

CURS 3



BIU = Bus Interface Unit

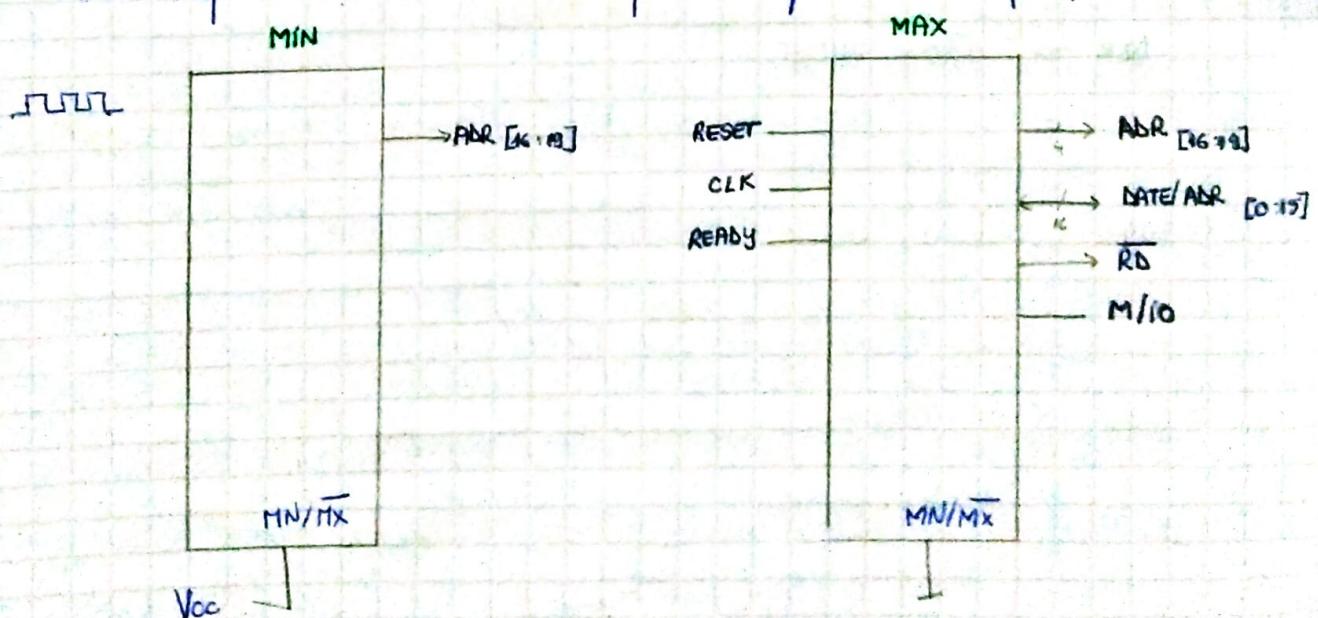
EU = Execution Unit

Componentele cu care comunică cu exteriorul:

- adrese
- date
- magistrală de comenzi prin care se transmite Memory Read/Writ
- stări pentru a verifica starea dispozitivului (e gata sau nu)

Acest procesor a avut 2 moduri de funcționare:

- MIN: în interiorul procesorului se scotesc toate resursele pentru comunicare cu exteriorul, însă doar pentru cele principale și nu suportă structura multiprocesor
- MAX: suportă și structura multiprocesor pentru a împărtășii muncă



- Trebuie să stiu modul în care funcționează \Rightarrow Vcc pentru MIN, masă pentru MAX
- Când avem o comandă, trebuie să stiu răspunsul (care nu e instantaneu) \Rightarrow semnalul READY ne permite să ne sincronizăm

8086 are 16 pini și poate adresa maxim 1MB de memorie \Rightarrow sunt necesare 20 de biti \Rightarrow Magistrala de date/adrese de 16 biti + încă 4 biti de adrese

- Trebuie să stiu când vrem să citim \Rightarrow avem nevoie de un semnal de read, iar pentru a determina de unde citim ne uităm pe magistrala de adresa.

