Calculatorul von Neumann + baza arhitecturii 80x86

CURS

Cuprins curs

Structura stratificata a unui calculator

Modelul von Neumann

Criteriile von Neumann

Sisteme von Neumann imbunatatite

Sisteme non-von Neumann

Structura fizica a unui sistem cu microprocesor

Ciclu masina

Unitatea centrala de procesare (UCP)

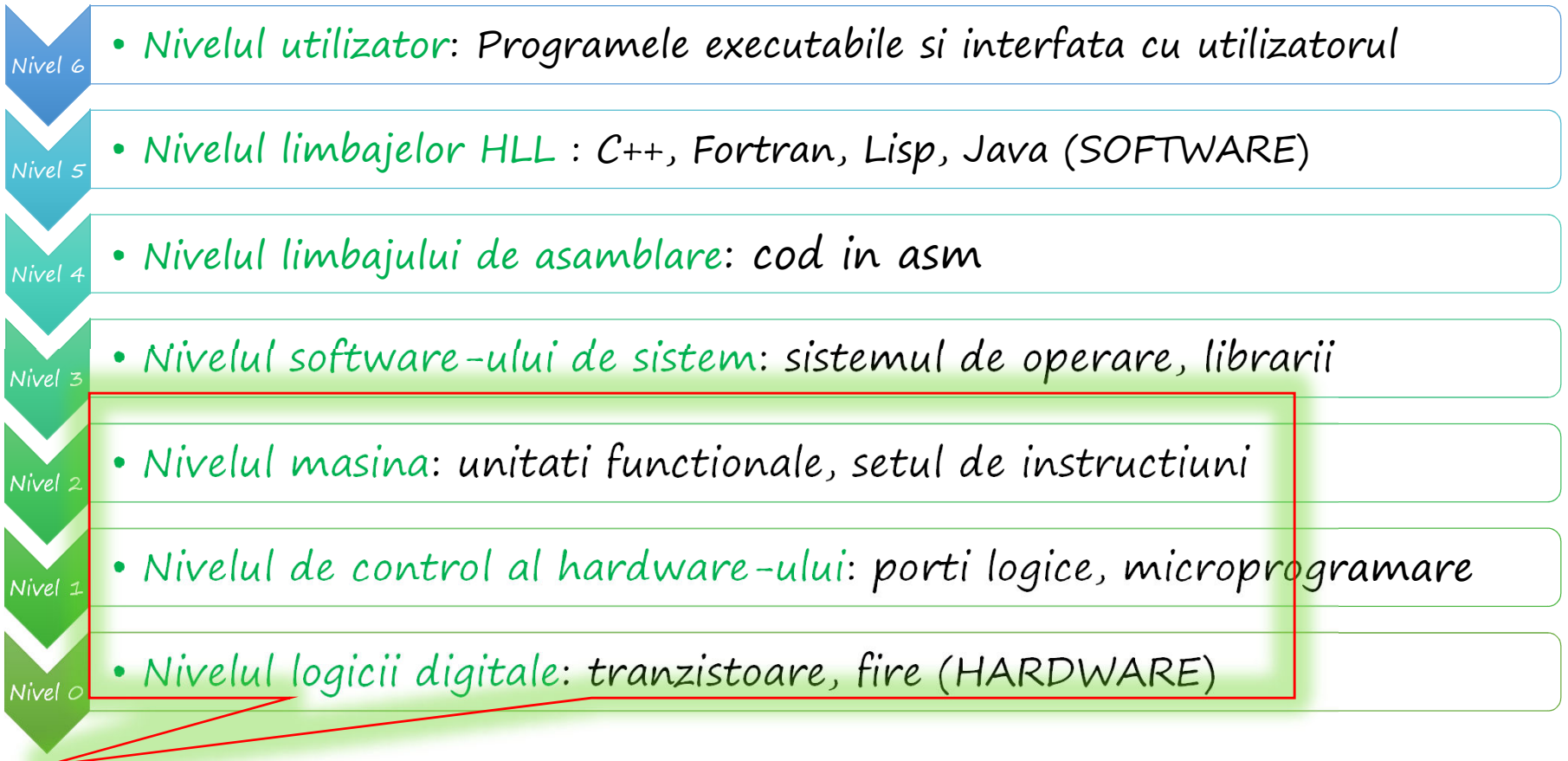
Memoria

Dispozitivele de intrare/iesire

Busul sistem

Ierarhia de nivele

In domeniul arhitecturii calculatoarelor: conceptul de NIVELE (masini virtuale, o abstractizare a sistemului)
– exista 7 posibile nivele/vederi separate/independente la care poate fi considerat un sistem de calcul :
Nivelul cel mai ridicat = "utilizator" (utilizatorul ruleaza programe => foloseste sistemul)
Nivelul cel mai scazut = "hardware" (al masinii fizice = tranzistoare si fire)



ISA – Instruction Set Architecture

Structura stratificata a unui calculator permite:

- mai multe nivele de abstractizare
- mai multe forme de acces la resursele unui calculator

Necesitatea stratificarii/ierarhizarii:

- programarea devine mai simpla, mai eficienta
- utilizatori de diferite categorii
- scaderea complexitatii prin descompunere functionala

La primele sisteme de calcul (contemporane cu ENIAC):

programarea se realiza la nivelul logicii digitale,

implicand conectarea de fire la prize

- nu exista arhitectura pe nivele, ierarhizata
- cu fiecare noua problema de rezolvat, sistemul trebuia din nou modificat

=> o noua configuratie ***hardware***

J.W. Mauchly si J.P. Eckert au propus
(inainte de a termina **ENIAC**)
o noua metoda pt a schimba comportamentul masinii lor de calcul

Ideea -> propusa (top secret in timpul razboiului) ca fundament pt noul lor proiect **EDVAC**

Matematicianul John von Neumann (*noy-mann*) a fost cel care a publicat pt prima data ideea !

-> computerele cu program memorat = "*von Neumann* systems",
"*von Neumann* architecture"

Modelul de bază pentru arhitectura unui sistem de calcul (SC) cu program memorat
a fost introdus in anii 1944-1945 de John von Neumann.

Pana nu demult, majoritatea calculatoarelor respectau criteriile din « ***modelul von Neumann*** »

Criteriile von Neumann

Computerele cu program memorat au primit în timp numele de „***sisteme von Neumann***”, iar arhitectura respectivă s-a numit „*arhitectură von Neumann*”.

Acest model de bază pentru **arhitectura unui SC cu program memorat** a fost introdus în anii 1944-1945 și până nu demult, majoritatea computerelor au respectat criteriile din „modelul von Neumann”.

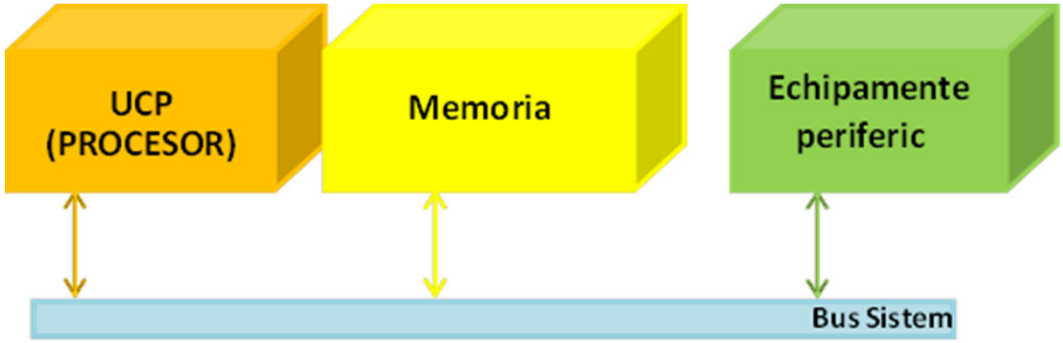
Cele ***5 caracteristici principale ale calculatorului cu program memorat***, criterii enunţate de von Neumann, sunt:

- 1. **intrare (I)** - prin intermediul căreia să se poată introduce un număr nelimitat de operanzi şi instrucţiuni în SC;
- 2. o **memorie (MEM)** - din care să se citească ***instrucţiunile*** şi ***operanzii*** şi în care să se poată memora rezultatele obţinute;
- 3. o **unitate de calcul (UAL)** - pentru a efectua operaţii aritmetice şi logice asupra operanzilor din memorie;
- 4. **ieşire (O)** - prin intermediul căreia un număr nelimitat de rezultate să poată fi redate în afara sistemului (utilizatorul să poată avea acces la ele);
- 5. o **unitate de comandă (UC)** - capabilă să interpreteze instrucţiunile preluate/ obţinute din memorie şi să selecteze diferite moduri de desfăşurare a activităţii viitoare a SC pe baza rezultatelor calculelor.

=> Un calculator (Sist de Calcul) *cu program memorat* trebuie să posede **3 *Componente hardware principale*: *UCP* (în care se regăsesc *UAL* și *UC*), *MEM* și *I/O* (periferice)** .

- **Ideea principala: atat instructiunile cat si datele sunt pastrate in aceeasi memorie**

Un calculator (SC) *cu program memorat* trebuie să posede **3 *Componente hardware principale***



(1) **UCP** (in gen. = procesor)
– asigura prelucrarea datelor si controleaza functionarea SC

(2) **Unitate de memorie** (interna /principala/ de baza)
din care se pot citi instrucţiuni şi operanzi (date) şi în care se depun rezultate;

(3) **Unitatea de intrare/iesire** (I/O)
– transferul datelor intre calculator si ext.

IDEEA DE BAZA

Cele 3 componente – legate printr-un **Bus** ce transporta informatia

Bus (magistrala) = un grup de semnale electrice sau fire folosite pentru a transfera informatie

Calculatorul von Neumann

Criteriile von Neumann

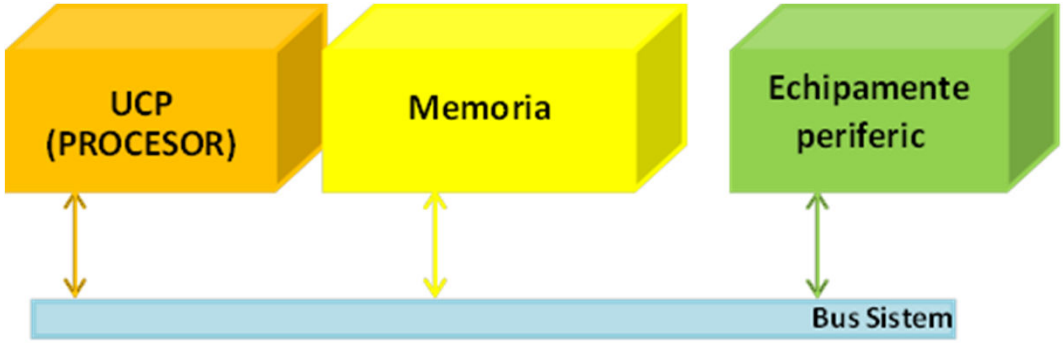
- **Ideea principala: atat instructiunile cat si datele sunt pastrate in aceeasi memorie**

Un calculator (SC) *cu program memorat* trebuie să posede **3 *Componente hardware principale***

(1) **UCP** (in gen. = procesor)
– asigura prelucrarea datelor si controleaza functionarea SC

(2) **Unitate de memorie** (interna /principala/ de baza) din care se pot citi instrucţiuni şi operanzi (date) şi în care se depun rezultate;

(3) **Unitatea de intrare/iesire** (I/O)
– transferul datelor intre calculator si ext.



