروح يحل كليها

الطريقة المستخدمة هي انه
يضيف اني بتشغيل instruction
انه يروح يفحص الانبوت



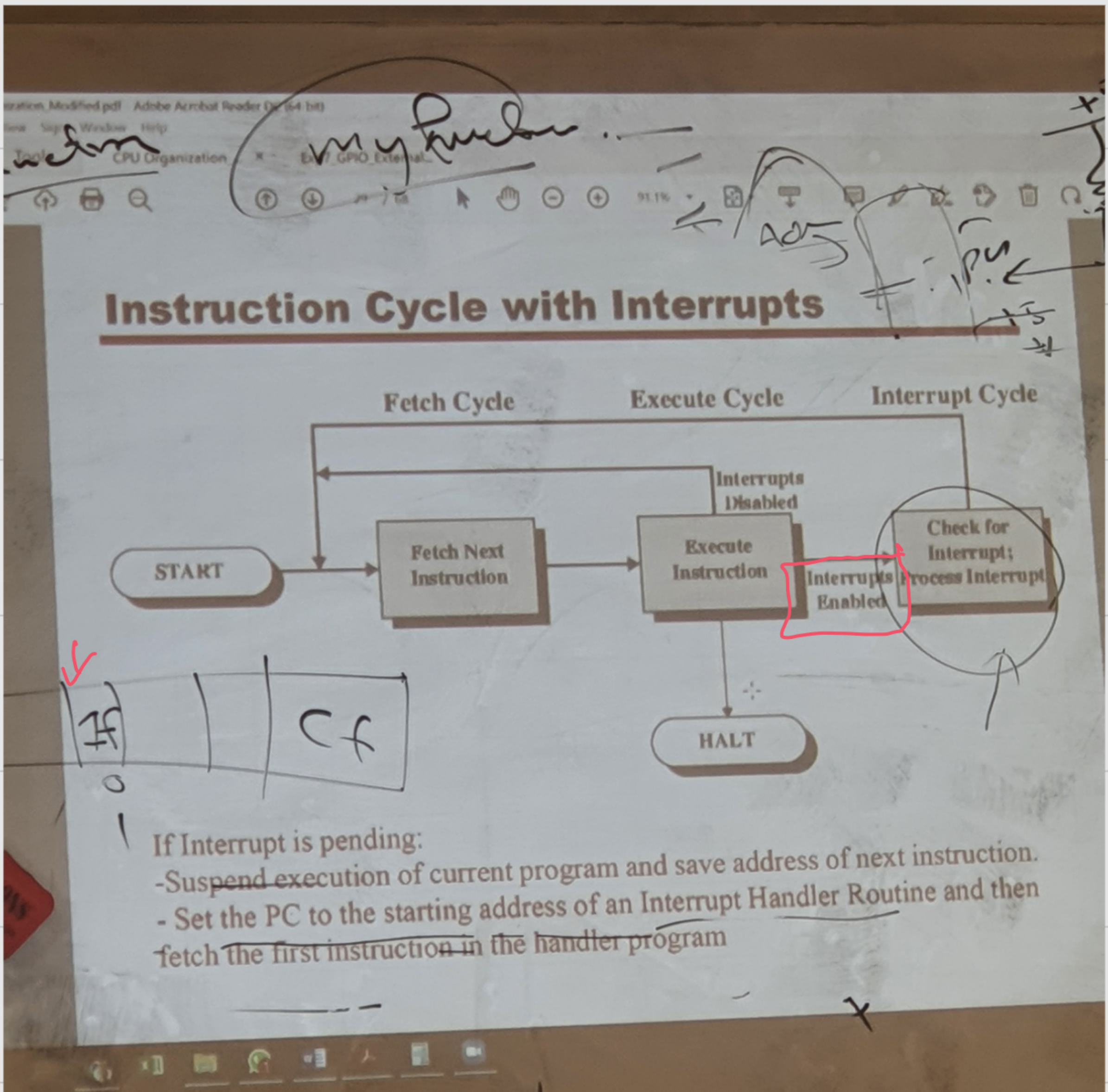
Interrupt Cycle

- Added to instruction cycle



- Added to instruction cycle
- Processor checks for interrupt
  - —Indicated by an interrupt signal
- If no interrupt, fetch next instruction
- If interrupt pending:
  - —Suspend execution of current program
  - —Save context Stack
  - —Set PC to start address of interrupt handler routine
  - —Process interrupt
  - —Restore context and continue interrupted program

