Universitatea de Vest din Timisoara, Facultatea de Matematica si Informatica

ARHITECTURA CALCULATOARELOR

Informatica, an I, 2021-2022

Dr. Mafteiu-Scai Liviu Octavian

Universitatea de Vest din Timisoara, Facultatea de Matematica si Informatica
ARHITECTURA CALCULATOARELOR, Informatica, an I

CURS 4

STRUCTURA CALCULATORULUI SECVENTIAL

Dr. Mafteiu-Scai Liviu Octavian

Introducere

Calculatorul = o masina care prelucreaza automat informatiile, daca:

- i se furnizeaza datele de intrare;
- i se furnizeaza un program (o lista de instructiuni) ce specifica modul de prelucrare a datelor de intrare;
- la final are capacitatea de a furniza rezultatele (date de iesire)

Pentru a realiza astea, un calculator (sistem de calcul) este alcatuit din:
- **hardware:** echipamentele fizice (partea fizica)
- **software:** programe si date (partea logica)

Nota: in acest curs, focusul este pe componenta hardware

Introducere

Arhitectura calculatorului (AC) refera din punct de vedere hardware, cum anume este proiectat si ce tehnologii hardware foloseste.

Exista trei moduri de a defini AC:

1. **System Design**: toate componentele hardware din sistem;

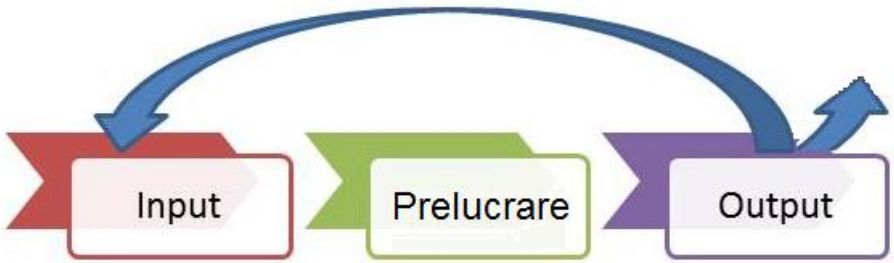
2. Instruction Set Architecture (ISA): este legat de limbajul intern al CPU si defineste functiunile si capabilitatile acestuia (*word size, processor register types, memory addressing modes, data formats* si *instructions set* folosit de catre programatori).

3. Microarchitecture: defineste magistralele pentru date, procesarea datelor si modul de stocare a acestor date.

Introducere

Componenta hardware este alcatuita din echipamentele fizice (electronice) care asigura:

- preluarea si conversia datelor de intrare in formatul intern al calculatorului (codificare);
- prelucrarea automata a datelor (cu ajutorul ……?.....)
- conversia rezultatelor intr-un format extern conform cerintelor (decodificarea).