Ideea principala von Neumann:

-atat instructiunile cat si datele sunt pastrate in aceeasi memorie

=> realizarea instructiunilor prin *procesare secventiala :*

- contine **o singura cale de date** (fizica sau logica) intre UCP si memoria principala, fortand *alternarea* de date (instructiuni) sau comenzi (de control)

="*von Neumann* **bottleneck**"

Cele 3 componente – legate printr-un **Bus** ce transporta informatia

Bus (magistrala) = un grup de semnale electrice sau fire folosite pentru a transfera informatie

Viteza busului afecteaza major performanta intregului SC, la fel ca viteza UCP sau dimensiunea memoriei

Imbunatatiri aduse (in timp) modelului conventional von Neumann:

2) A aparut principiul ierarhiei de memorie

Programele si datele continute in medii de stocare cu viteza scazuta la accesare (ex: HDD) pot fi *copiate intr-o memorie volatila*, rapid de accesat (precum RAM) inainte de a fi executate

S-a adaugat memorie cache si virtuala

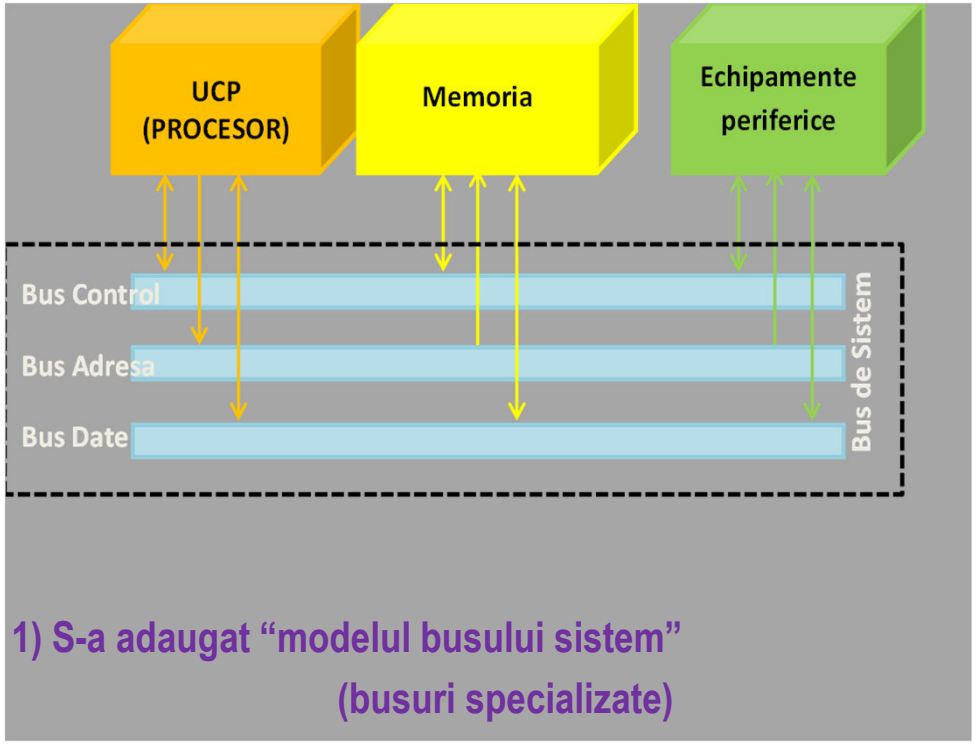
3) Folosirea registrilor index pt adresare, iar de la 386 în sus și a regiștrilor generali

4) Adaugarea de unitati (si nu cipuri) pt virgula mobila - s-a adăugat intern un **coprocesor matematic**, specializat pe realizarea de operaţii în virgulă mobilă de la 486 în sus; aceste *cipuri* (până la 487) sau *unităţi separate* (după 487) au fost denumite ca aparţinând seriei sau liniei **x87**;

5) Folosirea intreruperilor si a operatiilor I/O asincrone

6) S-au adaugat registre generale

- DAR prin **Adaugarea de procesoare** ???
- Ce s-a obtinut?



BD – muta datele din mem principala in registrele UCP (si invers)

BA – pastreaza adresa datelor pe care BD le acceseaza la mom curent

BC – vehiculeaza semnalele de control ce specifica cum va avea loc transferul informatiei (înspre CPU/ dinspre CPU), cu port/ cu mem, etc

- La sf. anilor 19**60** – computere de performanta ridicata – procesoare duale pt cresterea performantelor
- In 19**70** supercomputere cu 32 procesoare.
- in anii 19**80** - Supercomputere cu 1,000 procesoare
- In 19**99**, IBM anunta sistemul **Blue Gene** - peste 1 million procesoare.

- Sisteme din categoria non-von Neumann: neural networks, genetic algorithms, quantum computation, dataflow computation, parallel computing – cea mai populara.
- Computere DNA, cuantice, sisteme cu prelucrare de fluxuri de informatie, sisteme vectoriale, etc.

- Ce au acestea in comun – procesarea este **DISTRIBUITA !!!**
- intre mai multe unitati de procesare ce **LUCREAZA IN PARALEL**

- Procesarea Paralela e doar o metoda de furnizare a puterii computationale in crestere.

- Neclar: Unde vor duce toate aceste tehnologii ?
- Care va fi / unde se va delimita urmatoarea generatie de calculatoare ?

Calculatorul von Neumann - *Arhitectura uniprocesor*

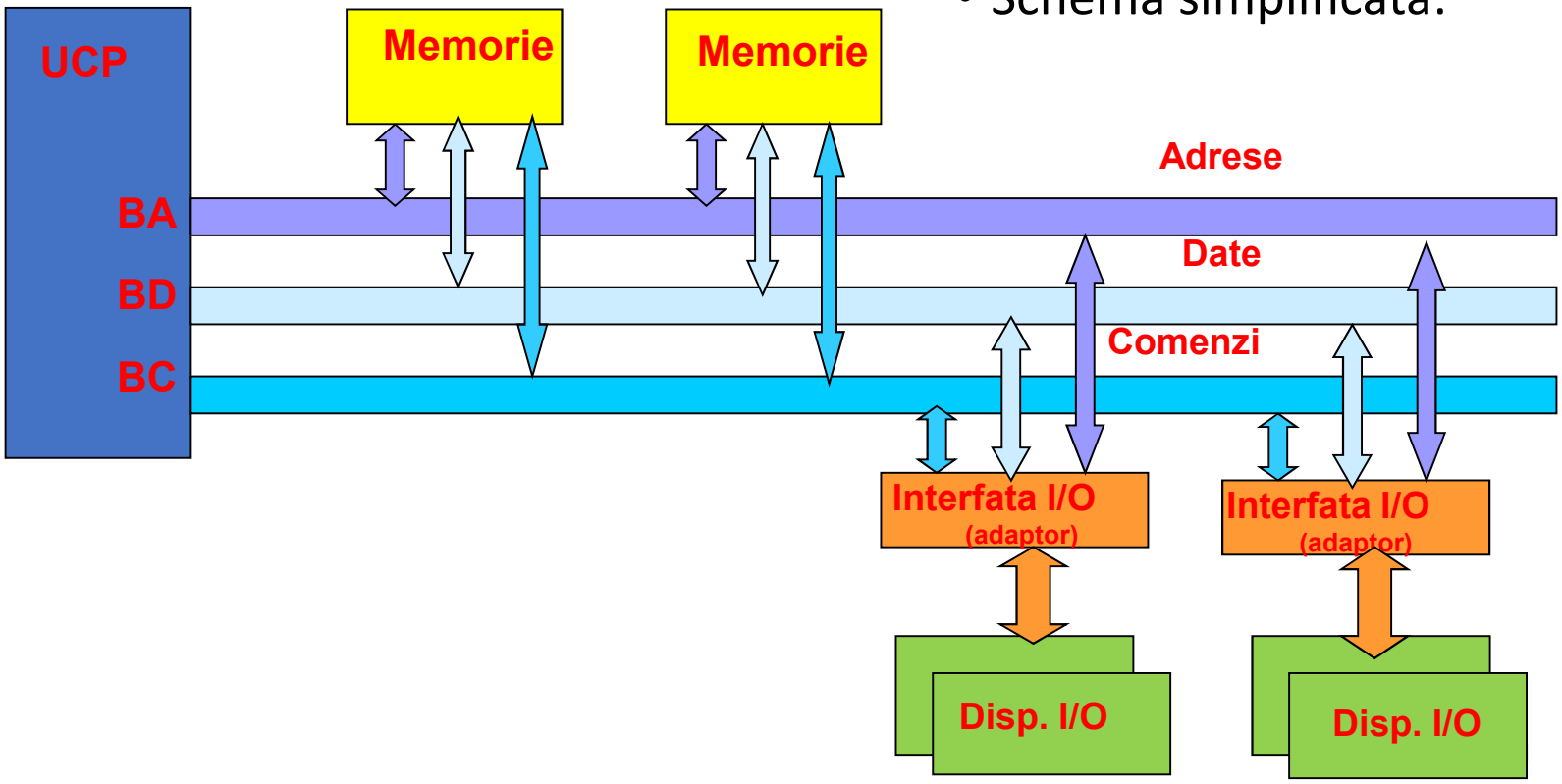
Structura fizica a unui sistem cu microprocesor

Microcalculatoarele tipice folosesc (dupa ***modelul von Neumann***)

- un generator de **ceas (clock)**
- o unitate centrală de prelucrare UCP (CPU – Central Processing Unit), sau simplu **PROCESOR**
- interfeţe cu **memoria** şi cu **dispozitivele externe de intrare/ieşire** (**adaptoare de interfata**).

Unităţile sunt interconectate prin **busuri (magistrale) specializate** care transferă informaţiile între acestea

- Schema simplificata:



Structura fizica a unui sistem cu microprocesor (2)

3.Decodificatorul de memorie (DECM):
la OUT: semnale de selecţie pt circ. de memorie,
la IN: linii din MA

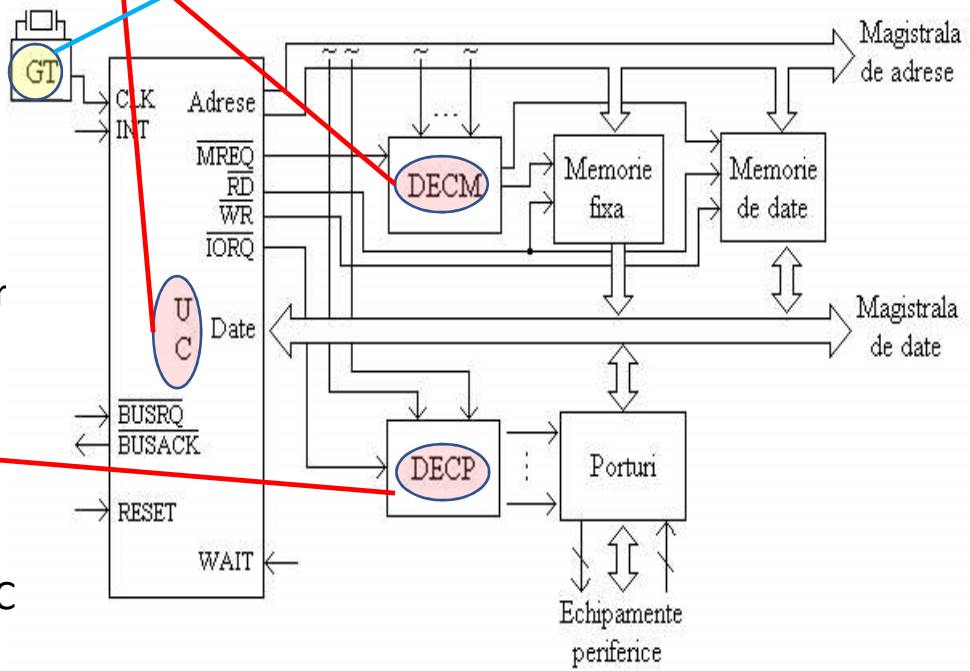
5. Memoria fixă
Implem. cu circ ROM, OTP, EPROM, EEPROM sau Flash, memorează programe de sistem sau aplicatii
- rutine de bază pt comunicarea UC cu perifericele, programe de test, şi un program încărcător (va încărca S.O. de pe suport extern în memoria RAM şi îl va lansa în execuţie).

6. Memoria de scriere/citire (RAM)
pt memorări temporare (si a S.O. a unei sesiuni de lucr
SRAM (RAM static) şi DRAM (RAM dinamic).

4. Decodificatorul de porturi (DECP)
La OUT: semnale selecţie porturi, la IN: linii din MA

7. Porturi de intrare/ieşire: asigură interfaţa dintre UC şi echipamentele de I/O, converteşte informaţia din formatul UC în cel al perifericelor şi invers.