Citire \rightarrow din memorie \Rightarrow M/I_O
 \downarrow din sistemul intrare/iesire

- Avem nevoie de un sistem de întreruperi \Rightarrow semnalul INT.

Întreruperea presupune următorul lucru

Vine cîndva din exterior și spune că e întrerupere de nivel ... Pe baza acelei adrese, se caută driver-ul corespondent (se face un call) și se execută întreruperea.

INTRERUPERI MASCABILE \rightarrow atunci când procesorul nu își dorește să fie întrerupt (DISABLE INTERACT)

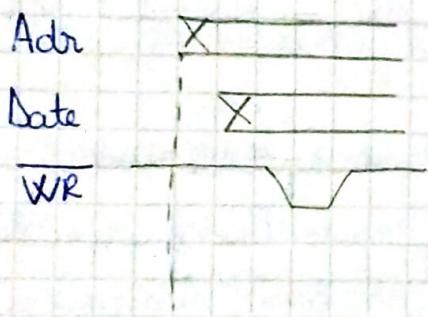
Există și ENABLE INTERACT în care putem preciza doar anumite întreruperi care sunt permise

INTRERUPERI NEMASCABILE \rightarrow întreruperi ce nu pot fi refuzate

Pentru a diferenția întrerupările mascabile/nemascabile se folosește semnalul NMI. Atunci când există o întrerupere nemascabilă, procesorul trebuie să o trateze indiferent dacă urez sau nu.

Întreruperile nemarcabile sunt necesare pentru situații de urgență (ex: căderea tensiunii de alimentare \Rightarrow sarcina de alimentare va fi preluată de UPS; dacă căderea durează mai mult decât poate dura UPS-ul, pică tot \Rightarrow se pierd toate datele) \Rightarrow se generează o întrerupere de către UPS, iar procesorul trimite semnalul pentru salvarea tuturor datelor (se comportă ca și shutdown)

- Date enable \Rightarrow semnal care ne spune că datele sunt stabilite pe magistrala de date și pot fi luate. Dacă luăm datele înainte de date enable, se vor lua datele care se găsesc pe magistrală atunci (greșit)
- Date transmitter / receiver \rightarrow trebuie să îl suntem pe cel din exterior că urcă să îi transmit / primim date



- Întreruperile vin între construcțiuni (dacă execut și adunare și primesc și întrerupere, termin de efectuat adunarea și doar după procesez întreruperea) \Rightarrow semnalul interrupt Acknowledge prin care procesorul spune că poate fi întrerupt atunci
- Semnalul ALE spune că pe magistrala AD sunt adrese și ele pot fi luate, deoarece sunt stabilite. (ALE = Address Latch Enable)
- Dacă DMA și procesorul vor să acceseze în același timp memoria, se folosește un semnal HOLD și HOLDA pentru a întârzi procesul.

- Semnalul Test → să rugă de cinere din exterior să cștepe
 - Semnalele negate din desen sunt active pe 0, iar celelalte active pe 1.
- Motiv:

Când vrem să dăm o comandă între 2 plăci, este bine să fie dată comanda pe 0, deoarece dacă se face o întrerupere transzistorul ar avea 1, deoarece nu mai circula curent.