PC & flags



Instruction Cycle with Interrupts

If Interrupt is pending:
-Suspend execution of current program and save address of next instruction.
- Set the PC to the starting address of an Interrupt Handler Routine and then fetch the first instruction in the handler program

في flag ذا صت فيه يخلي
ال CPU انه يتجاهل

اذا صار interrupt وانا
اسوي فال int

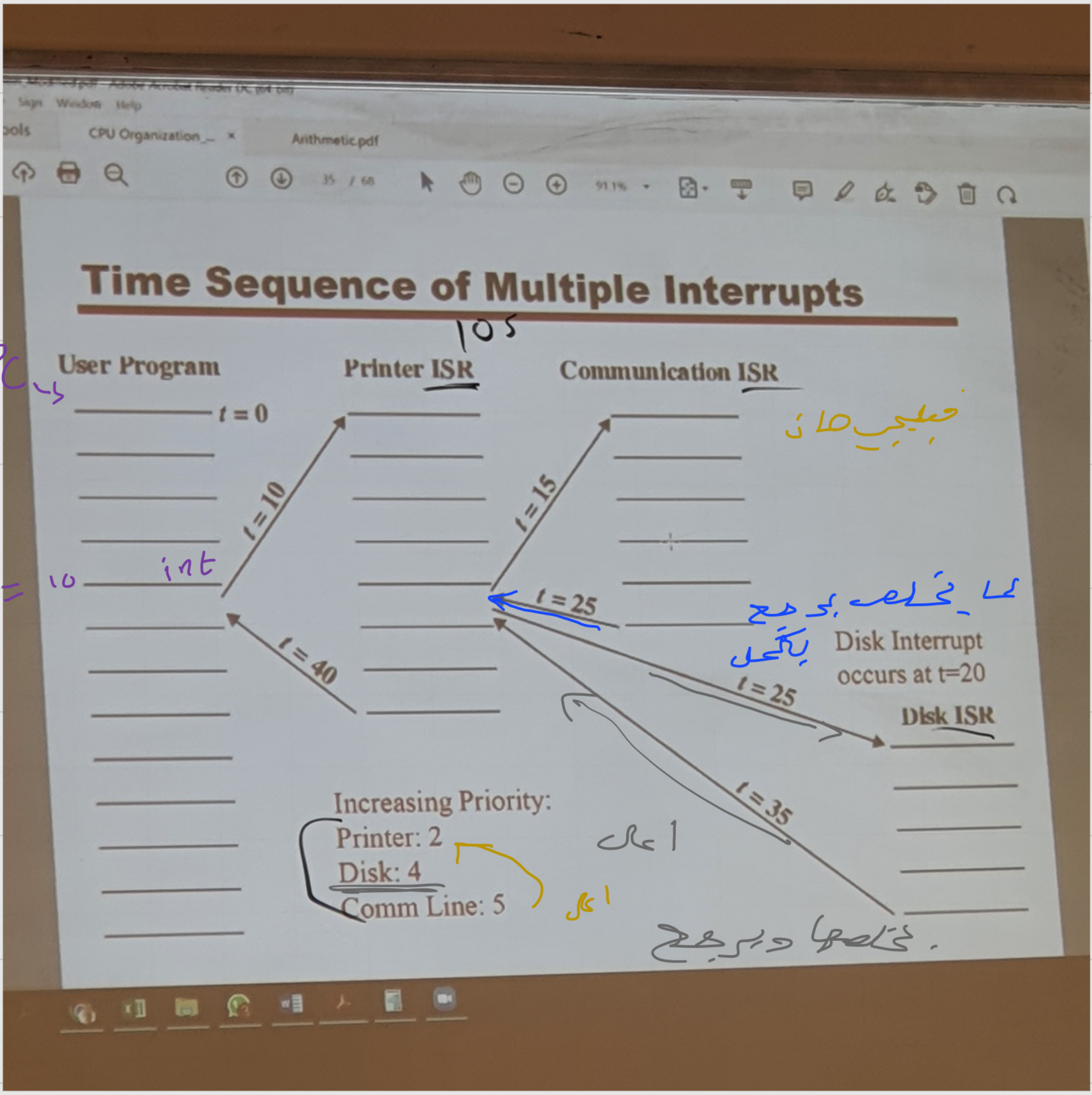
يلي فالاول يشتغل ⇐ Queue
بحيث يروح ينفذ لي اجت بعد