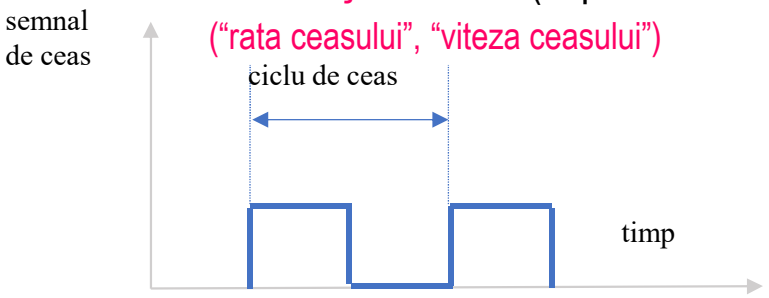
Schema bloc generala a unui microsistem digital
1. Unitatea centrală de procesare (UC) = microprocesor
+ alte circuite (2.**generator de tact**, generator al semnalului de iniţializare, amplificarea+demultiplexarea magistralelor)



8. Magistrale externe de adrese, date, comandă şi control – interconecteaza toate aceste componente

Procesorul actioneaza ca un controler al tuturor actiunilor/serviciilor furnizate de SC
- actiunile lui sunt sincronizate cu un semnal de ceas

SC fol. un **semnal de ceas (format din ciclii de ceas)** pt a raporta şi sincroniza execuţia diferitelor operaţii
- este def. de durata (perioada) ciclului de ceas (exprim în *secunde*) = timpul necesar trecerii unui ciclu de ceas
sau
de frecvenţa ceasului (exprimată în *Hz=1ciclu/secunda*) = inversul perioadei de ceas.

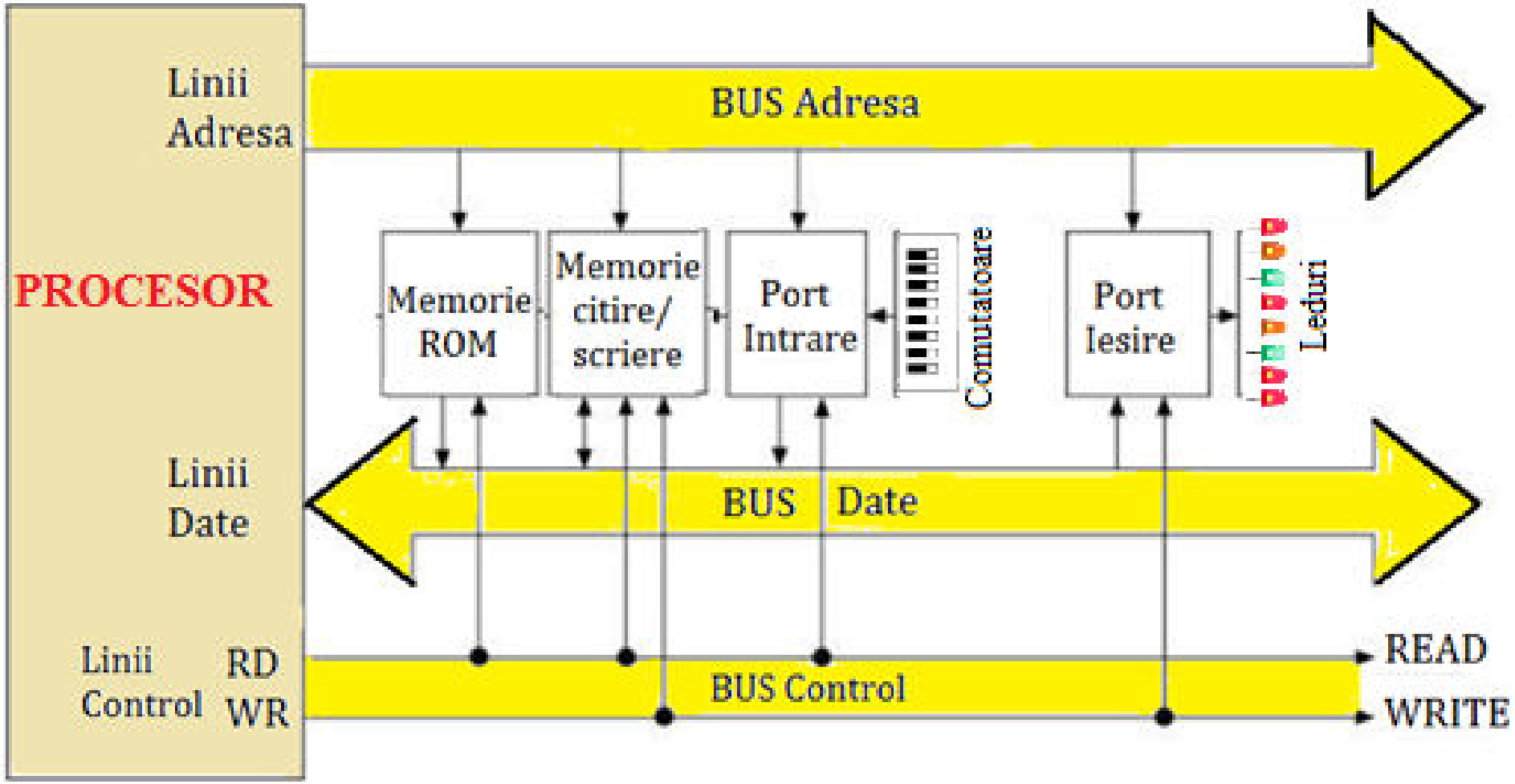


Ex: procesor la 2 GHz – are perioada ciclului de ceas de
$1/(2 * 10^9) = 0,5$ ns
Tema 1: specificati perioada ciclului de ceas pt un procesor "la 500 MHz"
Tema 2: daca un procesor are perioada ciclului de ceas de 2 ns, specificati frecventa lui

ciclul de ceas (clock cycle) = timpul intre 2 fronturi crescatoare (sau descrescatoare) consecutive ale unui semnal de ceas periodic
= masura utilizata pt a specifica timpul necesar executiei unei anumite sarcini (task)

- $timpul(durata)_{CPU}$ poate fi exprimat în funcţie de frecvenţa ceasului $f_C=1/t_C$
! Sunt invers proportionale !!!

Interfatarea cu exteriorul prin periferice de intrare și ieșire



Ciclu masina

- Fiecare SC contine un **ceas intern**
  - specifica ***intervalul de timp dintre 2 operatii consecutive*** si
  - ajuta la sincronizarea tuturor dispozitivelor din sistem

- CÂND SCRIEM PROGRAME (indiferent ca folosim HLL sau LLL) acestea cuprind ***secvenţe de instrucţiuni*** necesare îndeplinirii unei anumite sarcini.
- Aceste *instrucţiuni* sunt apoi "traduse" în ***secvenţe echivalente de instrucţiuni*** **în limbaj mașină** (de către asamblor) pe care procesorul le înţelege; ulterior, **S.O. încarcă programul în memoria principală** (cu ajutorul unui program încărcător, numit loader), **îi indică procesorului locaţia respectivă** și **"îl ghidează" spre execuţia lui**.

- Fiecare instructiune (mov ax,bx, etc) implica mai multe operatii interne (="ciclii masina") ce trebuie executate de procesor, toate realizandu-se sincronizat cu ceasul intern
- - o instructiune contine in gen. mai multi ciclii masina (de la 3 in sus), iar fiecare **ciclu masina** dureaza mai multi tacti ("stari")

- ne referim la High și Low *Level Languages*, adică limbaje de programare de nivel ridicat, respectiv scăzut

- **Exemple de ciclii masina:**
  fetch (extragere instructiune din memorie), **RD** / **WR memorie**, **RD** / **WR port**, etc

=> orice instructiune - este o combinatie de ciclii masina si
- incepe cu un ciclu de tip **FETCH** = "**extrage codul instructiunii din memorie**"

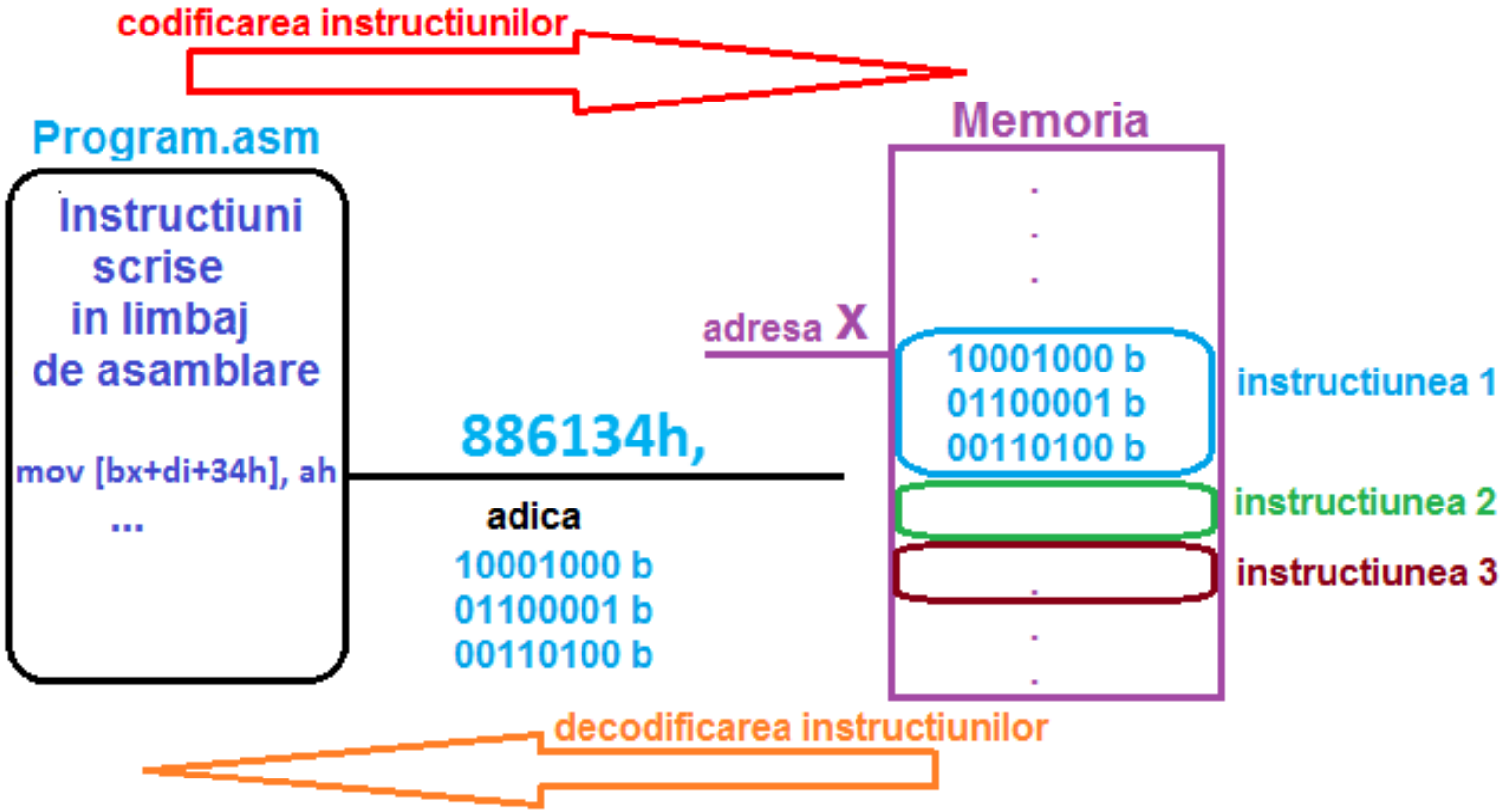
- **performanta executiei unei instructiuni** in general este specificata in ***ciclii ceas*** (in loc de secunde)

- Cand se mentioneaza termenul "*ceas*" se face referire la ceasul sistem (ceasul folosit de UCP)

- totusi, exista anumite busuri care detin si ele propriul ceas
(in gen cu durata mai mare decat durata ceasului UCP, ducand la usoare intarzieri, dar care nu afecteaza major performanta sistemului)

Codificarea/ decodificarea instrucţiunilor

- Codificarea instrucţiunilor:
  - trebuie realizată pentru că toate instrucţiunile (scrise în cadrul unui program) **trebuie transformate (de către asamblor) în cod mașină și depuse în memorie**, ca de acolo CPU să le poată lua și executa;
  - => orice instrucţiune (indiferent că e scrisă în LLL sau HLL) se va transforma în șiruri de 0 și 1 grupate în octeţi.
  - Acești octeţi sunt depuși în memorie (tot programul va «suferi» aceleași modificări) și deci va ajunge undeva în memorie

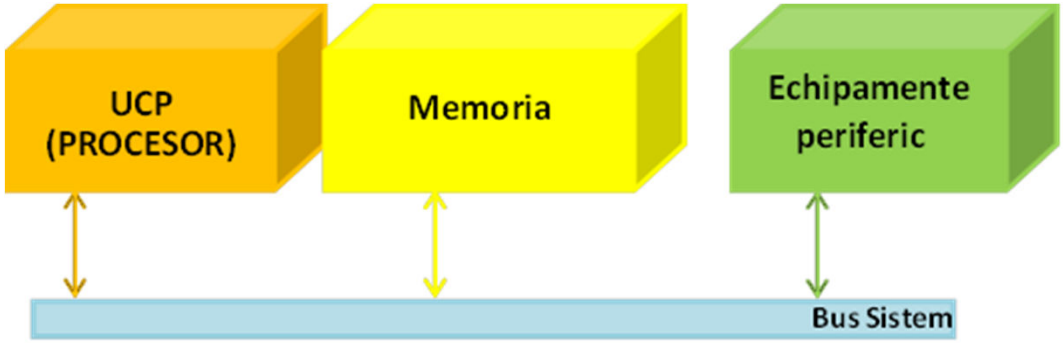


4. Calculatorul von Neumann

Criteriile von Neumann

- **Ideea principala: atat instructiunile cat si datele sunt pastrate in aceeasi memorie**

Un calculator (Sist de Calcul) *cu program memorat* trebuie să posede 3 ***Componente hardware principale***



(1) **UCP** (in gen. = procesor)
– asigura prelucrarea datelor si controleaza functionarea SC

(2) **Unitate de memorie** (interna /principala/ de baza)
din care se pot citi instrucţiuni şi operanzi (date) şi în care se depun rezultate;

(3) **Unitatea de intrare/iesire** (I/O)
– transferul datelor intre calculator si ext.