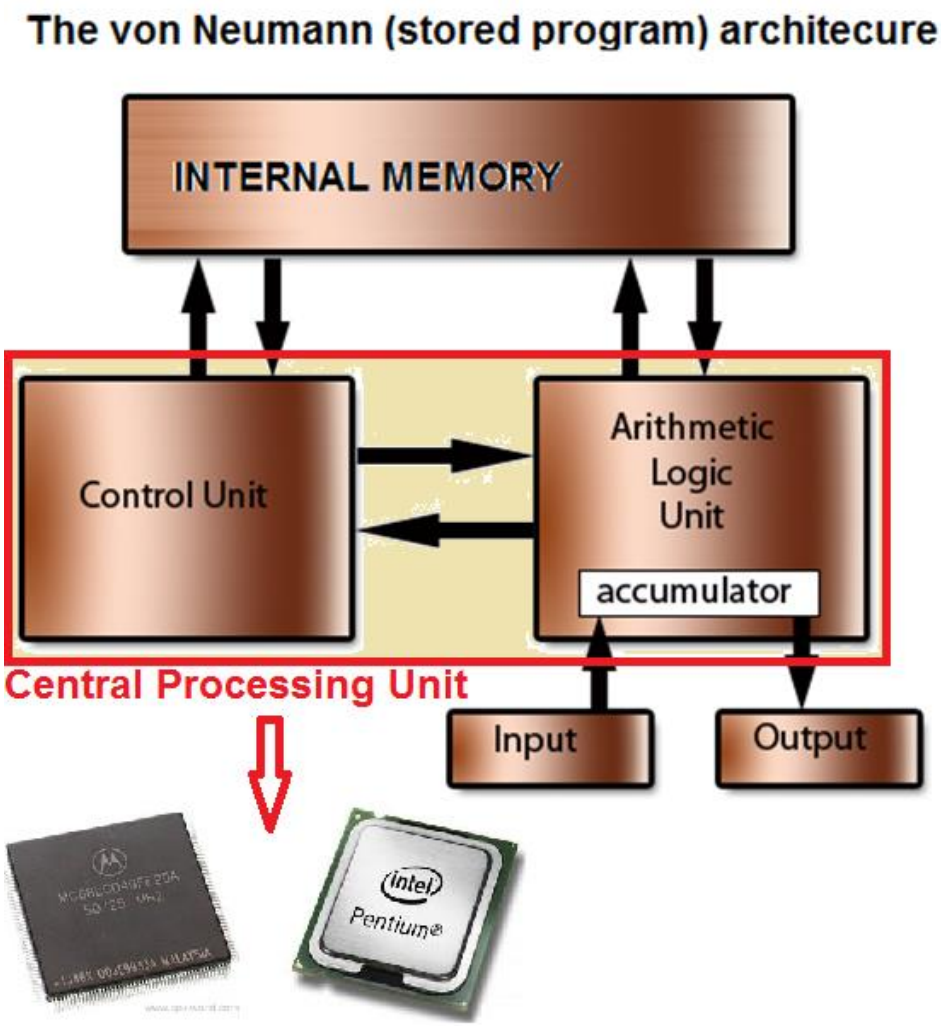


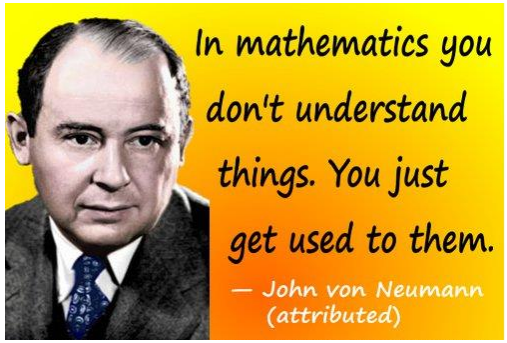
?

Va mai amintiti de conceptul de *program stocat* (stored program)?

UNDE si DE CE a aparut?

Structura von Neumann a calculatorului secvential





John von Neumann (Margittai Neumann János Lajos) (1903 Ungaria, 1957 Washinton DC US) matematician evreu de exceptie cu rezultate deosebite in fizica cuantica, matematica, analiza numerica, statistica, a si mai ales in **computer science** (arhitectura von Neumann (1945), automate celulare *(curs LF)*, teoria jocurilor etc etc)

! Zurich, student!

Programele si datele pot coexista in acelasi loc (memoria interna), adica putem avea o structura simpla si fixa a computerului astfel incat sa nu mai fie nevoie a modifica structura hardware in functie de programul dorit a fi executat (vezi ENIAC)

Structura von Neumann a calculatorului secvential

Ideile lui Neumann au devenit fundamentale pentru generatiile viitoare de computere:

- introducerea unei instructiuni masina , numita *instructiune de transfer conditional*, fapt care a permis ca o secventa de program sa poata fi intrerupta si apoi reinitiata in orice moment. Ideea apare intr-o forma primara la masina analitica a lui Babbage.

- stocarea datelor si a programelor in aceiasi unitate de memorie, astfel incat atunci cand se doreste, programele sa poata fi modificate precum datele;

- introducere concept RAM (Random Access Memory), chiar daca se foloseau cartele perforate, inlocuite ulterior de tuburi electronice, apoi de circuite integrate.