⇐ فلما تنفيذها بسوي Disable
int. Flag

⇐ يتم تخزين جميع ال interrupt لحد
تخلص ⇐ interrupt controller

⇒ priority بنعطي كل interupt قيمة تحدد الاولوية واذا كان يلي اجا بعد الاولوية اعلى بنفذها بعدين بيرجع يكمل الاولى

⇐ بيشوف int controller المسؤول عن هاض الاشي



Time Sequence of Multiple Interrupts

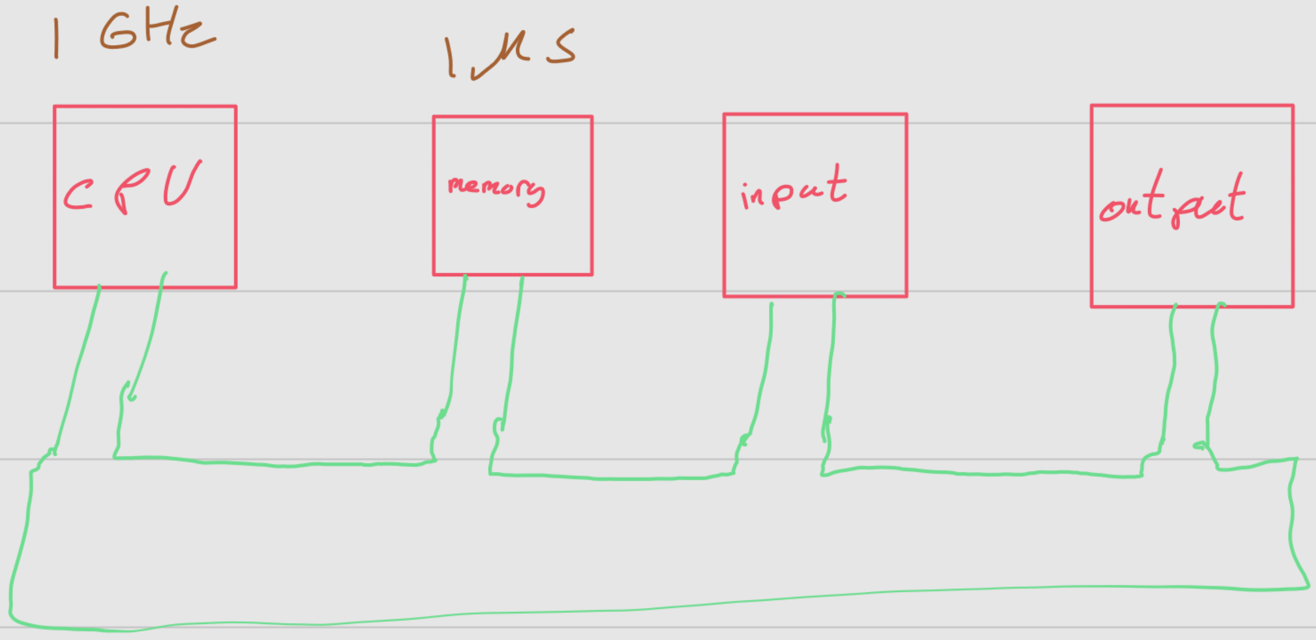
Increasing Priority:
Printer: 2
Disk: 4
Comm Line: 5

when same priority ⇒ Queue

input, output is USB

⇒ isn't the device

what is a Bus?



