Arhitectura Sistemelor de Calcul – Curs 1

Computer Science & Engineering Department

Universitatea Politehnica Bucuresti
Facultatea de Automatica si Calculatoare
cs.pub.ro
curs.cs.pub.ro

Titulari de curs

- Prof. Nicolae Ţăpuş – Seria CA
  - nicolae.tapus@cs.pub.ro

- Conf. Emil Sluşanschi – Seriile CB si CC
  - emil.slusanschi@cs.pub.ro

Cuprins

3

- Introducere & Sumar al Cursului
- Bibliografie
- Motivatie
- Un pic de recapitulare (si structurare) a unor lucruri cunoscute:
  - Masina von Neumann – elemente de baza
    - UAL
    - $\mathbf{U_{Cmd}}$
    - Memoria
    - Subsistemul I/E

Notarea la ASC

4

Notarea
- 5 puncte laborator – din 5.2 posibile
- 5 puncte examen

- Laborator
  - 4.4 p = pentru 4 teme (0.9p – Python, 0.8p - C, 1.2p + 1p Cell), + 0.5p prezentarea
  - 0.8p = prezenta (0.2) activitate la laborator (IC = 0.6)
  - Bonusuri: 0.25 top prezentare + max 20% la temele 1-4
- Examen (min 2.5 lab)
  - 4p = teorie
  - 1p = problema

Un scurt sumar al cursului

5

- Intro & Recapitulare
- Nivelul structural de descriere a sitemelor numerice PMS (Processor Memory Switches)
- Ierarhia de memorii, localitatea datelor, cache
- Comutatoare ierarhice & neierarhice
- Structuri de calcul cu prelucrare paralela – taxonomii
- Concurenta vs. Paralelism
- Structura SIMD:
  - Set de instructiuni
  - Organizarea datelor
- Intercalarea perfecta si conectarea inversa

Un scurt sumar al cursului (2)

6

- Arhitectura IBM Cell Broadband Engine
- Arhitecturi MIMD – Caracteristici
- Arhitectura Cm*
  - Comunicarea intra- si inter-cluster
- Interconectarea in sisteme cu resurse multiple – permutari elementare
- Retele de comutare
  - Ierarhice
  - Delta
  - Performantele retelelor
- Stil de programare, debugging, profiling, optimizare
- Top 500, Benchmarking & the current Top 10

Bibliografie

7

- *Structura si Arhitectura Sistemelor Numerice*; T. Moisa, N. Tapus – 1999
- *Introduction to Parallel Computing: Design & Analysis of Algorithms*; V. Kumar, A. Grama, A. Gupta, G. Karypis; Addison Wesley; 2nd Edition 2003
- *The Sourcebook of Parallel Computing*; J. Dongarra, I. Foster, W. Grapp, K. Kennedy; Morgan Kaufmann 2002
- *Computer Architecture: A Quantitative Approach*; J. Hennesy, D.A. Patterson; Morgan Kaufmann; 4th Edition 2007
- *Introduction to Parallel Processing; Algorithms and Architectures*; Behrooz Parhami; Kluwer 2002
- *Techniques for Optimizing Applications: High Performance Computing*; Rajat P. Garg, Ilya Sharapov; Sun 2001.
- *Practical Computing on the Cell Broadband Engine*; Sandeep Koranne; Springer 2009.
- www.top500.org

Bibliografie vs. Cursuri

8

- *Introduction to Parallel Computing: Design & Analysis of Algorithms*
  - **Cursurile 6, 8, 11**
- *The Sourcebook of Parallel Computing*
  - **Cursurile 4, 13, 14**
- *Computer Architecture: A Quantitative Approach*
  - **Cursurile 3, 4, 7, 9, 10, 11**
- *Introduction to Parallel Processing; Algorithms and Architectures*
  - **Cursurile 6, 7, 8, 9, 10, 11**
- *Techniques for Optimizing Applications: High Performance Computing*:
  - **Cursul 4**
- *Practical Computing on the Cell Broadband Engine*
  - **Cursul 5**
- www.top500.org, http://www.netlib.org/benchmark/hpl/, http://icl.cs.utk.edu/hpcc/
  - **Cursurile 13, 14**
- *Structura si Arhitectura Sistemelor Numerice*
  - **Cursurile 2, 3, 6, 7, 8, 9, 10, 11, 12**