IDEEA DE BAZA

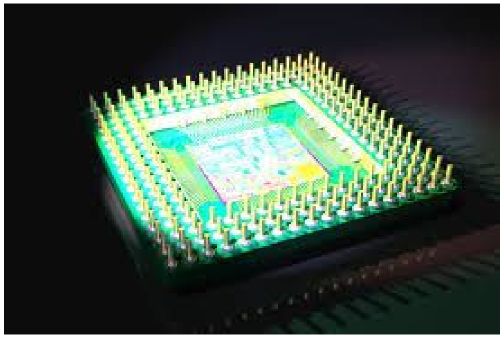
Cele 3 componente – legate printr-un **Bus** ce transporta informatia

Bus (magistrala) = un grup de semnale electrice sau fire folosite pentru a transfera informatie

4. Calculatorul von Neumann

Unitatea centrala de procesare (UCP)

- **Microprocesorul**
- **(UCP=Unitatea centrala de procesare)**



= Elementul de bază al unui SC

= un *chip* deosebit de complex plasat de obicei pe placa de bază a sistemului de calcul

- asigură:

- procesarea datelor: interpretarea, prelucrarea şi controlul acestora,
- supervizează transferurile de informaţii şi
- - controlează activitatea generală a celorlalte componente care alcătuiesc SC

4. Calculatorul von Neumann

Unitatea centrala de procesare (2)

Un calculator cu program memorat:

3 ***Componente hardware principale***

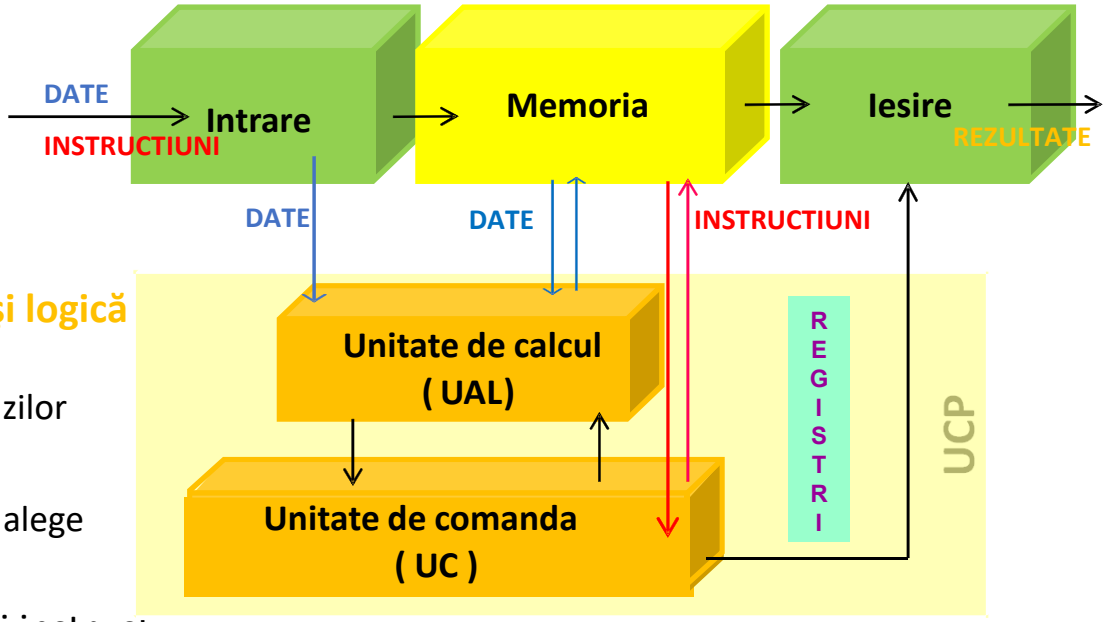
(1) **UCP** – contine **UAL, UC, registri**

•Unitate de calcul (UAL – unitate aritmetică şi logică sau unitate de execuţie)

- **e**xecuta operaţii aritmetice şi logice asupra operanzilor

•Unitate de comandă (control) (UC)

- interpreteaza instrucţiunile extrase din memorie şi alege diferite acţiuni pe baza rezultatelor obtinute.



• **Registri** – pt stocarea temporara a datelor si instruct.

- *accesarea* **datelor stocate in registri** este *mai rapida* decat cea a datelor din memorie
- *nr de registri* variaza de la un tip de procesor la altul:

Ex: Pentium are 8 registri de date si alti 8 registri diferiti, in timp ce Itanium are 128 registri doar pt date de tip intreg

(2) **Unitate de memorie**

(3) **Unitatea de intrare/iesire** (I/O)

In modelul von Neumann **NU exista distinctie intre instructiuni si date (operanzi)**, ele implicit fiind **executate secvential** (pe masura ce apar in program)
(cu exc. situatiilor cand apar apeluri de proceduri –duc la salturi in program)

La **sistemele von Neumann**:
conceptul memoriei singulare (amestecat date+instructiuni)

La **sistemele Harvard**:
exista memorii (*cache*) **separate** pentru instructiuni si date

Unitatea centrala de procesare (3)

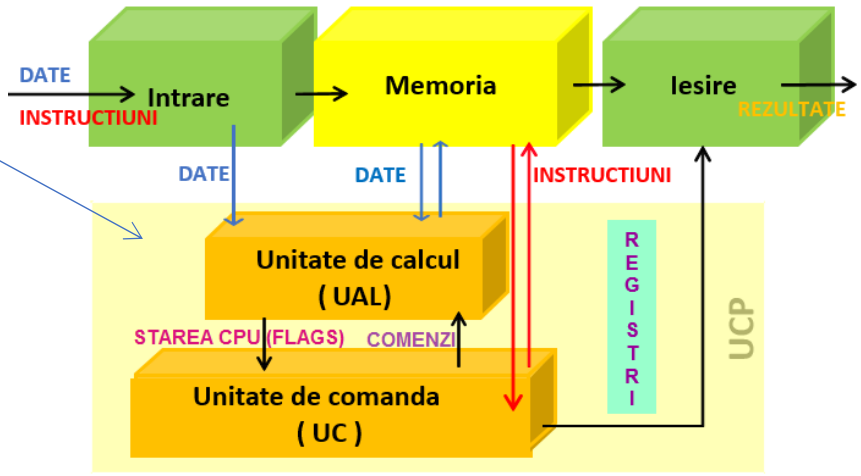
Unitatea aritmetică şi logică (UAL)

- realiz. operaţiile aritmetice/logice

Datele numerice sunt **procesate** pe baza unui *set de instrucţiuni*, având ca op. de bază: adunarea, scăderea şi operaţiile logice.

- contine un ***sumator*** *(*toate op. aritm. se reduc la o succesiune de op. de adunare)

Schema bloc simplificata a UCP



Datele asupra cărora se realiz. operaţia pot proveni: din memorie sau de la o intrare externa (port).

Operaţia de executat e specif. de semnalele generate de **UC**

Genereaza informatii (semn,transport/imprumut,etc) in ***registrul de stare*** ***(flags)*** despre rezultatul ultimei instructiuni aritmetice/logice executate (“feedback”)

Registrii = o memorie interna pt UCP

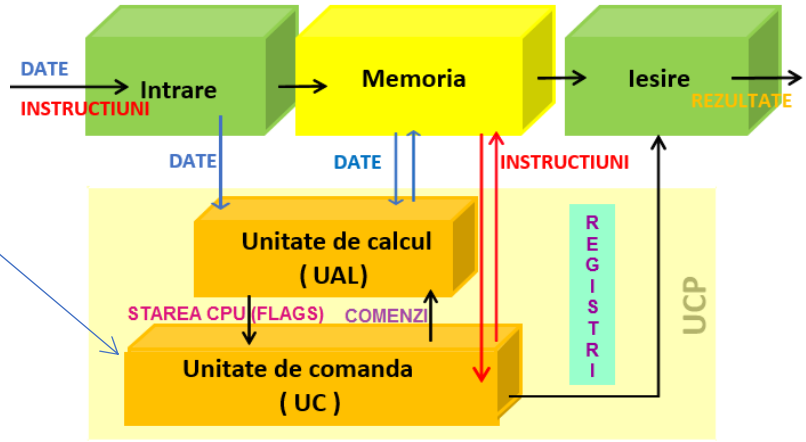
- pastreaza temporar operanzii unei operatii aritmetice/logice, rezultatele sau adresele lor
- pot fi: *generali* sau *speciali*
- o parte din registrii nu sunt accesibili prin program (registrii de lucru, ex: reg instructiuni)

Interconexiunile = magistrala interna a UCP

- asigura comunicatia dintre UAL, UC si registrii

Unitatea centrala de procesare (4)

Schema bloc simplificata a UCP



Unitatea de comanda si control (UC)

- controleaza functionarea UCP
- coordonează op. celorlalte unităţi

prin generarea semnalelor de temporizare şi control;

Realizeaza operatia de “fetch”

= extrage instructiunea din memorie si o decodeaza pt a vedea ce operatie are de executat UCP

- la terminarea executiei unei instructiuni,

se trece la urmatoarea (adresa instructiunii e pastrata in reg. PC)

- toate procesoarele au in gen un registru numarator de program *program counter* (PC) = un marker (pointer) la instructiunea pe care UCP urmeaza sa o execute;

uneori PC e numit si pointer la instructiune *instruction pointer* (IP).