one Bus between all

وفي switch اه بعمل الوات

⇐ نفس بروتوكول الانترنت

[illegible]

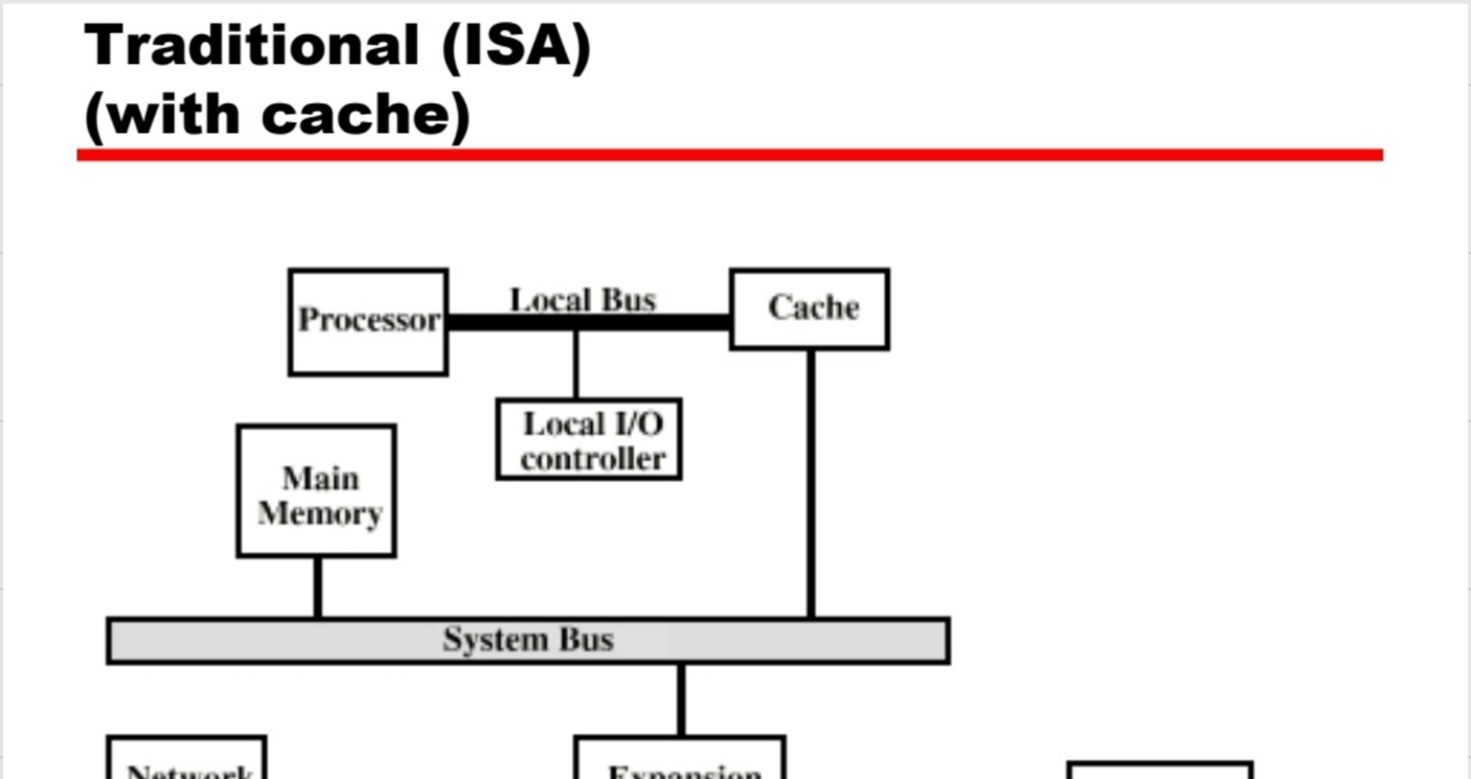
بيلوني جي وقت D زم لصل

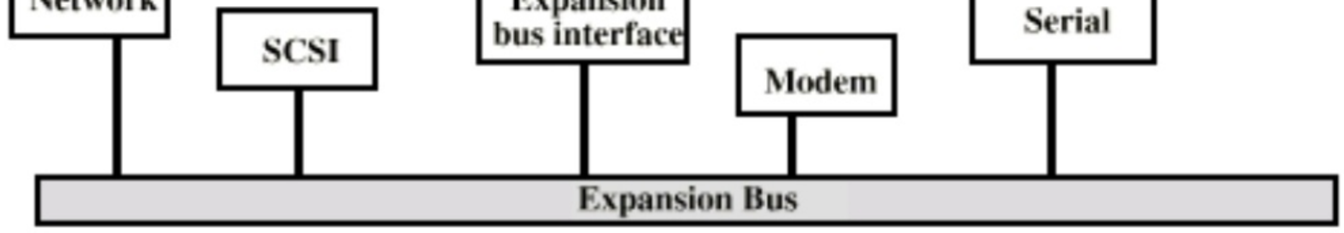