⇒ Comanda pe 0 înseamnă că modulele există, legăturile între ele există și totul este în regulă

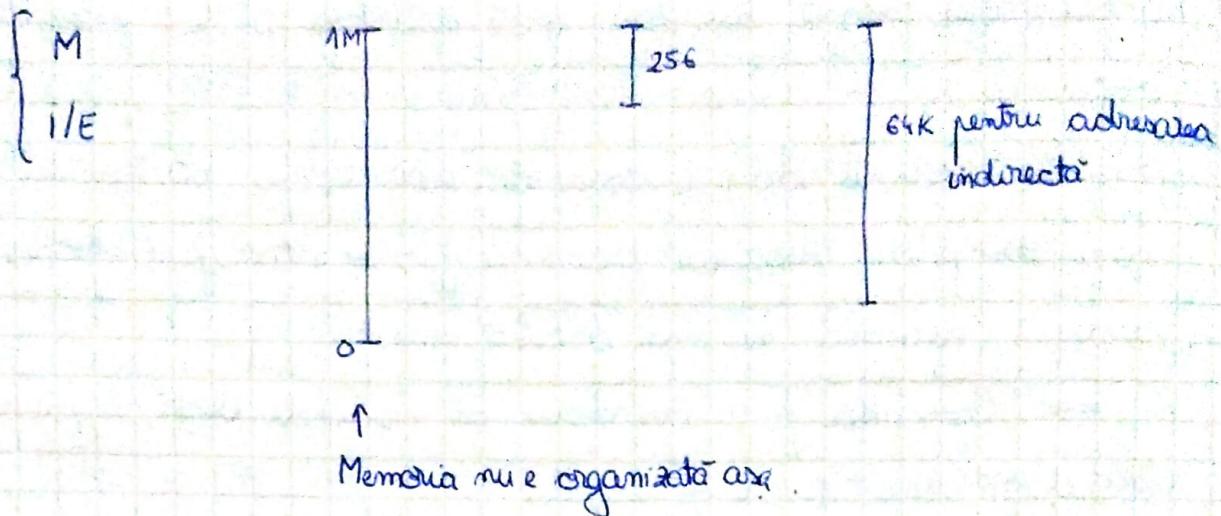
MAX

- $S_2, S_1, S_0 \rightarrow$ ne codifică ciclul maximii curente (cuprinde toate celelalte semnale din MIN din dreapta)
- Trebuie să putem implementa un remafor pentru cazul în care mai multe procesează vor accesa la același zonă de memorie
⇒ semnalul lock
- Sincronizarea la nivel de stocare cu un coprocesor matematic sau un coprocesor de intrare/iesire ⇒ semnalul RQ/GT 01

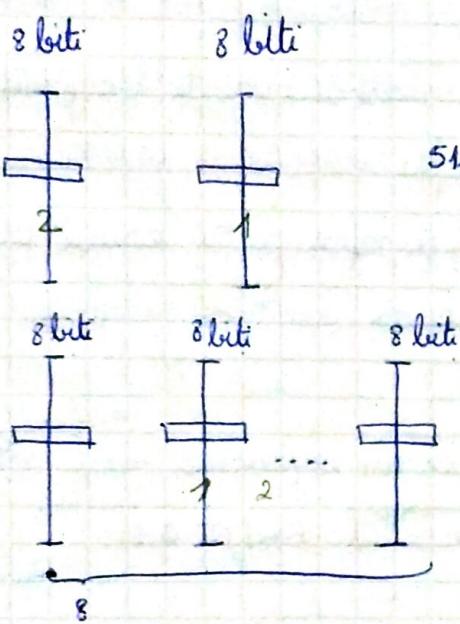
Diferența dintre unitate și coprocesor

- unitate → se comportă în mod slave (iți dau eu tot ce îi trebuie și el face)
- coprocesor → are în interior o unitate de comandă similară cu a procesorului și se comportă prin la prin cu acesta; între acesta și procesor există doar mesaje de tipul „Așteaptă-mă să fac eu”
- QS → semnal prin care se anunță starea stivei

POZIȚII



ORGANIZAREA MEMORIEI



512 curante

1	2
---	---

↳ Se fac 2 citiri

↓
Functionează de 2 ori mai lent.

