Subject موضوع الدرس

ES Concepts :-

Date التاريخ

Computing System → Processor → memory → I/O

(*) General porpuse لعمل حاجات عامة كتير

(*) specific Purpose بيستخدم لحاجة معينة

Constrains of ES [specific Purpose one]

→ Power → cost → time

الحاجات الي بيراعيها لما نعمل Embedded system

How to make an Embedded system?

(1) way (1) SB [system on Board]

(2) way (2) SOC [system on Chip]

دايما الـ SOC أوفق من SB من حيث الـ Constrains

طالما الـ Configability ومشكلتها أن الـ system

يبقى modified نقدر نعدل فيه حاجات

(*) IC: integrated Circuit → single chip has a specific Purpose (small)

(*) VLSI: IC but numbers of gates is bigger. has mills of transistors.

MP , MC , SOC , RAM , Rom → ICs

Subject موضوع الدرس Date التاريخ

Micro Processor unit

(*) MPU

→ processor : النوع القديم من الأنابيب

micro processor ← → MP : new processor → transistors

→ CPU :- Primary Processor (P) → primary, (S) secondry, (S)

(*) MCU → Computer system

S.P. ① MPU ② Memory ③ I/O

CPU [primary] GPU - DSP [secondary]

(*) GPU → Graphics خاص بالحاجات الي ليها علاقة بال

(*) DSP → خاص بالعمليات الرياضية المعقدة.

على عكس ال ALU ← عمليات حسابية بسيطة.

SOC $\xrightarrow{\text{عبارة عن}}$ MCU (but hish performance)

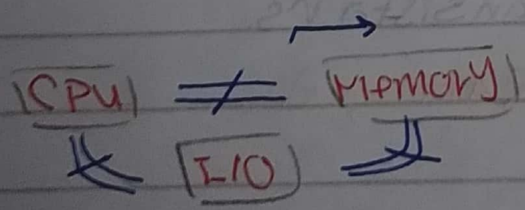
ايه هي انواع ال APP الي ممكن نعملها ?

① Bare metal SW → MCU   App → Drivers → HW

② OS APP   App → Kernal → Drivers → HW

↳ SOC → RPI

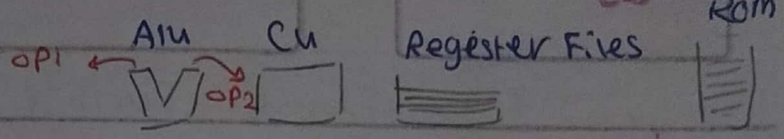
[Kit]
[[SOC + lids] + sensors acuator] → ECU



Bus set
↳ Data
↳ Add
↳ Control

Subject موضوع الدرس Date التاريخ

⊛ Processor :-



instructur_ life _ cycle [f_d_e]

← ال ROM بتحتوى على مجموعة من ال instructurs ف بيتعمل fetch ليهم لـ Reg_room عن طريق PC [Program Counter]

← ناجيبتهم من برا (ROM) لجوا (Reg-room) عشان نعرف نتعامل معاهم جوا . بتحطهم في Reg إسمه: instructure Regist

↳ insturcture decoder

← بعد كده عاوز أفهمهم باستخدام decode ال Cu is ID

← ليه بعمله decode ؟ عشان ابعت signal لـ Alu المفروض تعمله

← محتاج نعرف ايه عشان اعمله ① inst-set ② -format

العملية دى ايه ؟ add - sub ---- ← OP-Code

op-Code	op1	op2
000	111	101

الرقمين الى هعمل العملية عليهم ← op1, op2

← العملية الى بيعملها Alu ← excution والناتج بيتخزن في واحد من Reg - file

← عشان نفهم ال instur ← H.W [ID]

← S.W [Compile]

ISA :- instrur - set - Archit

Reduce ↳ RiSC :- بيفهم اوامر قليلة ← يحتاج S.W قوى

Complex ↳ CiSC بيفهم اوامر كثيرة ← يحتاج S.W ضعيف عادى

i/p ⟶ (logic gates) ⟶ o/p

فيه طريقتين لصناعة ال decoder [ID] → Hard wired [RiSC]

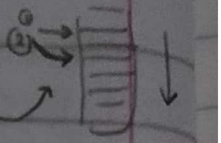
→ memory mapped عملية search جوا ال memory [CiSC]

→ They are the same

General Purpose Register

(*) Reg-file → GPR data Store [temp]

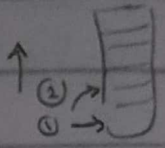
→ SP/FR (specific Purpose)



↳ ① PC → Program-Counter

↳ next inst بيشاور على

↳ ② SP → stack Pointer →

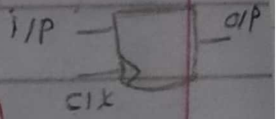


↳ ③ ACC → بيخزن الـ data بشكل

↳ ④ IR → inst-Reg مؤقت في Reg-f

↳ ⑤ PSW → process-status-word

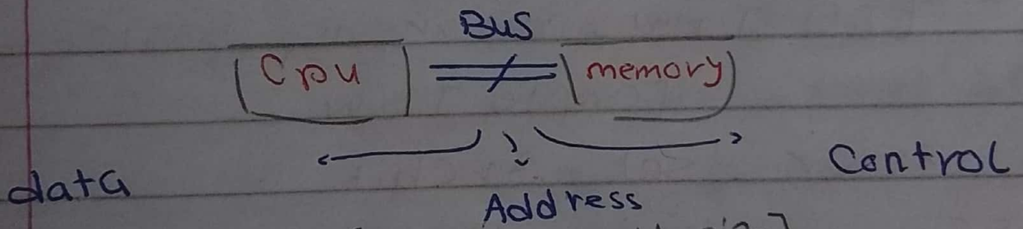
↳ flags → كل واحد منهم لوظيفة بتبقى مجموعة من الـ flags



(*) Memory :-

↳ Access time [write - Read]

→ Basic main element → Flip / Flop



[max memory size]

n (Address Line) → 2^n (Address location)

Read لو عاوز

① هشتغل على add. line الـ add اللي عاوزها

② هشتغل على control ← R⇒"0"

③ هستقبل من data اللي موجود في location

write لو عاوز أ

① هبعت الـ add على add-Buss

② هبعت على Control ← w ⇒ "1"

③ هبعت الـ data على data-Buss

(*) types of memory: ① volatil RAm (Rwm)
② non-volatile Rom ③ Hybrid

(*) RAm: ① SRAm → static
→ based on transistor
bit كل → تحتاج f/f → 6 Transistor تحتاج
→ faster than DRAm ← Advan
→ high cost ← dis ad

② DRAm → Dynamic
→ based on capacitor مشكلة بيفرغ بسرعة
الحل → Refresh Circuit → ليها الأولوية عن CPU
but it has simple H/W. وبالتالي ده بيخليها أبطأ

(*) Rom: based on FGM [floating gate mosfet]
→ Read only بالنسبة → (MP)
① → mask programble ROm → [OTP] one time prog.
② → PROm → programble Rom → [OTP] (user)
له هو الى بيحرق ال prog. ← بيحرق ال fuse يحوله من 1 ← 0
③ → EPROm → Erasble by (UV)

[RAm _ Rom]
(*) Hybrid :- E²PROm → Erasble by electricity
→ internal → External
Flash → Block Access [sector by sector]
→ faster Than E²PROm. ↗ reason
NVRAM → SRAm + Battary ,
SRAm + E²PROm + Battary