الواتغير ⇐ ليخلص ال memory

قراءات أكثر ⇐ خاطالازم يخليها

1000 clock cicle

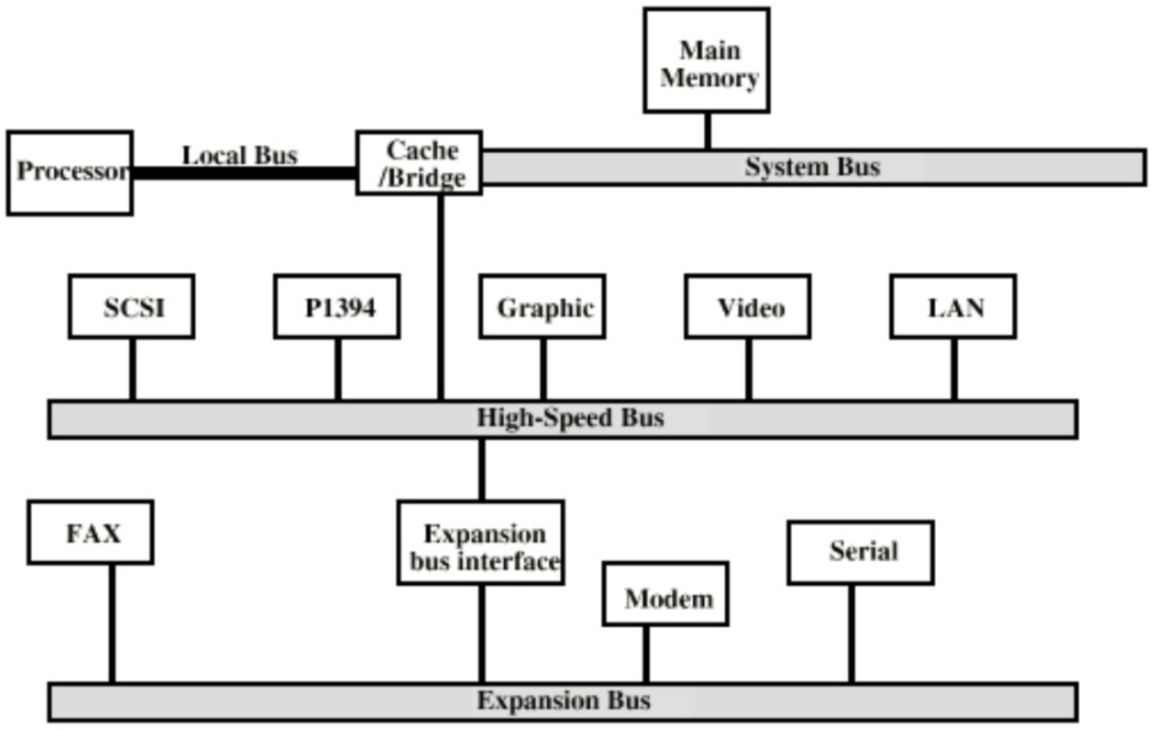
width of data Bus
have big effect on
bandwidth