Exista si un registru de instructiune *instruction register* (IR)

ce pointeaza spre instructiunea ce se executa la mom. curent

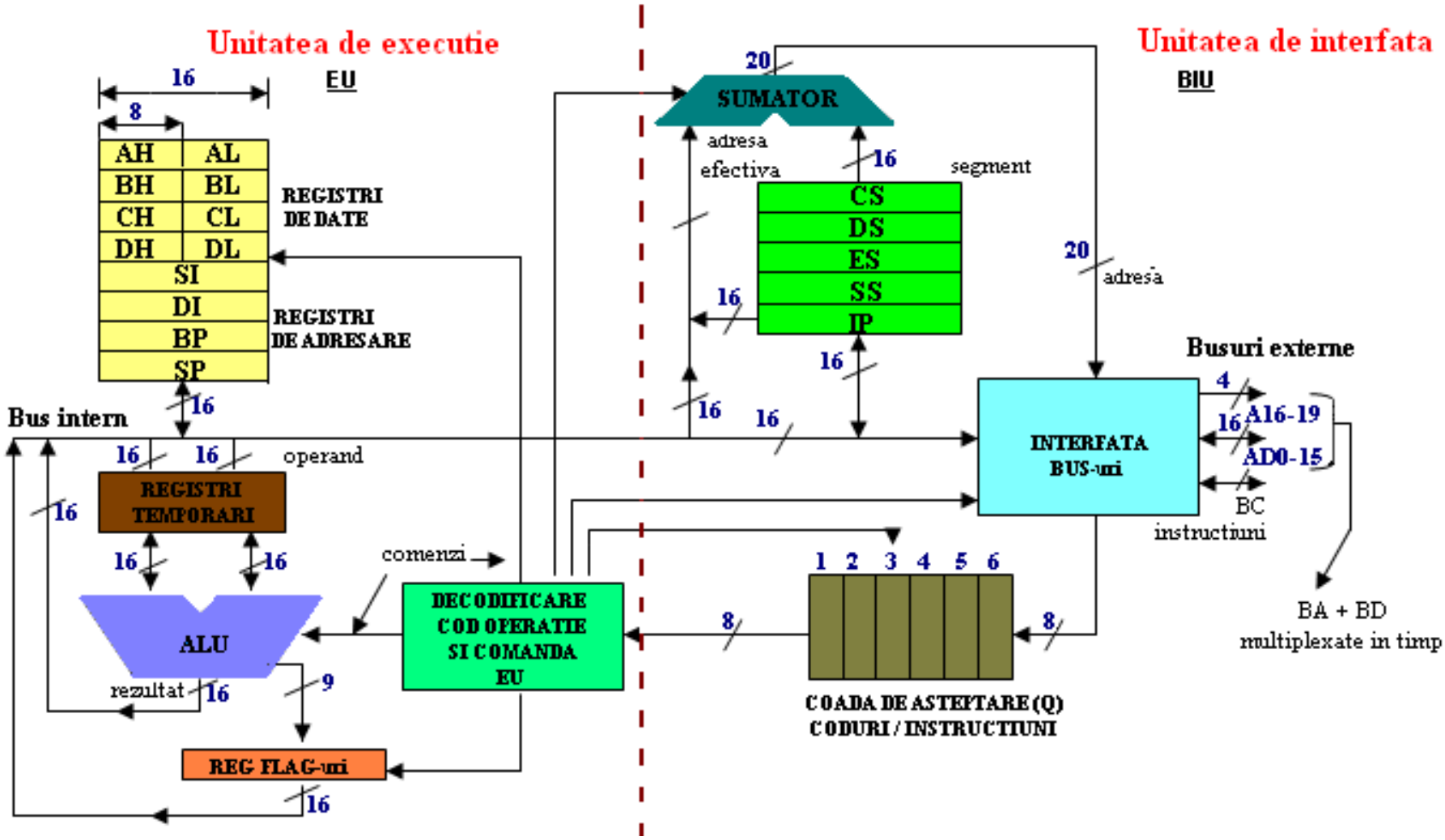
Registrii = o memorie interna pt UCP

- pastreaza temporar operanzii unei operatii aritmetice/logice, rezultatele sau adresele lor
- pot fi: *generali* sau *speciali*
- o parte din registrii nu sunt accesibili prin program (registrii de lucru, ex: reg instructiuni)

Interconexiunile = magistrala interna a UCP

- asigura comunicatia dintre UAL, UC si registrii

Cum sunt aranjate de fapt ALU si UC in interiorul CPU ? Unde sunt ele pe schema de mai jos ?

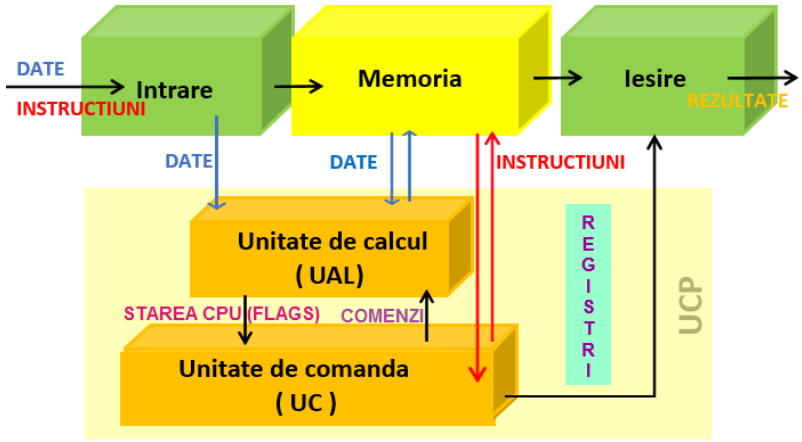


4. Calculatorul von Neumann

Unitatea centrala de procesare (5)

UC ii specifica lui **UAL** care este operatia pe care trebuie sa o execute, iar **UAL** dupa obtinerea rezultatului, il poate plasa inapoi in registrii sau in memoria principala.

Schema bloc simplificata a UCP



La procesoarele RISC (de ex. MIPS si Itanium), rezultatul este intotdeauna scris *intr-un registru*

Procesoarele CISC (cele din fam. 80x86, de ex Pentium) nu au aceasta restrictie, avand posibilitatea de a depune rezultatul *intr-un registru* sau *intr-o locatie din memoria* principala

!!! Executia unei instructiuni implica 1-operatia de fetch, 2-executia instructiunii si 3-scrierea rezultatului !!!

= ***ciclu de executie***

Modelul "von Neumann" - bazat pe operatiile ***Fetch – decode - execute*** pentru a executa programe :

1. Fetch, 2. Decode, 3. Data Transfer, 4. Execute - in gen cei 4 pasi , nu 3 !

O iteratie a ciclului se desfasoara astfel:

1. UC aduce ("fetches") urmatoarea instructiune de program din memorie, folosind PC (pt a determina unde este localizata instructiunea)
2. instructiunea este decodificata intr-un limbaj pe care UAL il intelege
3. operanzii necesari executarii instructiunii sunt adusi din memorie si plasati in registre, in UCP
4. UAL executa instructiunea si plaseaza rezultatul in registre sau in memorie

CISC (Complex Instruction Set Computer)
RISC (Reduced Instruction Set Computer)

Unitatea centrala de procesare (6)

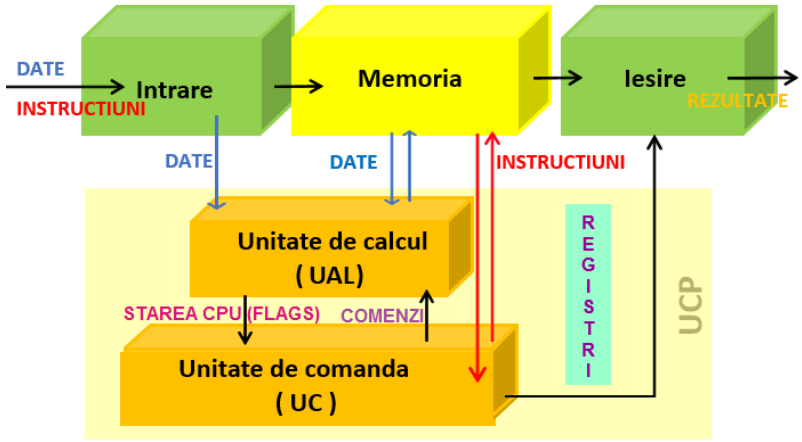
UC conţine un dispozitiv de secventiere:
1. UC citeşte instrucţiunea (fetch) prin generarea unei adrese şi a unei comenzi de citire către memorie.
2. Instrucţiunea de la adresa resp este transferată UC pentru decodare.
4. UC generează apoi semnalele pt execuţia instrucţiunii de catre UAL

Pipelining

UC va trebui sa astepte pana ce instructiunea este adusa din memorie (operatia de fetch)
Apoi UAL va trebui sa astepte pana cand operanzii necesari instructiunii sunt adusi din memorie
=> se pierde timp pastrand UC si UAL in stare *idle* pana ce instructiunea si operanzii sunt adusi din memorie .
Solutie: pregatirea instructiunii si a operanzilor in avans
(inainte ca UC sau UAL sa aba nevoie de ele)
– pt pastrarea lor exista registrii speciali : *prefetch buffers*.

Daca se stie ca *programul e secvential*, se poate realiza acesta operatie de pre-fetch pt instructiunea ce urmeaza a fi executata imediat ce UC a terminat de decodificat instructiunea curenta (similar, prin realiz operatiei de pre-fetch asupra operanzilor, se evita starile de *idle* ale ALU)

Schema bloc simplificata a UCP

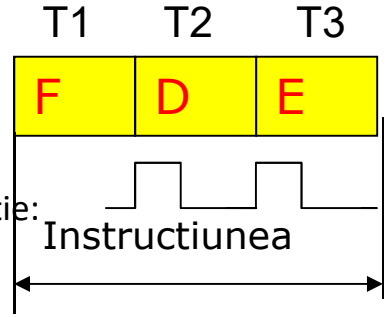


Pipelining

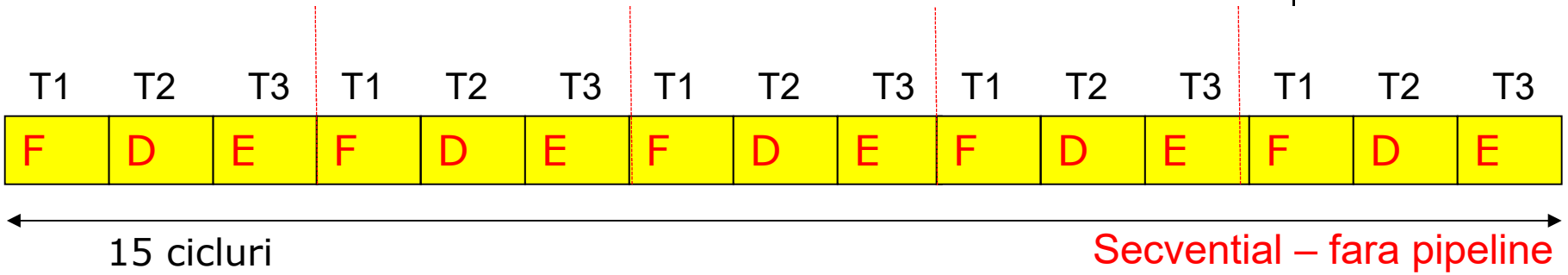
- nu creste viteza de executie a instructiunilor prin reducerea nr de cicluri:
- fiecare instructiune va dura acelasi nr de cicluri, doar ca: va creste nr de instr. executate pe unitatea de timp
- **executare suprapusa a instructiunilor**

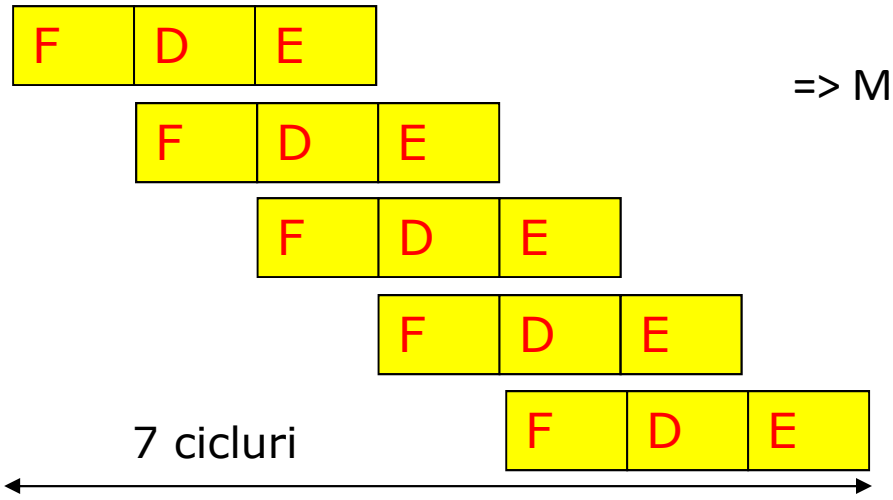
- Un pipeline simplu cu 3 etaje

O instructiune se executa in 3 pasi:
F-Fetch, D-Decode, E-Execute



F,D,E = operatii realizate de componente independente unele de altele
- fiecare din aceste componente, imediat ce a terminat taskul curent, se pregateste pt urmat. operatie:
- acest proces de creare a unei cozi de instructiuni = **pipelining**