Motivatie

9

- “A first course in Computing for ALL Engineering Students” – Yale Patt, The University of Texas at Austin (USA)
- Ce este un computer pentru un inginer (necalculatorist)?
- Un instrument pentru a rezolva probleme (Matlab)
  - Prelucrarea algoritmilor
  - Prelucrarea numerelor
- Un procesor ce controleaza un sistem (avion, fabrica, fluxul traficului, monitor de inima)
  - Senzori/Intrari
  - Actuatori/Iesiri
  - Functii
  - Stari

Educatia inginereasca

10

- Ce trebuie sa stie un inginer?
  - Sa foloseasca unelte (HW/SW)
  - Sa proiecteze si sa dezvolte/realizeze sisteme
- Pentru aceasta trebuie sa invete:
  - Cum sunt reprezentate numerele
  - Cum functioneaza calculatoarele
  - Cum functioneaza un algoritm pe un calculator
  - Cum se ajunge de la senzori/intrari prin programe la actuatori/iesiri
- Ceea ce nu este (neaparat) necesar in (acest) curs:
  - Excel, Word, Web browsing, Invatare pe de rost

Ingineri vs. Probleme

11

- “Problem solving is programming”
- Inginerii au rezolvat mereu probleme – asta este treaba lor!
- Inginerii NU descriu problemele lor unor matematicieni, si asteapta de la acestia o ecuatie care sa specifice problema
- Se asteapta ca inginerii sa poata sa descrie singuri problemele lor in mod matematic
- In cazurile in care se apeleaza la serviciile unui matematician, trebuie sa ne asiguram ca dialogul este “coerent” si “util” – ambii utilizeaza un limbaj comun!

Ingineri vs. Programatori

12

- Problemele de astazi sunt in mare majoritate rezolvate de programe
- Putem sa incredintam rezolvarea problemei unui om care nu stie nimic despre tehnologia utilizata in rezolvarea problemei?
- Trebuie sa ne asteptam ca inginerul sa poata descrie problema in mod algoritmic?
- In cazul in care apelam la serviciile unui programator trebuie sa ne asiguram ca exista un dialog “coerent” si “util” intre inginer si programator?

Intro in Computing

13

- De ce este acesta un subiect esential pentru TOTI inginerii – si necesita mai mult decat o prezentare punctuala a unor tehnici de programare?
  - Competente de baza – fizica/analiza/electronica
    - Utilizeaza instrumentul si proiecteaza procesorul ce-l vei folosi
  - Engineering is about Design
    - Studentii pot intelege mai bine lucrurile intr-o abordare bottom-up
  - Engineering is about Tradeoffs (Totul se plateste…)
    - Sunt multe exemple in programare: iterativ vs. recursiv
  - Engineering is about State – exemple multiple in HW si SW
  - Necesitatea unei activitati creative pentru studenti
    - Studentii programeaza de la 0; depaneaza si ruleaza programe
  - Inginerii doresc sisteme deterministice
    - Totul trebuie sa functioneze “corect”

Programarea in Limbajul X

14

- De ce aceasta abordare e gresita?
  - Abordarea (traditionala) este aproape totdeauna top-down
    - Astfel se ajunge la memorare in loc de intelegere
  - Efectele memorarii:
    - Students don’t ever get it!
    - Daca nu au memorat tot, nu realizeaza greselile pe care le fac
    - Daca au memorat tot, au impresia ca forma e cea mai importanta si au impresia ca au inteles desi le lipseste intelegerea de fond…
    - Cookbook education – forma fara fond!
  - Nu ofera nici o intuitie asupra uneltelor importante
  - Nu ofera nici o intuitie in modul de interactionare intre procesor si restul sistemului
  - Nu ofera nici o intuitie asupra tradeoff-urilor si starilor sistemului

Ce este important?