Traditional (ISA)
(with cache)



لانه كل اشي اله سرعة
معينة ← بخط Buses
بعض يلي سرعاتهم متقاربة

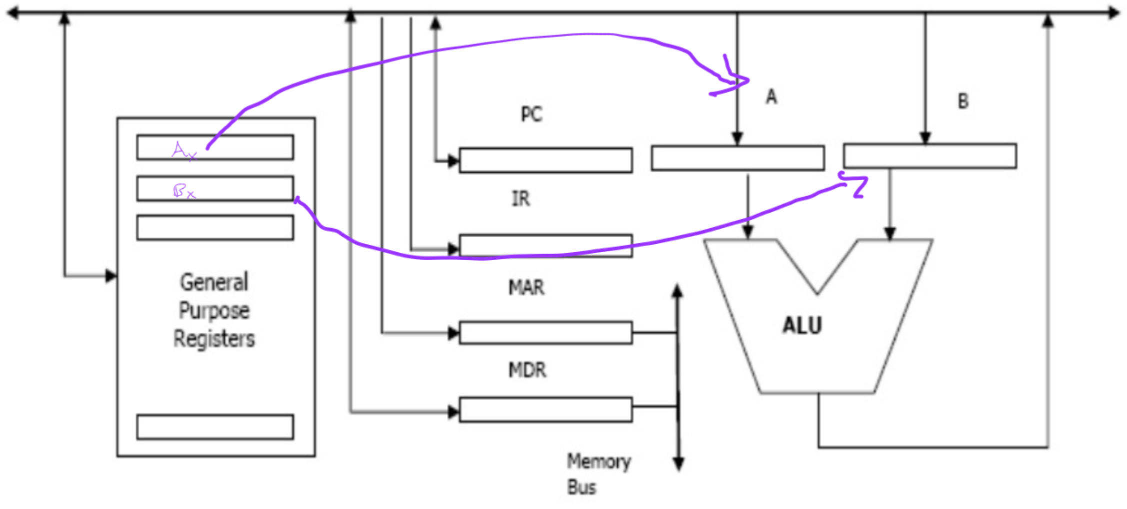
High Performance Bus



لأنه صار تكوير الأشياء
عادوا التقسيم

CPU local Bus Organization

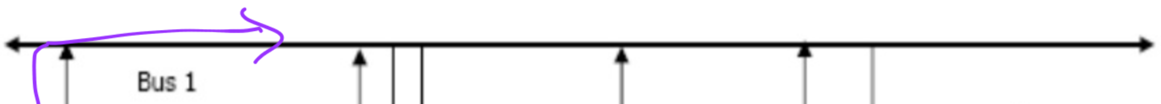
- One-Bus Organization

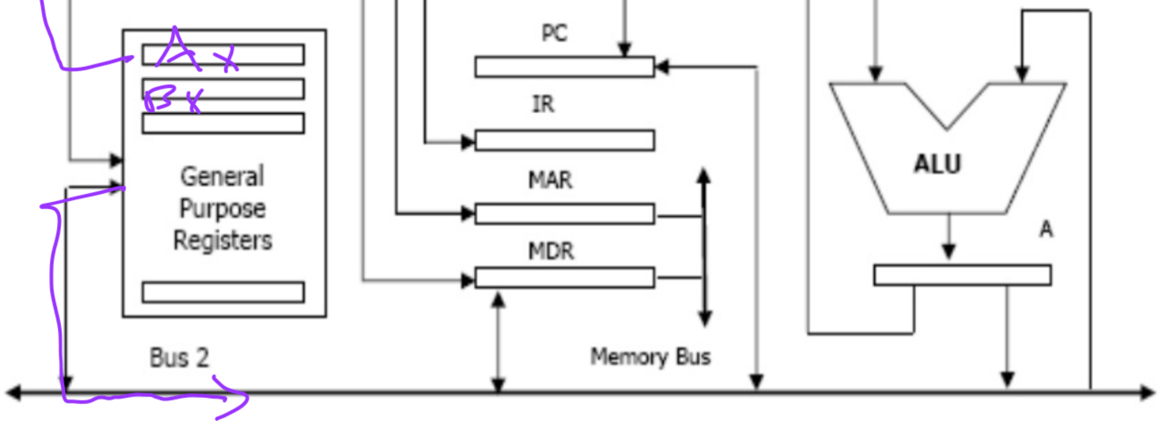