=> Mai mult de 50% din timp s-a salvat

Clasificarea lui Flynn

- - arhitecturile de calculatoare – nu exista un sistem de clasificare al lor unanim recunoscut sau acceptat
- **Clasificarea lui Flynn** (1966) - deşi foarte generală, este totuşi cea mai folosită
- - nu foloseste structura maşinilor ci relaţia instrucţiune de executat – date de prelucrat
- Prof. Michael Flynn a clasificat ***arhitecturile de calculatoare*** în 4 categorii = "taxonomia Flynn"
- Criteriul: prezenţa unui singur şir sau a mai multor şiruri/fluxuri ("stream") de instrucţiuni şi de date
- Şir/flux de instrucţiuni: set de instrucţ secvenţiale executate de o UCP, avand asociat un sg contor de program
- Un sistem cu n UCP are n contoare de program şi deci n fluxuri de instrucţiuni
- Şir/flux de date: un set de operanzi = fluxul secvenţial de date necesar şirului de instrucţiuni
-
- SISD = Single Instruction Stream Single Data Stream
- SIMD = Single Instruction Stream Multiple Data Stream
- MISD = Multiple Instruction Stream Single Data Stream
- MIMD = Multiple Instruction Stream Multiple Data Stream

Nume arhitectura	Flux de instruct	Flux de date
SISD	1	1
SIMD	1	n
MISD	n	1
MIMD	n	n

- - sistemele conventionale uni-procesor – calculatoarele von Neumann = SISD
- - sistemele paralele: pot fi SIMD sau MIMD
- - practic nu au existat sisteme MISD, dar exista autori care includ aici sistemele pipeline
- - clasificarea lui Flynn a fost extinsa in 1978 de D.J. Kuck care a impartit fluxul de instructiuni in scalar si vectorial (array) atat pentru single cat si pentru multiple stream => in total 16 categorii de arhitecturi

Paralelism la diferite nivele

Performanta poate fi crescuta prin ***realizarea unui nr de operatii in paralel***, in loc de secvential

Paralelismul poate fi implementat la diferite nivele :

Instruction-level Parallelism

Cel mai simplu mod de a executa o secvență de instrucțiuni într-un procesor:
de a completa toate etapele instructiunii curente înainte de a începe pașii următoarei instrucțiuni.
Dacă vom suprapune executia etapelor mai multor instrucțiuni succesive, timpul total de executie va fi redus. De exemplu, următoarea instrucțiune poate fi preluata din memorie (fetch) în același timp cat se executa o operație aritmetică pe operanzii din registri (instrucțiunea curentă).

Această formă de paralelism este numita ***pipelining***.

Multicore Processors

Mai multe unitati de procesare pot fi fabricate intr-un singur cip
In literatura tehnica: termenul ***core (miez, nucleu)***= fiecare din aceste unitati de procesare (“procesoare”)

=> termenul ***procesor*** se foloseste atunci pentru a referi cipul complet
- exista procesoare ***dual-core***, ***quad-core***, ***octo-core***
= cipuri care au 2, 4, 8 miezuri procesor pe acelasi cip fizic

Multiprocessors

- Sistemele de calcul pot conține mai multe procesoare, fiecare eventual conținând mai multe nuclee.
- Astfel de sisteme sunt numite "***multiprocesoare***" - fie executa sarcini ale unor aplicatii diferite în paralel, fie execută subsarcini ale unei aceleiasi aplicatii în paralel.

- Toate procesoarele au de obicei acces la toate zona de memorie în astfel de sisteme
  => "***multiprocesor cu memorie partajată "***
- - performanța ridicata data de complexitatea si costurile crescute,
  - utilizarea de mai multe procesoare și unități de memorie,
  + rețelele mai complexe de interconectare.

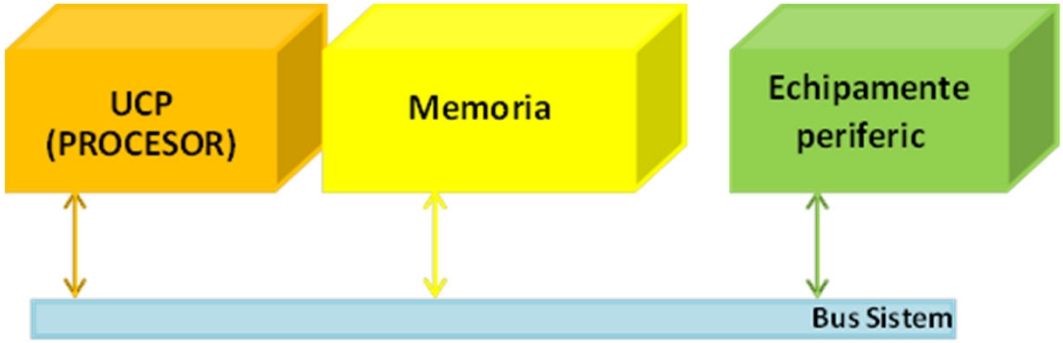
- Spre deosebire de sistemele multiprocesor, exista si notiunea de ***grup interconectat de calculatoare*** pentru realizarea de putere de calcul ridicata.
- Computerele au în mod normal acces doar la propriile unități de memorie.
- Când sarcinile ce sunt executate au nevoie de a partaja date, fac acest lucru prin schimbul de mesaje într-o rețea de comunicații.
- Această proprietate le distinge de multiprocesoarele cu memorie partajată
- ***=> "multicomputere pe baza de mesaje "*** *(message-passing multicomputers*)

4. Calculatorul von Neumann

Criteriile von Neumann

- **Ideea principala: atat instructiunile cat si datele sunt pastrate in aceeasi memorie**

Un calculator (Sist de Calcul) *cu program memorat* trebuie să posede 3 ***Componente hardware principale***



(1) **UCP** (in gen. = procesor)
– asigura prelucrarea datelor si controleaza functionarea SC

(2) **Unitate de memorie** (interna /principala/ de baza)
din care se pot citi instrucţiuni şi operanzi (date) şi în care se depun rezultate;

(3) **Unitatea de intrare/iesire** (I/O)
– transferul datelor intre calculator si ext.

IDEEA DE BAZA

Cele 3 componente – legate printr-un **Bus** ce transporta informatia

Bus (magistrala) = un grup de semnale electrice sau fire folosite pentru a transfera informatie

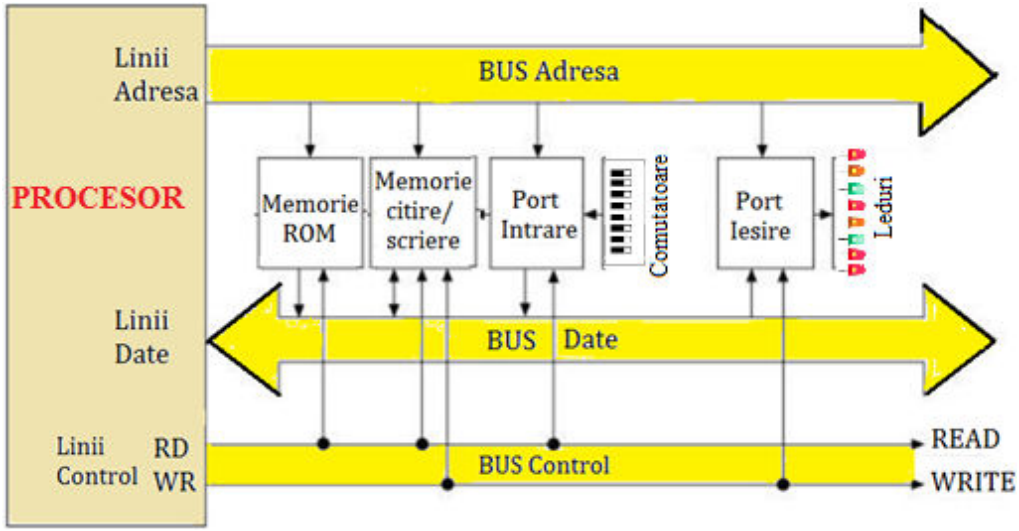
4. Calculatorul von Neumann

Busul sistem

- **BUSUL SISTEM cuprinde:**
- **Busul de ADRESA (BA)** :
- = linii paralele de semnal: 16,20,24,32,…
- este unidirectional : prin el *procesorul transmite* adresa locatiei de memorie unde se gaseste informatia (date/instructiuni)
- sau bidirecţional dacă procesorul conţine mem. cache

- **Busul de DATE (BD)** :
- = linii paralele de semnal: 8, 16, 32, 64 biti
- - este bidirectional si in plus un singur dispozitiv are iesirea validata (va putea transfera date pe busul BD)
- => dispozitivele au iesiri three-state (3 stari)

- **Busul de CONTROL (BC)** :
- = cateva linii de semnal: 4…10, sau mai multe
- - procesorul trimite semnale pe aceste linii inspre memorie si porturile I/O in vederea sincronizarii, gestionarii intreruperilor si accesului la memorie prin DMA (direct memory access), etc



Schema bloc simplificata a unui microsistem standard cu microprocesor

BUS =un grup de linii destinate transferului paralel al informatiilor: nr liniilor magistralei = (in gen.) cu lungimea cuvantului transferat

- poate fi unidirectional sau bidirectional

- câte o linie pentru fiecare bit de informaţie transmis, deci 16 linii pentru o magistrală de 16 biţi.

- la un moment dat nu poate fi selectata decat o sursa si una sau mai multe destinatii, altfel pot apare rezultate imprevizibile (pierderi de date)

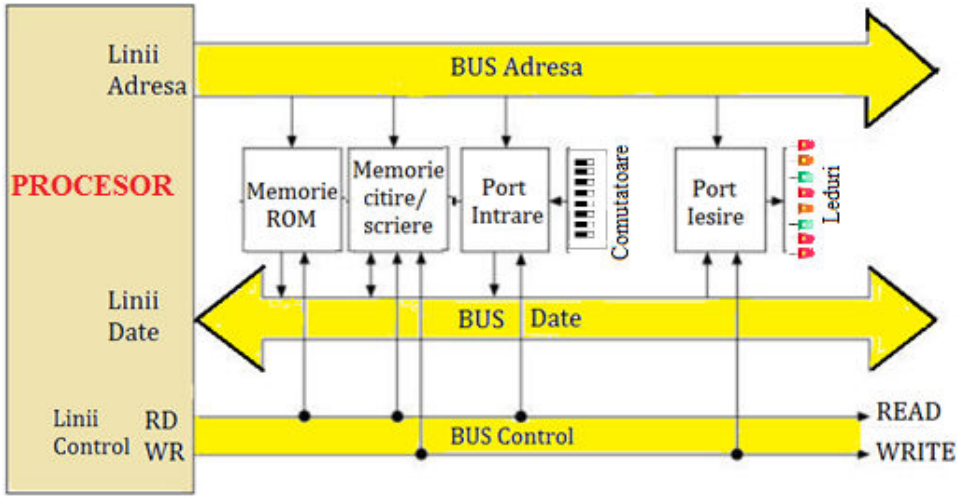
- selectia sursei/destinatiei se realiz cu multiplexoare si decodificatoare

Busul sistem (2)

- **Latimea (size) BA** – determina cantitatea de memorie fizica pe care o poate adresa procesorul
- Ex : procesorul Pentium are 32 linii adresa
- =>Pentium poate adresa pana 4Gocteti (sau 2^{32} locatii) de mem

- **Latimea (size) BD** – indica dimensiunea informatiei transferate intre procesor si memorie sau dispozitivele I/O
- Ex : procesorul Pentium are 64 linii date
- => Fiecare transfer de date poate muta maxim 64 biti o data

- **BC** include semnale ce indica tipul actiunii ce are loc pe busul sistem; exemplu: memory read, memory write, I/O read, I/O write, interrupt, interrupt acknowledge, bus request, bus grant
- Ex: cand procesorul doreste *sa scrie date in memorie*, va genera semnalul *memory write*
- Ex: cand procesorul doreste *sa citeasca date de la un dispozitiv I/O*, va genera semnalul *I/O read*



