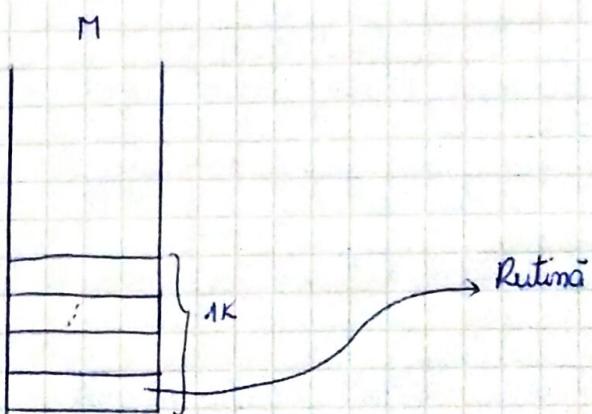


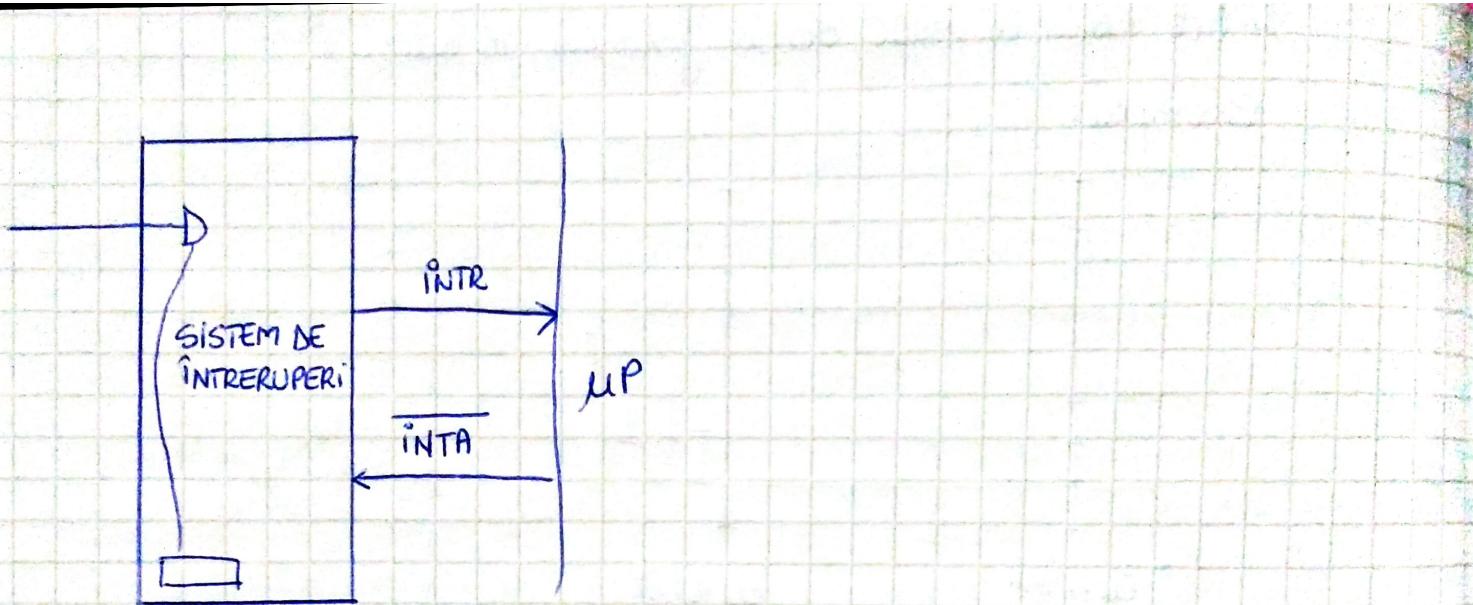
14.03.2018

CURS 4

Microprocesorul poate lucra cu intreruperi
intreruperi → software (instructiunea intr)
 → hardware [INTR] { poni ai microprocesorului
 { NMI



Intrerupere hardware INTR se numeste mascalibă decarece ne foloseste adresarea indirectă . INTR poate fi activată cu ENABLE INTERACT sau poate permite "doar anumite instrucțiuni să îl întrețină".