15

- Design-ul trebuie sa fie Top-down
- Educatia/invatarea trebuie sa fie Bottom-up – pentru a facilita intelegerea
- Abstractizarea este vitala, dar… nu Bottom-up, ci motivata Bottom-up
- Pentru a proiecta bine un sistem, trebuie mai intai intelese componentele acestuia!
- Trebuie mereu mers de la concret catre abstract
- Trebuiesc traversate si intelese nivelele intermediare
- Memorarea NU este intelgere
- Studentii lucreaza mai bine in grupuri

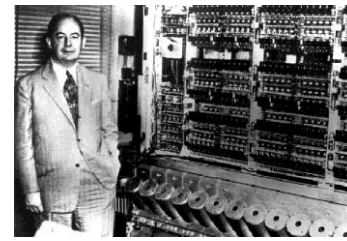
Sistem de Calcul

16

- Format din mai multe blocuri functionale:
  - Elemente de procesare – P
  - Elemente de memorare – M
  - Elemente de interconectare de tip magistrala – L
  - Unitati de comanda – K
  - Operatori de date – D
  - Switch-uri de interconectare – S
  - Terminale - T

Masina von Neumann

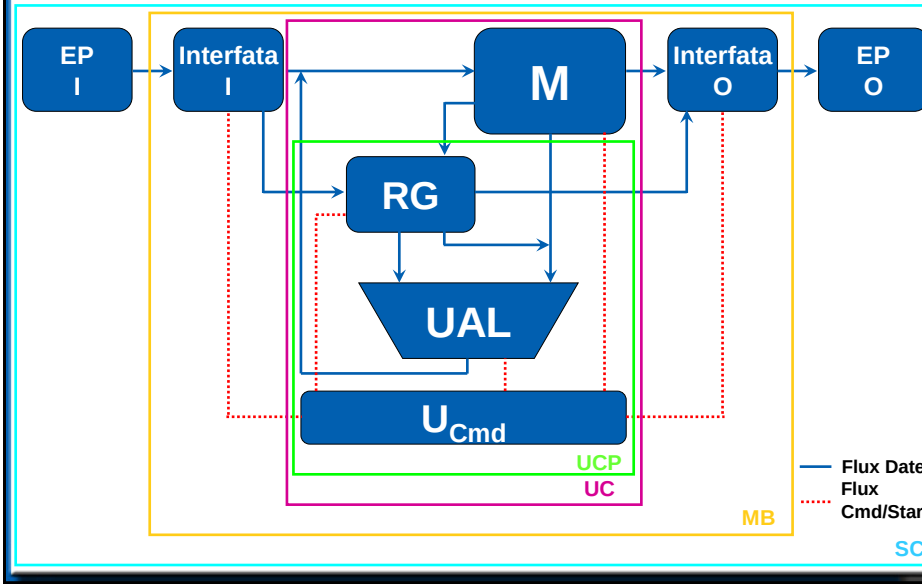
17