Schema bloc simplificata a unui microsistem standard cu microprocesor

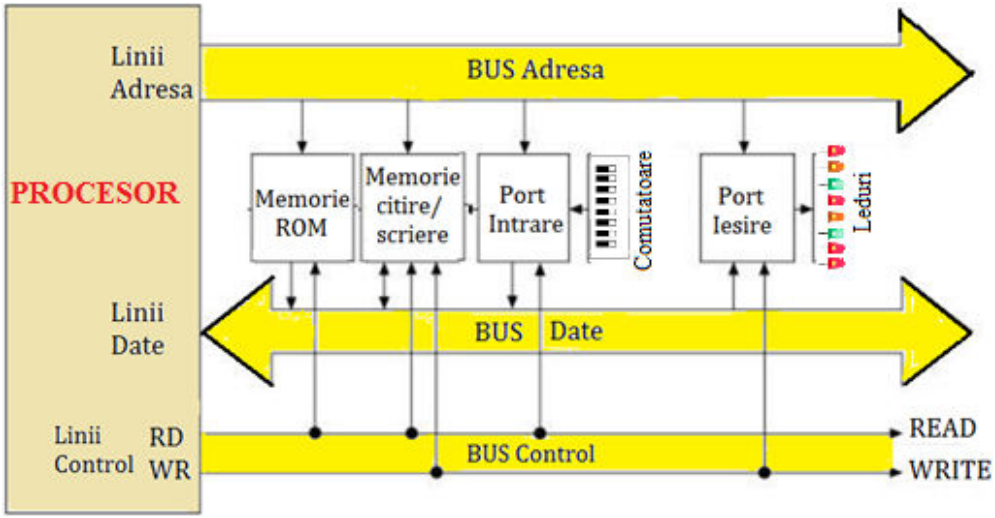
Tema: specificati cantitatea de memorie adresabila si dimensiunea maxima a datelor pentru un procesor Itanium stiind ca are latimea BA si latimea BD duble comparativ cu cele ale procesorului Pentium

Semnalele read si write – dau directia fluxului de date de pe BD
- cand ambele=1 => CPU si memoria nu comunica intre ele
- cand Read=0 => CPU citeste date din mem (sist transfera date din mem -> CPU)
- cand Write=0 => CPU scrie date in mem (sist transfera date de la CPU -> mem)

Busul sistem (3)

la familia 80x86, BD are 8,16,32,64 linii
- faptul ca un sistem are BD pe 8 biti nu limiteaza accesul acestuia la date pe 8 biti (octeti)
=> doar poate transfera 1 singur octet pe ciclu de memorie
Latimea BD afecteaza cel mai mult performanta SC

Familia 80x86 furnizeaza **2 spatii adrese distincte**:
una pt mem, alta pt dispoz I/O
Latimea BA pt mem variaza la dif fam 80x86 (tabel),
Latimea BA pt porturi e aceeasi:
16 biti = > 65.536 locatii I/O diferite
(IBM original avea doar 1024)



Schema bloc simplificata a unui microsistem standard cu microprocesor

Procesor	Latimea BD	Latimea BA	Cant. maxima de memorie adresabila
8088, 80188	8 biti	20 biti	1 Mega
8086, 80186	16 biti	20 biti	1 Mega
80286	16 biti	24 biti	16 Mega
80386SX	16 biti	24 biti	16 Mega
80386DX	32 biti	32 biti	4 Giga
80486	32 biti	32 biti	4 Giga
80586/Pentium	64 biti	32 biti	4 Giga

Desi familia 8086 suporta 2 spatii adrese,
NU suporta 2 BA, adica
sistemul partajeaza ("shares") BA pt ambele (disp I/O si mem)

Liniile de control aditionale decid daca adresa de pe BA e
pt mem sau pt un dispoz I/O

Dimensiunea operanzilor

la familia 80x86, BD are 8, 16, 32, 64 linii – in gen dimensiunea operanzilor este limitata de aceste valori

- faptul ca un sistem are BD pe 8 biti nu limiteaza accesul acestuia la date pe 8 biti (octeti)
  => doar poate transfera 1 singur octet pe ciclu de memorie

Latimea BD afecteaza cel mai mult performanta SC

8008 = "Procesor pe 8 biti " -> registrii interni sunt de 8 biti
- operanzii interni: pe B (octet) => poate transfera **1** singur octet pe ciclu de memorie

8086 = "Procesor pe 16 biti " -> registrii interni sunt de 16 biti
- operanzii interni: pe B (octet), W (cuvant) => poate transfera **2** octeti pe ciclu de memorie

80386 = "Procesor pe 32 biti " -> registrii interni sunt de 32 biti
- operanzii interni: pe B (octet), W (cuvant), DW (dublucuvant) => poate transfera **4** octeti pe ciclu de memorie

Xeon, Pentium 4, DualCore (au *EM64T*)= "Procesor pe 64 biti " -> registrii interni sunt de 64 biti
- operanzii interni: pe B (octet), W (cuvant), DW (dublucuvant) sau QW (cvadruplucuvant)
  => poate transfera **8** octeti pe ciclu de memorie

Procesor	Latimea BD	Latimea BA	Cant. maxima de memorie adresabila
8088, 80188	8 biti	20 biti	1 Mega
8086, 80186	16 biti	20 biti	1 Mega
80286	16 biti	24 biti	16 Mega
80386SX	16 biti	24 biti	16 Mega
80386DX	32 biti	32 biti	4 Giga
80486	32 biti	32 biti	4 Giga
80586/Pentium	64 biti	32 biti	4 Giga

http://ark.intel.com/
http://www.cpu-world.com/CPUs/CPU.html

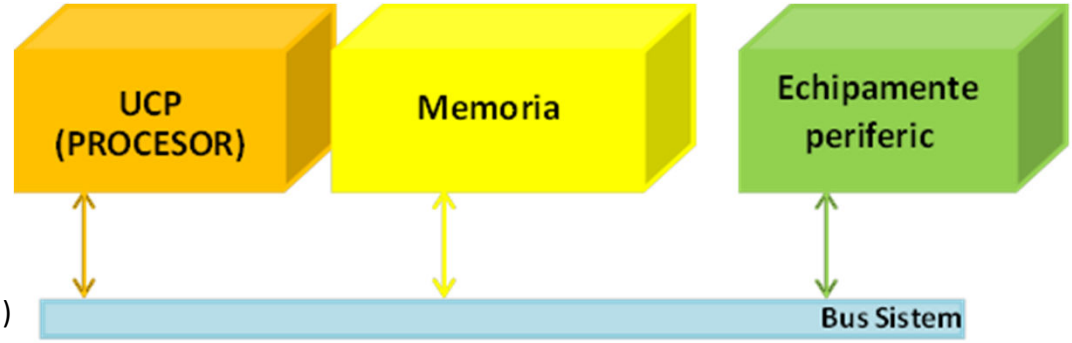
4. Calculatorul von Neumann

Criteriile von Neumann

- **Ideea principala: atat instructiunile cat si datele sunt pastrate in aceeasi memorie**

Un calculator (Sist de Calcul) *cu program memorat* trebuie să posede 3 ***Componente hardware principale***

(1) **UCP** (in gen. = procesor)
– asigura prelucrarea datelor si controleaza functionarea SC



(2) **Unitate de memorie** (interna /principala/ de baza) din care se pot citi instrucţiuni şi operanzi (date) şi în care se depun rezultate;

IDEEA DE BAZA

(3) **Unitatea de intrare/iesire** (I/O)
– transferul datelor intre calculator si ext.

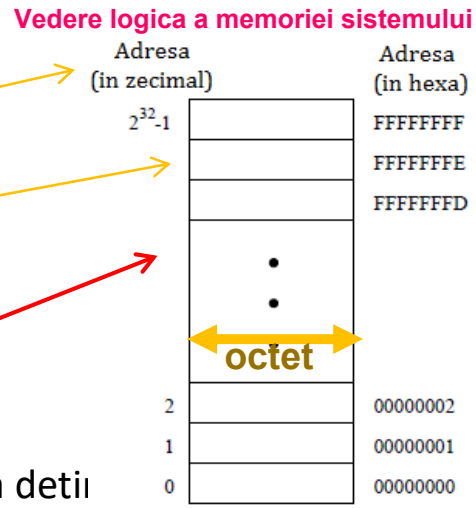
Cele 3 componente – legate printr-un **Bus** ce transporta informatia

Bus (magistrala) = un grup de semnale electrice sau fire folosite pentru a transfera informatie

Memoria

- este sursa/destinatia tuturor informatiilor

- - memoria dintr-un SC=mici comutatoare electronice cu 2 stari stabile: inchis/deschis– corespund starilor 0 si 1 => fiecare comutator poate fi reprez de un bit => unitatea de memorie = milioane de secvente ordonate de *biti*;
- pt a fi mai usor de lucrat cu ea – organizata in octeti => unitatea de memorie = o secventa ordonata de *octeti*
  - - fiecare octet e identificat prin numarul lui de ordine = locatia lui = adresa
  - => **Memoria** - organizata ca ***o colectie de locatii de memorie***
  - (fiecare locatie are asociata **o adresa** prin care este accesata)
  - - adresarea se realizeaza prin **linii de adresa**;
  - nr lor det. capacitatea maxima adresabila a memoriei;
  - ex: 16 linii => 2^{16} locatii de memorie
  - - o locatie de memorie se caracterizeaza prin:
  - ***adresa*** (pozitia locatiei in cadrul memoriei)
  - + ***continut*** (valoarea memorata in acea locatie)

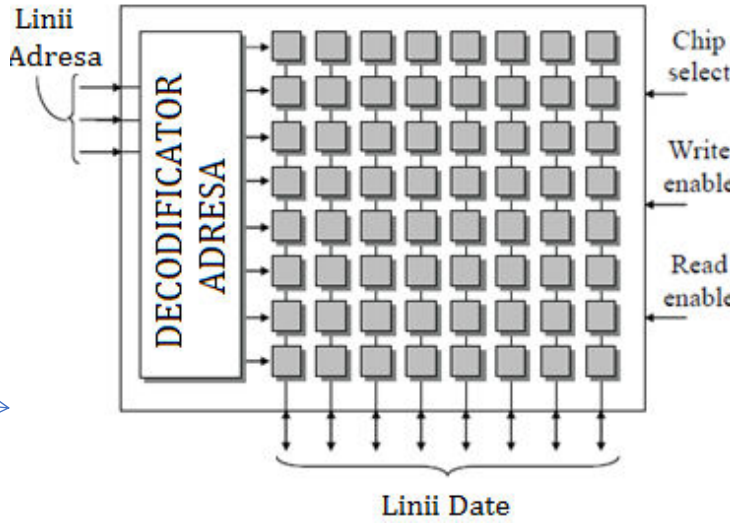


  - Ex: procesorul Pentium poate adresa prin intermediul celor 32 linii de adresa detii memorie principala (2^{32}=4G) = **spatiul de adresare al memoriei** -> cantitatea de memorie dintr-un SC e determinata de cat de mult din acest *"spatiu de adresare al memoriei"* e implementat cu chipuri de memorie
  - - cantitatea de informatie ce se poate memora intr-o locatie adresabila individual = nr cifrelor binare (nr biti)
  - = lungimea cuvantului de memorie;
  - Ex: memoria adresabila pe *octet*: fiecare *octet* are o adresa specifica (unica)
  - => se pot folosi entitati la adresare de 16, 32, 64 biti (adica se pot manipula cuvinte, dublucuvinte, cvadruplucuvinte, etc),
  - desi **unitatea adresabila este octetul**

Tema: specificati dimensiunea spatiului de adresare al Intel Itanium stiind ca procesorul are 64 linii de adresa

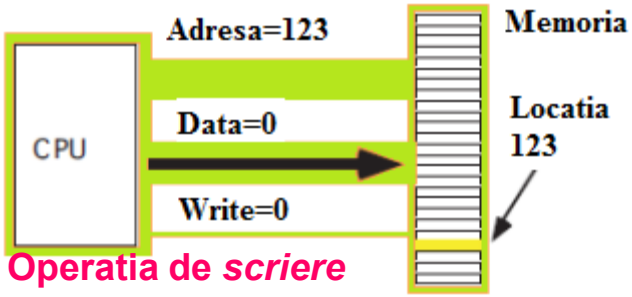
- **capacitatea memoriei** - exprimata in Kocteti(KB) sau multiplii lui: 1KB, 1MB, 1GB
- memoria (*ideal*) trebuie sa aiba:
- *capacitate mare, viteza ridicata ($timp_{acces}$ redus), cost scazut*
- viteza d.p. costul => 2 tipuri:
- ***o mem interna*** (=principala) rapida si
- ***o mem externa*** mai lenta, cu capacit mai mare
- nu afecteaza sensibil viteza de calcul: prelucrarea datelor si transferul informatiei se realizeaza la viteza de acces a memoriei interne