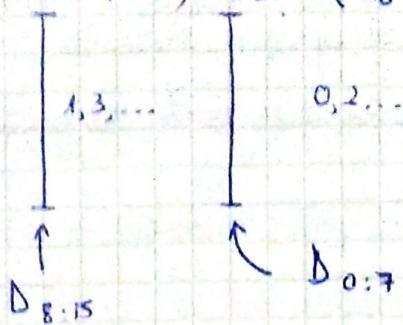
⇒ Ca să nu se mai facă 2 citiri, putem alinia datele începând cu o adresă multiplu de 8, astfel încât ar fi necesară doar o singură citire.

Atunci când vrem să facem mov AL, ... trebuie să îi spunem procesorului de unde să ia cei 8 biti (primul sau cel de-al doilea bloc)
⇒ avem nevoie de nemulajul BHE.

Dacă facem mov AH, ... ne vor lua ambele blocuri (16 biti)

A ₁₅ ...	A ₁ , A ₀	Adresă
0 ...	0 0	0
0 ...	0 1	1
0 ...	1 0	2
		:

8 biti ($A_0 = 1$) 8 biti ($A_0 = 0$)



Când lucrăm la nivel de curviște, se folosește BHE (Byte High Enable).
Niciu nu dăm următoarele valori:

BHE	A_0			
0	0	16	$D_{8:15}$	$D_{0:7}$
0	1	8	$D_{8:15}$	-
1	0	8	-	$D_{0:7}$
1	1	-	-	-

Folosim cînd vom
proiecta memoria

Memoria va avea niște celule rezervate:

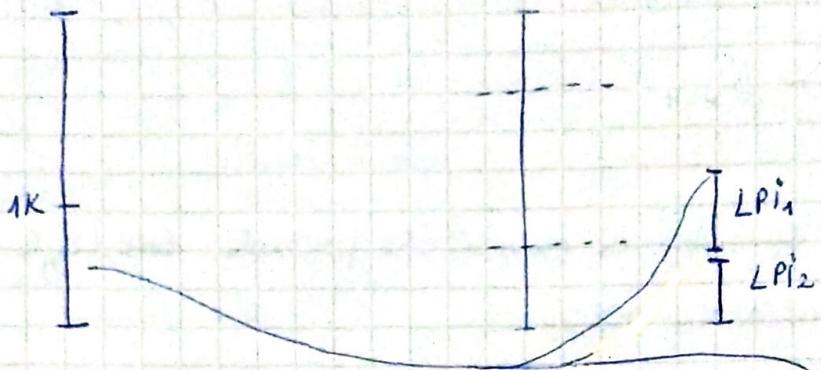
- primul K de memorie \Rightarrow rezervat pentru SO
- ultimele 16 celule \Rightarrow rezervate pentru Bios



Reset ne duce la adresa FFF0, unde trebuie să găsim obligatoriu
memorie permanentă

- initial procesorul începe cu un disable interrupt pentru a se aștepta să fie pregătit totul (analogie: când merge la muncă, trebuie întîi să scoți toate dosarele și apoi primești clientii. Nu primești clienti până nu e pregătit mediul de lucru)

Dacă avem 2 imprimante, sistemul de operare încarcă ambele driveri.
 În cazul în care primul o întârziere, nu știm unde ne ducem.

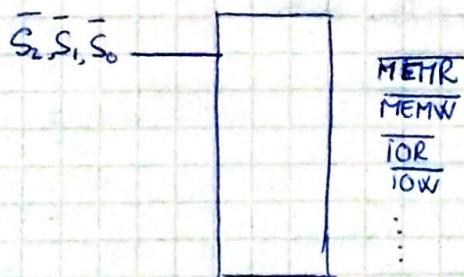


Atunci când dău o întârziere, SO se uită în memorie la ce imprimantă e default și merge la driverul corespunzător \Rightarrow adresare indirectă.

SISTEMUL DE INTRARE / IEȘIRE

Porturile cu adresa pară $\Rightarrow D_{0:7}$
 Porturile cu adresa impară $\Rightarrow D_{8:15}$

Dacă sunt puse invier, vom citi și vom scrie același.