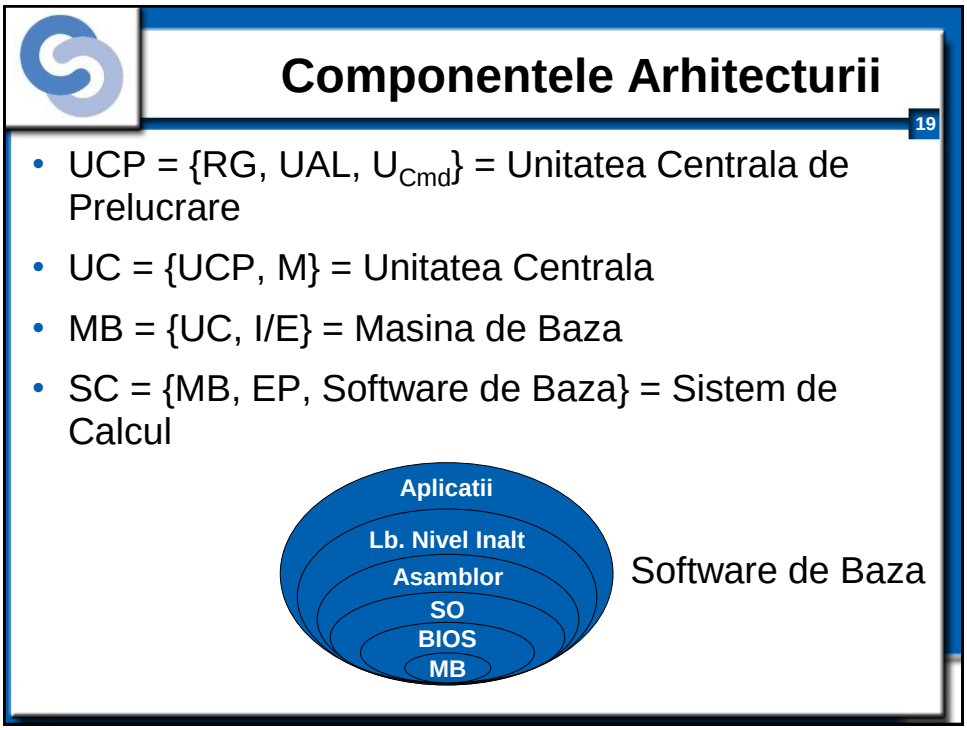


- John von Neumann (1903 – 1957)
- Structura cu acces secvential
  - CPU
  - Memorie (Instructiuni & Date)
  - *Ex: EDVAC* (1945) – binar
- *“There's no sense in being precise when you don't even know what you're talking about.”*
- *“In mathematics you don't understand things. You just get used to them.”*
- *“It would appear that we have reached the limits of what it is possible to achieve with computer technology, although one should be careful with such statements, as they tend to sound pretty silly in 5 years.” (Said in 1949)*

Arhitectura masinii von Neumann

18

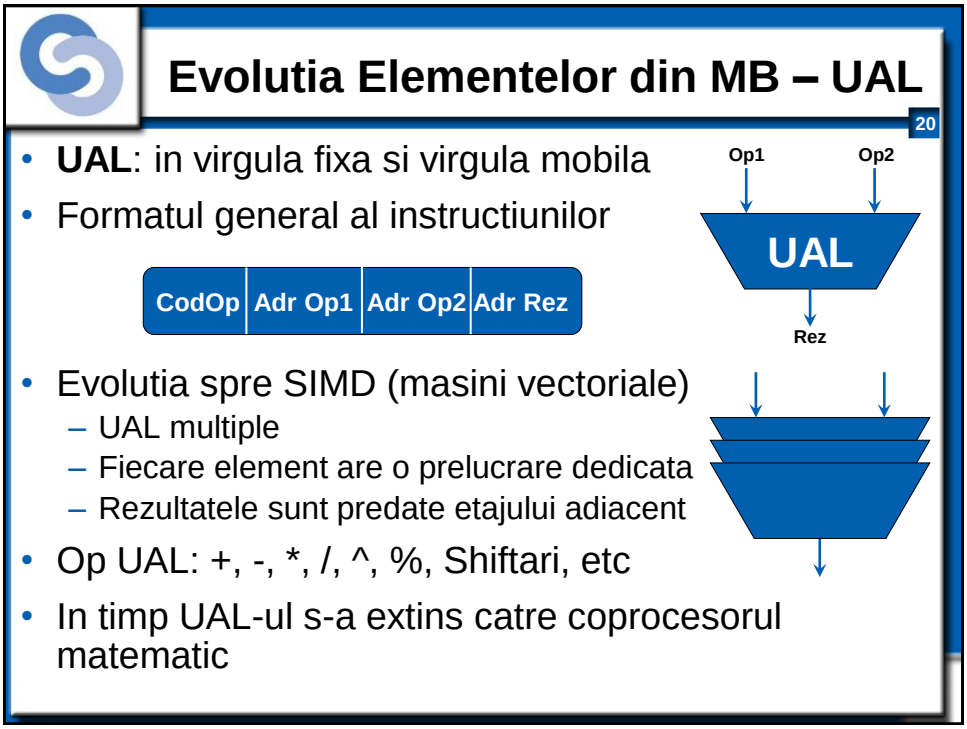




Componentele Arhitecturii

19

- UCP = {RG, UAL, U_{Cmd}} = Unitatea Centrala de Prelucrare
- UC = {UCP, M} = Unitatea Centrala
- MB = {UC, I/E} = Masina de Baza
- SC = {MB, EP, Software de Baza} = Sistem de Calcul