Se folosește un și logic pentru a face mascare selectivă (doar unele interruperi cu voci să treacă mai departe)

! O interrupere este un cîșc marină.

La TI se generează mereu ALE.

(procesorul îmi furnizează la exterior adresa)

Se intră în ciclul maxim de înterrupere după instrucțiunea curentă dacă ~~nu este~~ aceasta are voie să întreagă.

⇒ Se va da interrupt acknowledge undeva.

După INTA, durează puțin până primește adresa ⇒ nu pot primi în același ciclu maxim, ci în următorul.

Este obligatoriu ca după începerea întârzierii aceasta să nu fie întârziată de altă instrucțiune (este posibil ca întârzierea să folosească adresa generată pentru rutină).

lock → 2 cicluri maximi sunt indivizibili

O instrucțiune se ură la LOCK înainte de a începe. Dacă e 1, atunci are voie să se execute, altfel nu.

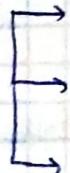
Pentru AD trebuie să fie adresa celulei capacă asociată întârzierii
vector de întârziere

MCE → echivalentul lui ALE

va fi folosit de sistemul de întârziere pentru a reține
organizat pe mai multe nivele

• Porte logice (AND, OR, NAND, etc.)

• Registruri



• Decodificare / Multiplexare

• Tri-state 74286, 70...

• Chip-uri 8252,

Există 8 întârzieri în sistemul de întârzieri.

TEST

Este un remanent extern asociat lui WAIT.

Dacă e activ, procesorul trece într-o stare de așteptare. Se stă în acea stare până când este dezactivat remanentul TEST.

Instrucțiunea WAIT nu execută repetitiv până când este valoarea lui TEST este 0.

Pe durata wait-ului pot apărea 2 evenimente:

1. întârziere
2. cerere HOLD

Dacă apare o întârziere, nu execută următoarea recurentă:

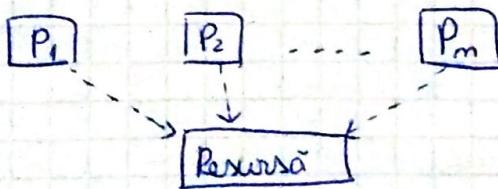
- Microprocesorul citește încă o dată instrucțiunea WAIT (nu îl-a incunyat îP)
- Se face legătura cu rutina de întârziere, nu execută întârzierea, nu citește din nou WAIT și nu verifică valoarea lui TEST

Dacă apare o cerere HOLD:

Se permite un ciclu de acces la memorie după care se reia stare de așteptare

Vrem să folosim procesorul în mod maxim pentru a ne permite să lucrăm cu structuri microprocesor (modul minim nu permite acest lucru).

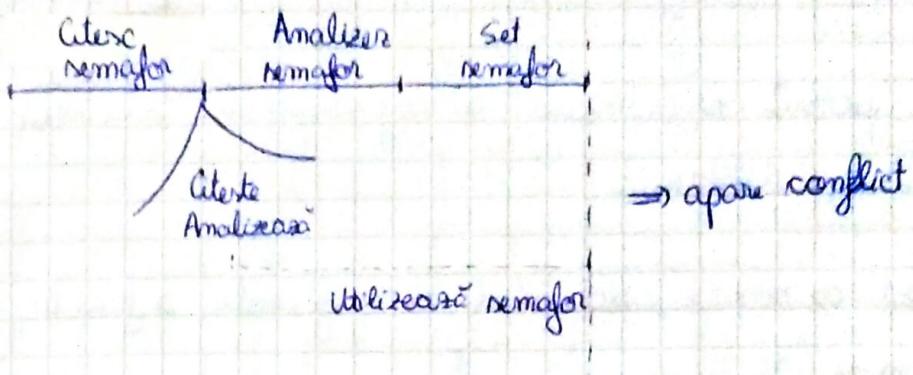
Există astfel situația în care fiecare procesor vrea să acceseze o resursă înăuntru aceasta este urmă:



Folosim un semafor pentru a ne asigura că niciun alt procesor nu folosește resursa

0 → liber

1 → ocupat



→ Ciclii trăznii să fie indivizibili !