$\overline{S_2} \ \overline{S_1} \ \overline{S_0}$

0 0 0 interrupt acknowledge

0 0 1 citire port I/O

0 1 0 scriere port I/O

0 1 1 interpretează instrucțiunea HALT

1 0 0 citire instrucțiune (faza de fetch)

1 0 1 citire din memorie

1 1 0 scriere în memorie

1 1 1 - (folosit pt. a verifica că s-a terminat ciclul curent sau pt.

a lăsa intreruperea să intre)

Procesarele citire, interpretare și execuție instrucțiuni în limbaj masină

CICLU INSTRUCȚIUNE = citire + interpretare + execuție

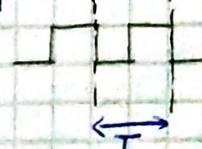
Acesta conține mai multe instrucțiuni



CICLU INSTRUCȚIUNE = \sum ciclui MASINĂ

legătura cu exteriorul

Ciclu masină = $\sum T_j$, unde T_j = stare \Rightarrow o perioadă a ciclului de cca



În stare T_1 are loc operația de început de ciclu, adică vom comunica adrese și stări pe magistrala de AD.

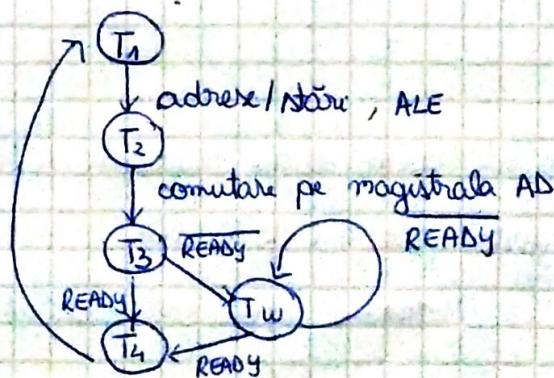
Trimitem ALE după ce sunt gata și trecem în T_2 .