Evolutia Elementelor din MB – UAL

20

- **UAL**: in virgula fixa si virgula mobila
- Formatul general al instructiunilor

CodOp	Adr Op1	Adr Op2	Adr Rez

- Evolutia spre SIMD (masini vectoriale)
  - UAL multiple
  - Fiecare element are o prelucrare dedicata
  - Rezultatele sunt predate etajului adiacent
- Op UAL: +, -, *, /, ^, %, Shiftari, etc
- In timp UAL-ul s-a extins catre coprocesorul matematic

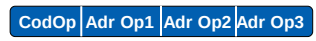
Unitatea de Comanda – U_{Cmd}

21

- Realizeaza citirea, interpretarea si executia instructiunilor masina
- Legata de:
  - Setul de instructiuni
    - Intructiuni Universale; Ciclu variabil
    - RISC cu ciclu fix
    - Complex CISC – emularea unei masini intermediare
  - Codificarea instructiunilor
    - 0 adrese 

    - 1 adresa 

    - 2 adrese 

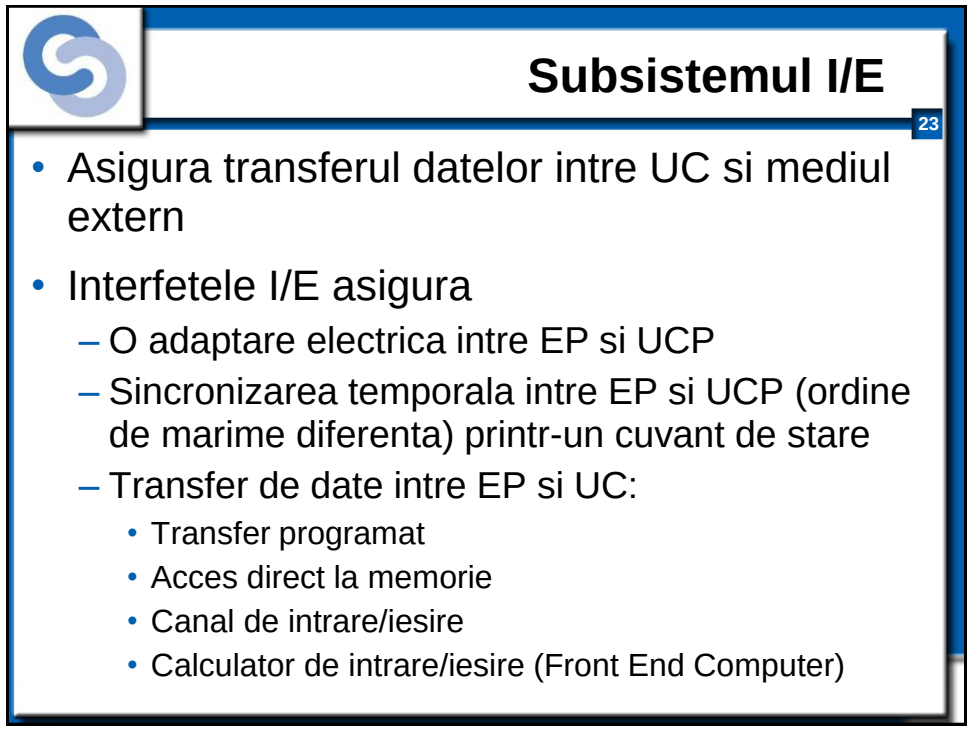
    - 3 adrese 

  - Modurile de adresare – peste 12 la ora actuala

Memoria

22

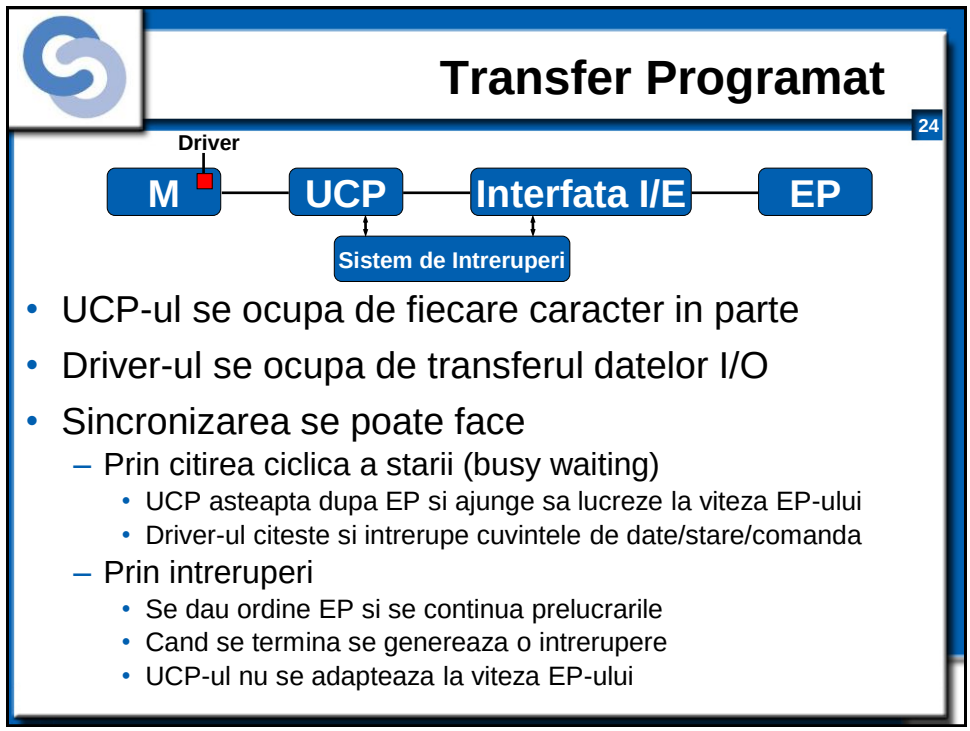
- Oraganizare fizica
  - Permanente: ROM/PROM/EPROM/EEPROM/FLASH
  - Volatile: RAM/SRAM (statice)/ DRAM (dinamice)
- Organizare logica
  - Ierarhica – pe 3 nivele
    - Rapida dar cu capacitate limitata = Cache