{ Test and Set

RMW = Read Modify Write

ASTEPT:

MOV AL, 1

LOCK

MOV SEMAFOR, 0

UNLOCK

↑ pregătesc valoare pt a fi pusă în semafor

LOCK

↑ pentru a se întări că următoarele 2 instrucțiuni sunt indivizibile

XCHG AL, SEMAFOR

TEST AL, AL

JNZ ASTEPT

:

utilizare resursă

nu poate să intre în meniu între aceste operații
(pentru că sunt indivizibile)

mai rapid decât testarea cu 0 sau 1, deoarece
această metodă necesită un singur ciclu

Înstructiunea HLT trage procesorul într-o stare idle (low-power)

Poate ieși din această stare în 2 moduri:

1. RESET \Rightarrow ne trage la prima instrucțiune din BIOS

2. întrerupere \Rightarrow ne trage la rutina de tratare și apoi în idle din nou

Rg011 permit sincronizarea cu coprocesor. Schimbul de magistrală ne face la nivel de stare.

La nivel de resurse, schimbul de magistrală ne face la nivel de ciclu maxim.

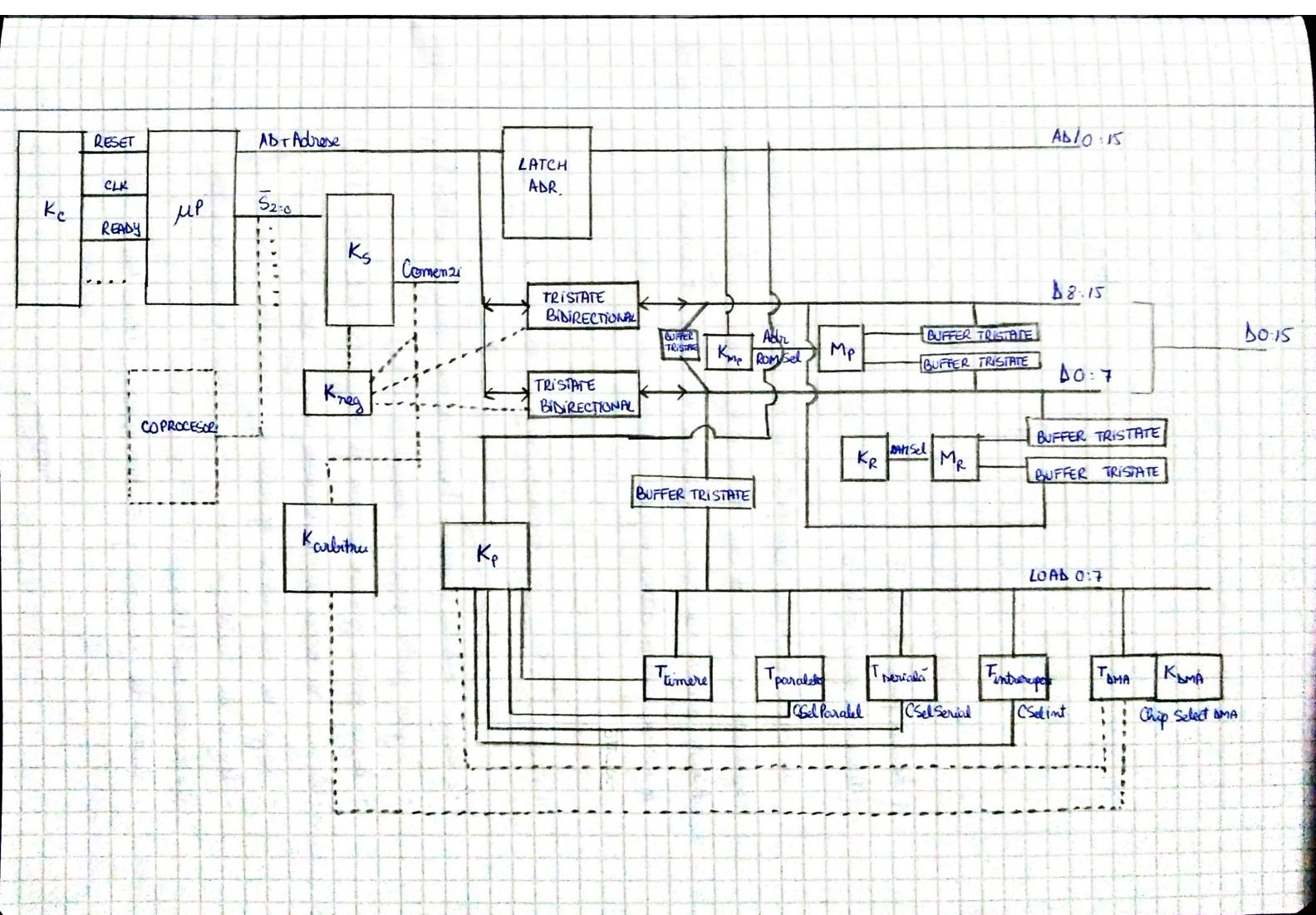
MOTHER BOARD

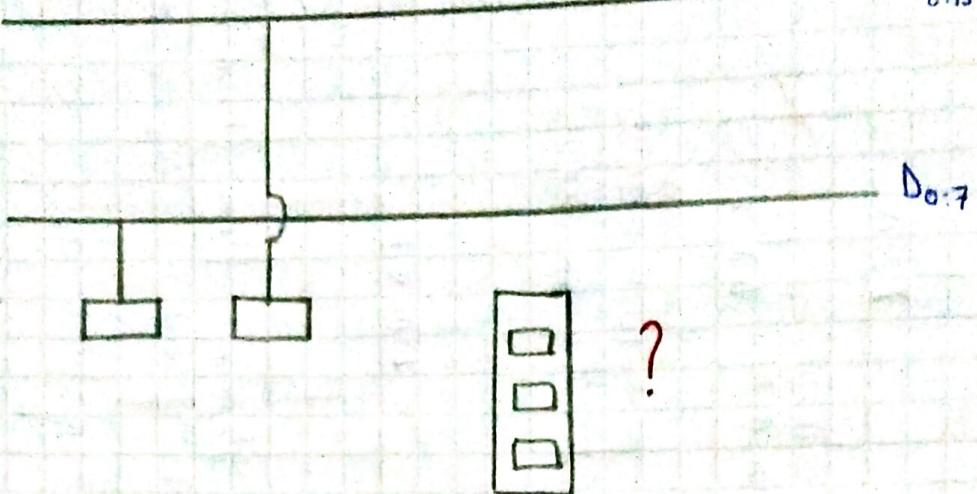
SPECIFICAȚII DE PROIECTARE

1. Microprocesor x86, cu o posibilă structură multiprocesor
2. Memorie RAM $\sim 512K$
 - \rightarrow statică
 - \downarrow dinamică