Diagrama bloc a sistemului de memorie

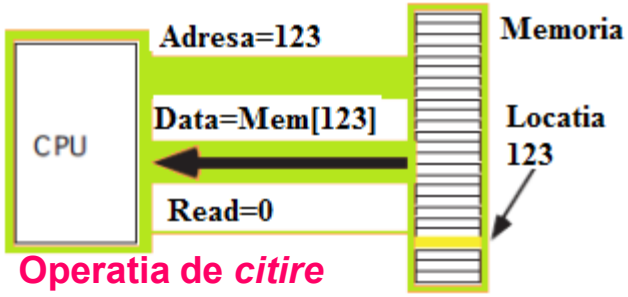


Operatia de *selectie* a locatiei

- Operatii realizate cu memoria:
- Operatia de selectie a locatiei
- Operatia de scriere si Operatia de citire



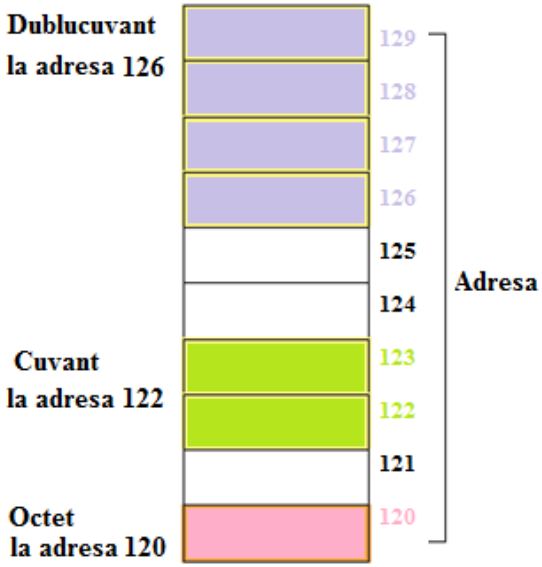
Operatia de *scriere*



Operatia de *citire*

Memoria (3)

- **Cum sunt stocate datele in memorie ?**
- Ca octeti (memorie adresabila pe octet) ! -> Intel - conventia Little Endian:
- - un octet = [120]
- - un cuvant = 2 octeti =[123][122]
- - un dublucuvant = 4 octeti=[129][128][127][126]
- Datele vin/merg (d)in memorie in grupuri de cate 8 biti !
- Ordonarea octetilor in memorie - dupa conventiile Little end-ian si Big end-ian
- Exista procesoare care pot selecta oricare din variante, de ex. MIPS si PowerPC implicit au big endian, dar pot functiona si dupa schema little endian
- – o schema nu e mai buna decat cealalta, insa trebuie tinut cont de reprezentarea aleasa ! In special la transferul de date intre mai multe SC
- (ce ar putea folosi scheme diferite la reprezentarea informatiei)



- Valoarea scrisa pe 4 octeti: 8 7 6 5 4 3 2 1 h se depune in memorie incepand de la adresa 0000h folosind una din conventiile:

- Little Endian
LSB ("End"-ian) se depune in memorie la locatia cu adresa cea mai mica ("Little")
- Intel

- Big Endian
LSB ("End"-ian) se depune in memorie la locatia cu adresa cea mai mare ("Big")
- Motorola

Adresa	Continutul
0003	87
0002	65
0001	43
0000	21

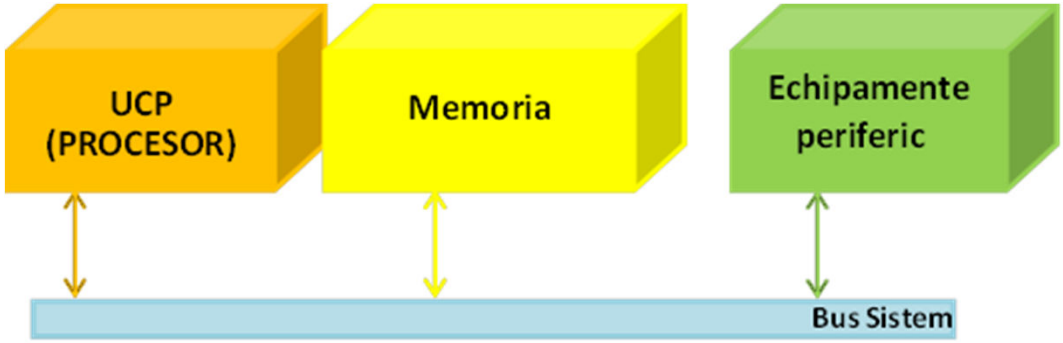
Adresa	Continutul
0003	21
0002	43
0001	65
0000	87

4. Calculatorul von Neumann

Criteriile von Neumann

- **Ideea principala: atat instructiunile cat si datele sunt pastrate in aceeasi memorie**

Un calculator (Sist de Calcul) *cu program memorat* trebuie să posede 3 ***Componente hardware principale***



(1) **UCP** (in gen. = procesor)
– asigura prelucrarea datelor si controleaza functionarea SC

(2) **Unitate de memorie** (interna /principala/ de baza)
din care se pot citi instrucţiuni şi operanzi (date) şi în care se depun rezultate;

IDEEA DE BAZA

(3) **Unitatea de intrare/iesire** (I/O)
– transferul datelor intre calculator si exterior

Cele 3 componente – legate printr-un **Bus** ce transporta informatia

Bus (magistrala) = un grup de semnale electrice sau fire folosite pentru a transfera informatie

Unitatea de intrare / iesire (I/O) sau **periferice** = echipamente ce realiz comunicarea SC cu exteriorul
perifericele - conectate prin interfete I/O– asigura transportul informatiilor de pe suporturi externe de informatie in memorie si invers, asigura conversia datelor + au rol de memorie tampon
(necesara deoarece UAL si UC au viteze de lucru mult mai ridicate decat dispoz ext)

- - ***circ de interfatare*** dintre dispoz I/O si UCP **gestioneaza transferul datelor**,
- convertind semnalele busului sistem intr-un format acceptat de dispoz respectiv

- interfatarea unui dispoz de I/O la SC se realiz in gen. printr-un ***controler I/O***
(dispoz I/O realiz comunicarea cu SC prin bus-ul sistemului, iar controlerul I/O realiz interfatarea intre ele)

In gen. un *controler I/O* are 3 tipuri de registre interne: registru de *date*, registru de *comanda*, registru de *stare*
- cand UCP vrea sa interactioneze cu un dispoz I/O, va comunica doar cu controlerul I/O asociat dispoz

- UCP poate accesa registrii interni ai controlerului I/O prin intermediul porturilor I/O

(un port = adresa unui registru intr-un controler I/O,
in analogie cu o adresa de memorie cand se face referire la o locatie de memorie)

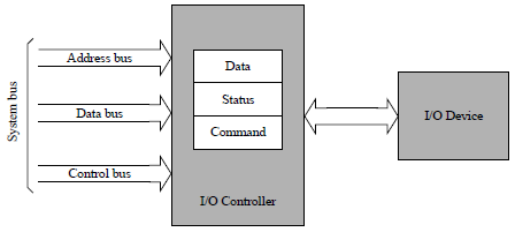
=> s-au alocat spatii din spatiul adreselor de memorie (unde nu exista memorie fizica asociata)
pentru porturile I/O =“I/O mapate in memorie” – scrierea la un port I/O e asem cu scrierea intr-o locatie de memorie => toate UCP suporta aceasta schema in mod mostenit (inerent)
- majoritatea procesoarelor (inclusiv PowerPC si MIPS) suporta I/O mapate in memorie
- daca spatiul adreselor de memorie este mic, nu se fol maparea I/O in mem (ar restrange si mai mult acest spatiu)

4. Calculatorul von Neumann

Periferice (2)

- - Intel 8086: avea 20 linii de adresa => putea adresa doar 1 MB de memorie
- => s-a alocat un spatiu separat pt adresele porturilor
- (=> spatiu de adrese I/O separat, maparea fiind numita "adrese I/O izolate")
- – sunt necesare instructiuni speciale de I/O pt accesarea spatiului de adrese I
- UCP este responsabila pt transferul datelor, cuvant cu cuvant
- -> executa bucle pana la terminarea datelor de transferat
- Aceasta schema = "I/O programate" – dezav major: consuma timpul UCP

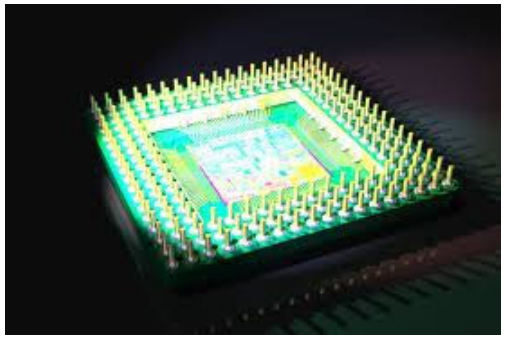
Diagrama bloc a interfetei cu un dispozitiv I/O generic



- Alta schema: prin interm tehnicii cu acces direct la memorie *direct memory access* (DMA): prin care UCP da comenzi: "transfera 10KB la portul I/O cu adresa 100" iar *controlerul DMA* realizeaza transferul datelor in continuare, eliberand UCP de aceasta sarcina; cand operatia e completa, este anuntat procesorul prin intermediul unui mecanism de intreruperi. Aceasta schema se fol la transfer de date in volum (bulk data) si nu pentru a transfera un cuvant (nu ar fi eficient).
- - UCP comunica cu dispoz I/O externe prin registre de intrare / iesire – in 2 moduri:
- A. In schema *I/O mapate in memorie*, registrele din interfata apar in harta memoriei SC, neexistand vreo diferenta intre accesul la o zona de memorie sau accesul la un dispoz I/O -> avantaje dpdv al vitezei, dar dezav ca foloseste din spatiul memoriei
- B. cu *instructiuni I/O*, UCP are instructiuni specializate ce realizeaza intrari/iesiri, deci doar procesorul poate da aceste comenzi speciale
- Conform principiului de compensare a diferenţelor de viteză dintre nivele de memorie, între memoria principală şi memoria auxiliară pot apărea nivele de memorie tampon (transparente utilizatorului) - sunt specifice fiecărui dispozitiv periferic fol. ca memorie externă şi se găsesc in gen în cadrul circuitelor de control ale dispozitivelor.

- **Dispozitive de intrare/iesire** (I/O) sau **periferice**
- Perifericele – 2 scopuri principale: Comunicarea cu exteriorul si Stocarea datelor
  - Functional: ele pot fi adresate (apelate) de catre CPU similar cu memoria,
    - dispun si ele de cate un set de adrese: 65 536 porturi I/O posibile
      - EX: 8088 -> busul de adrese AD15-0 pentru localizarea porturilor
      - au in gen 3 tipuri de registre: date, comenzi, stare
    - Transferul datelor intre porturi si CPU se face pe busul de date
    - Registrul acumulator (ACC) este cel care primeste/trimite datele pe porturi
    - Exemple:
    - IN AL, 110h ; de la adresa de port 110h se preia un **octet** in registrul accumulator **AL**
    - IN AX, 110h ; de la adresa de port 110h se preia un **cuvant** in registrul accumulator **AX**
    - OUT 112h, EAX ; din registrul extins **EAX** se scoate pe portul cu adresa 112h un **dublucuvant**
    - Concluzii:
- In general: schimbul de informatii cu exteriorul se realiz sub controlul CPU,
  - DAR exista tehnici prin care accesul la memorie se realiz cu o interventie minima a CPU:
  - - transferuri DMA;
- - controlerul I/O – actioneaza ca o interfata intre busul sistem si periferic,
- eliberand CPU de gestionarea detaliilor + gestioneaza interfata electrica

•... Am vazut :



- **Microprocesorul**
- **(Unitatea centrala de procesare) – dpdv logic:**

= Elementul de bază al unui SC

= un *cip* deosebit de complex plasat de obicei pe placa de bază a sistemului de calcul

- asigură:
  - procesarea datelor: interpretarea, prelucrarea şi controlul acestora,
  - supervizează transferurile de informaţii şi
- - controlează activitatea generală a celorlalte componente care alcătuiesc SC

•DAR la nivel software ?

Arhitectura software a microprocesorului 8086