    - De lucru, mai lenta, cu capacitate mai mare = RAM
    - Memorii virtuale lente dar cu capacitate foarte mare
  - Functionala:
    - RAM LIFO
    - RAM FIFO



Subsistemul I/E

23

- Asigura transferul datelor intre UC si mediul extern
- Interfetele I/E asigura
  - O adaptare electrica intre EP si UCP
  - Sincronizarea temporala intre EP si UCP (ordine de marime diferenta) printr-un cuvant de stare
  - Transfer de date intre EP si UC:
    - Transfer programat
    - Acces direct la memorie
    - Canal de intrare/iesire
    - Calculator de intrare/iesire (Front End Computer)



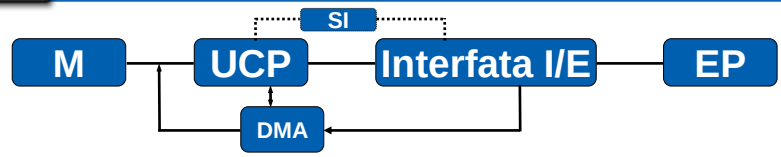
Transfer Programat

24

- UCP-ul se ocupa de fiecare caracter in parte
- Driver-ul se ocupa de transferul datelor I/O
- Sincronizarea se poate face
  - Prin citirea ciclica a starii (busy waiting)
    - UCP asteapta dupa EP si ajunge sa lucreze la viteza EP-ului
    - Driver-ul citeste si intrerupe cuvintele de date/stare/comanda
  - Prin intreruperi
    - Se dau ordine EP si se continua prelucrarile
    - Cand se termina se genereaza o intrerupere
    - UCP-ul nu se adapteaza la viteza EP-ului