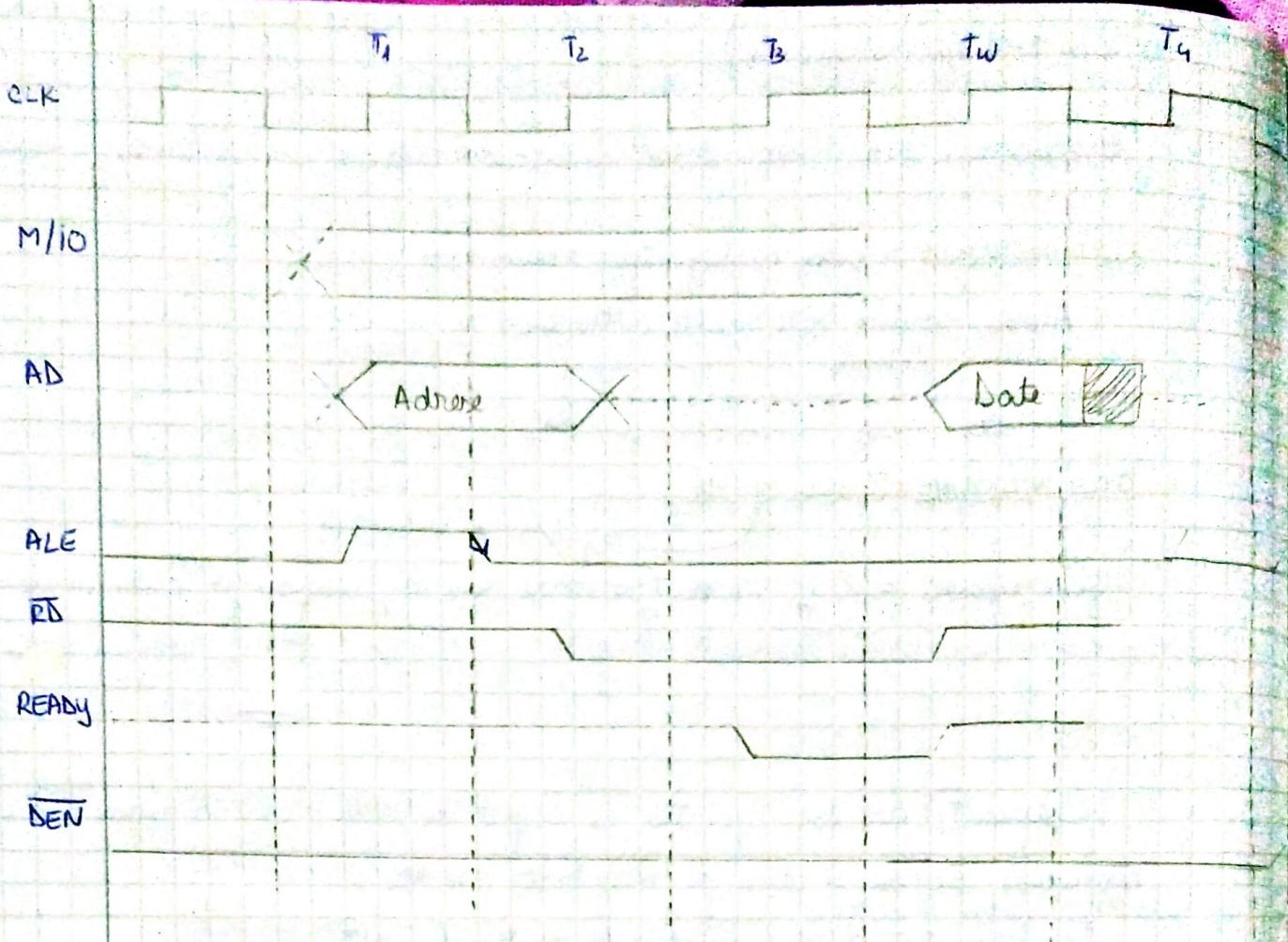
Starea T_2 : comutare de magistrală adrese/date și se trece în T_3
necesară, decarece se folosesc în comun aceleași magistrale

Starea T_3 : Ar trebui să iau datele, dar nu știm dacă cel din exterior e mai rapid sau nu \Rightarrow ne uităm la semnalul READY.

Dacă READY = 1 (cel din afară a răspuns la semnal în timp util) \Rightarrow am terminat ciclul

Dacă READY = 0 , se intră într-o stare de așteptare (echivalent cu o prelungire a ciclului stării T_3) \Rightarrow Se introduc "noi stări" care depind de relația cu exteriorul.





La apariția lui RESET:

$$\left. \begin{array}{l} IP = 0 \\ CS = FFFF \\ DS = 0 \\ ES = 0 \\ SS = 0 \end{array} \right\} 16CS + IP = FFFF0 \rightarrow \text{incepe să citească prima instrucțiune care va fi cea din BIOS}$$

Se lucrează cu segmente de 64K, decarece CS are 16 biti?

$$\begin{array}{r} \text{CS} \quad \boxed{} \boxed{} \boxed{} \boxed{0} \\ + \\ \text{IP} \quad \boxed{} \boxed{} \boxed{1} \end{array}$$

Adresă $\boxed{} \boxed{} \boxed{1} \boxed{1}$

De ce s-a făcut CS pe 16 biti când erau suficienți 4 biti?

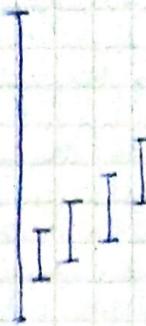
- pentru a nu exista registrii de 4× de 16 biti
- putem pune segmentele în 16 moduri unele peste celelalte dacă folosim

4 biti sau în 4096 de căi folosim 16 biti.

Exemplu:



cu 4 biti



cu 16 biti