كل وحدة لحال لها 1 Bus

CPU local Bus Organization

- Two-Bus Organization



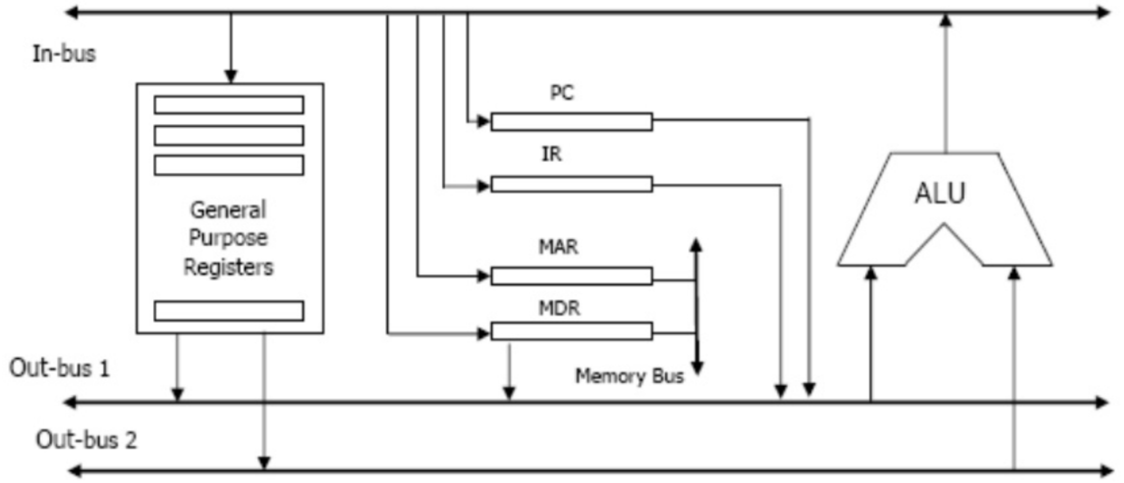


Two-Bus Datapath

هاد في 2 Bus

CPU local Bus Organization

- Three-Bus Organization



Three-Bus Datapath

بحط الانبوت وبطلع بولد
الاوتبوت

ROM: Read only memory

EROM: يمكن برمجتها خارجيا

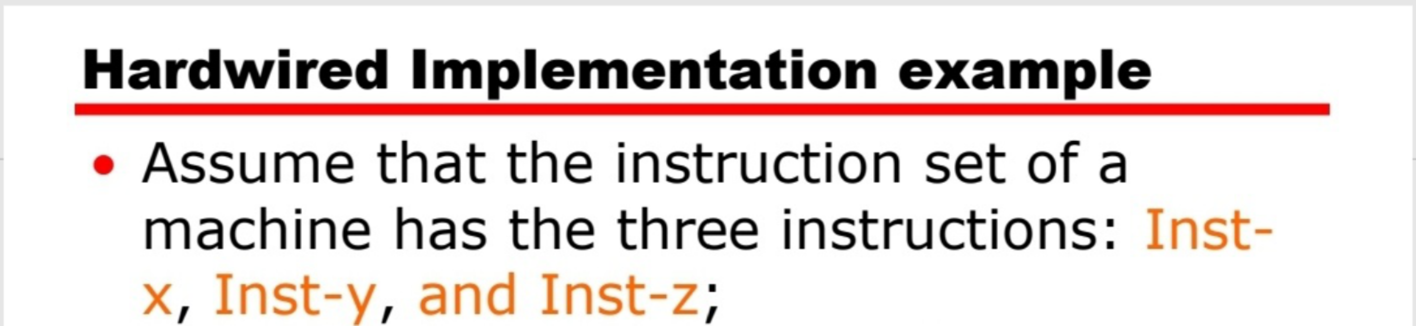
EEROM:

EEEPROM:

Flash: يعاد برمجتها عادي

Hardwired: اللي لا يمكن تغييره

Microprogramed: يمكن برمجته



Hardwired Implementation example

- Assume that the instruction set of a machine has the three instructions: Inst-x, Inst-y, and Inst-z;

- and A, B, C, D, E, F, G, and H are control lines.
- The following table shows the control lines that should be activated for the three instructions at the three steps t0 , t1 , and t2 .

Step	Inst-x	Inst-y	Inst-z
t_0	D, B, E	F, H, G	E, H
t_1	C, $\boxed{A}$, H	G	D, $\boxed{A}$, C
t_2	G, C	B, C	

اذا بدنا A

$A = t_1 \cdot inst\text{-}x + t_1 \cdot inst\text{-}z$

فلما نبرمجه باستخدام Gates

يكون Hardwired

Hardwired Implementation example

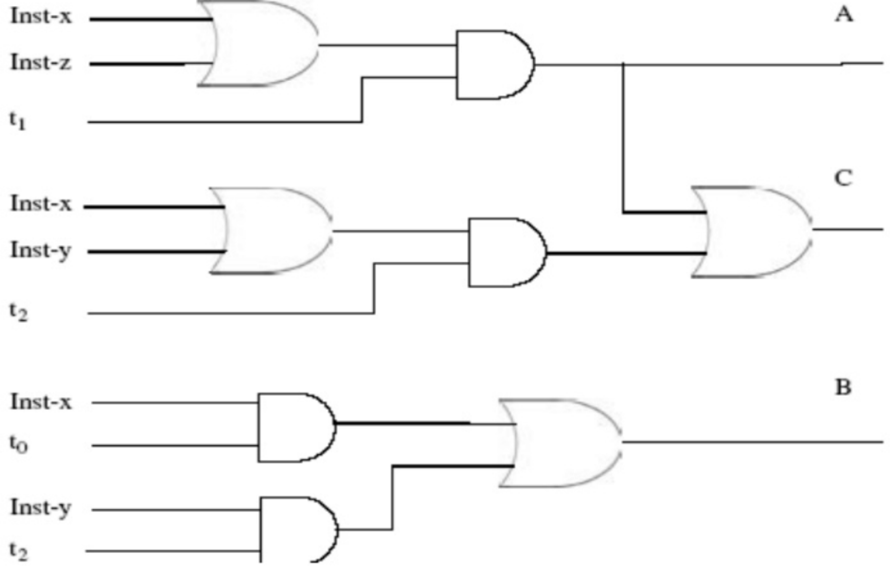
The Boolean expression for control lines A, B and C

$$A = \text{Inst-x} \cdot t_1 + \text{Inst-z} \cdot t_1 = (\text{Inst-x} + \text{Inst-z}) \cdot t_1$$

$$B = \text{Inst-x} \cdot t_0 + \text{Inst-y} \cdot t_2$$

$$\begin{aligned} C &= \text{Inst-x} \cdot t_1 + \text{Inst-x} \cdot t_2 + \text{Inst-y} \cdot t_2 + \text{Inst-z} \cdot t_1 \\ &= (\text{Inst-x} + \text{Inst-z}) \cdot t_1 + (\text{Inst-x} + \text{Inst-y}) \cdot t_2 \end{aligned}$$

Logic Circuit for control lines A, B and C


Inst-x
Inst-z
t1
A
Inst-x
Inst-y
t2
C
Inst-x
t0
Inst-y
t2
B