Acces Direct la Memorie prin DMA

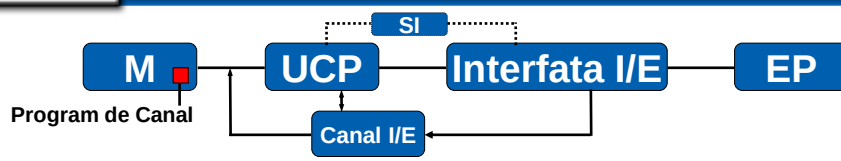
25



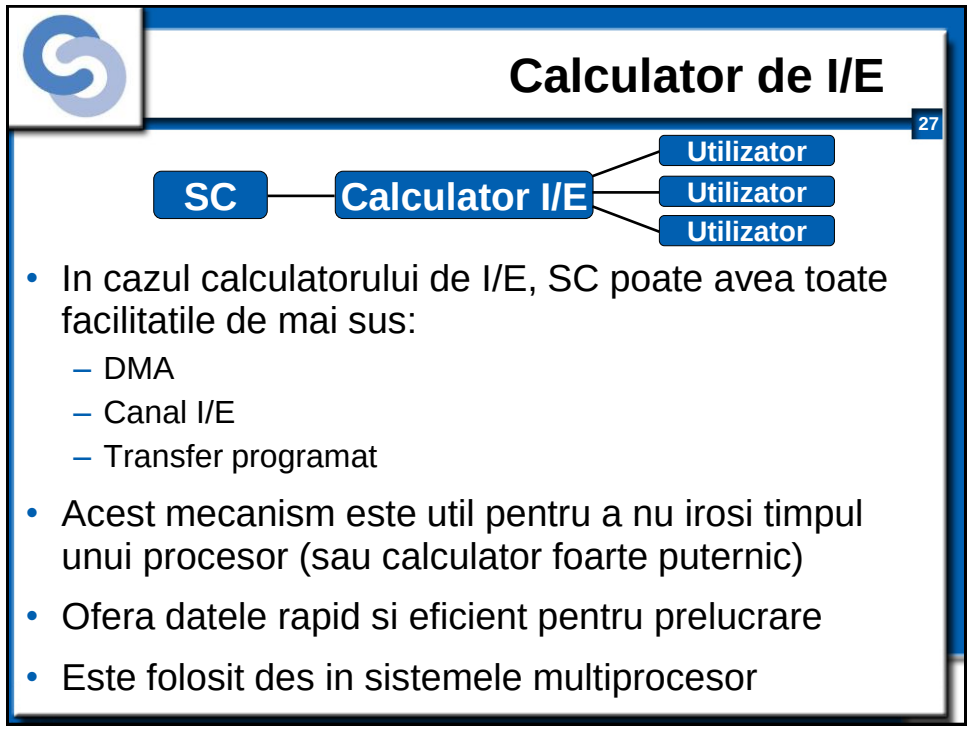
- Asigura transferul de date intre EP si M fara interventia UCP
- Dialogul intre UCP si DMA are loc doar la initiere & incheiere
- Dialogul se face prin transfer programat (Intreruperi)
- Initiere: UCP → DMA: adrese date, sensul (R/W), adresa EP
- La orice eroare DMA cheama UCP-ul
- UCP-ul nu poate initia un alt transfer pana nu s-a incheiat cel curent
- Arbitrarea conflictului de acces la M al UCP/DMA se face prin
  - Furt de ciclu
  - Rafala

Canal de I/E

26



- Canalul I/E e un procesor specializat ce asigura transferul intre EP si M fara interventia UCP
- Dialogul UCP – EP se face printr-un program de canal (din M – se da doar adresa de inceput a programului)
- UCP poate inlantui mai multe programe de canal = mai multe transferuri de blocuri de date (mai bine ca la DMA)
- Canalul aduce informatia necesara transferului in registrele proprii prin DMA (acces direct)
- Canalul reincearca transferul in caz de eroare


Calculator de I/E
27
SC
Calculator I/E
Utilizator
Utilizator
Utilizator
• In cazul calculatorului de I/E, SC poate avea toate facilitatile de mai sus:
– DMA
– Canal I/E
– Transfer programat
• Acest mecanism este util pentru a nu irosi timpul unui procesor (sau calculator foarte puternic)
• Ofera datele rapid si eficient pentru prelucrare
• Este folosit des in sistemele multiprocesor