3. Memorie permanentă $\sim 64K$ Bios
4. Timere
5. Interfață Keyboard
6. Interfață paralelă
7. Interfață serială
8. Sistem de întreruperi
9. DMA
10. Procesor intrare-iesire

FAN-OUT = capacitatea de a arăgura un curent de resurse (~ 10 iesiri de obicei)

FAN-IN = cat consumă fiecare circuit





{ Adresă pară $\Rightarrow D_0:7$

Adresă impară $\Rightarrow D_8:15$

În cazul chip-ului am putea să asociem adrese pară/impare registrilor continuti de chip și să îl legăm la magistrala corespunzătoare. Această metodă este greșită, decocce IBM a asociat adresele 4,5,6, deci suntem să adrese pară și adrese impare.

memorie, periferice etc.

Pe lângă unitățile primare, trebuie să mai adăugăm unități logice de comandă:

- logica pentru generare semnal ceas (K_c)
- logica pentru generare comenzi pe magistrala sistemului (K_s)
- logica de acces la magistrala de adrese și date (K_{Mag Ad})
- logica de interpretarea spațiului de adrese intrări-iesiri (K_{perif})
- logica de arbitrage între procesor și DMA pt. excluderea mutuală a accesului la memorie (K_{arbitru})
- unitatea de comandă pentru RAM și ROM (K_{MP}, K_{MR})