Nota: Arhitectura von Neumann (VNA - von Neumann Architecture) este uneori numita ***Princeton Architecture*** dupa numele cunoscutei universitati americane.

Structura von Neumann a calculatorului secvential

Componentele din arhitectura von Neumann trebuie sa execute urmatoarele functii : memorare, comanda si control, prelucrare si intrare-iesire.

1. *Functia de memorare:* memorarea datelor si a programelor, functie indeplinita cu ajutorul memoriei interne. In memoria interna se afla programul si datele care sunt exploatate la un moment dat.

2. *Functia de prelucrare*: care asigura efectuarea:

- operatiilor aritmetice (adunare, scadere, inmultire si impartire);
- operatiilor logice (SI logic, SAU logic si NOT (negatia)).

3. *Functia de comanda si control* :

- extragerea instructiunilor din memoria interna;
- analiza instructiunilor;
- comanda pentru executarea fiecarei operatii
- extragerea datelor de intrare din memoria interna;

…… ce se intampla intre acesti doi pasi …….??????????????????

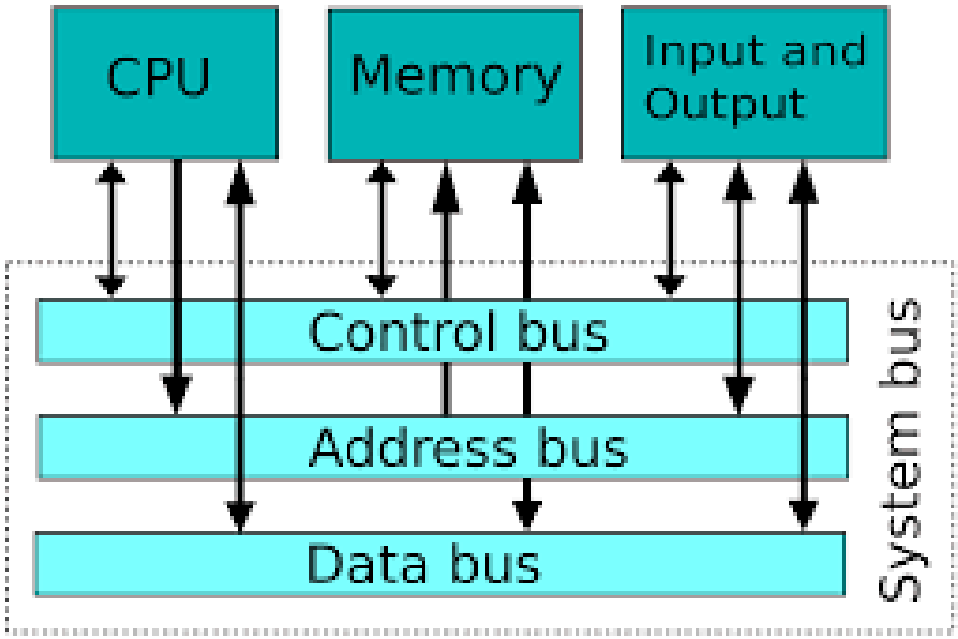
- aranjarea rezultatelor (datele de iesire) in memoria interna.

4. *Functia de intrare-iesire* care asigura introducerea datelor si a programelor in memoria interna si transmiterea datelor de iesire (rezultatele) spre exterior. Functia este asigurata de catre dispozitivele de intrare respectiv de iesire.

Structura de tip magistrala a calculatorului secvential (system bus model)

Modulele din arhitectura von Neumann sunt interconectate printr-o **magistrala** (initial un manunchi de fire conductoare) pe care circulă datele de calcul și datele de program (instrucțiuni). Totul poate fi privit ca si o magistrala cu 3 benzi, chiar daca lucrurile sunt un pic mai complicate *(nu cu mult).*

Sistemul bus imparte practic structura hardware in 3 module:



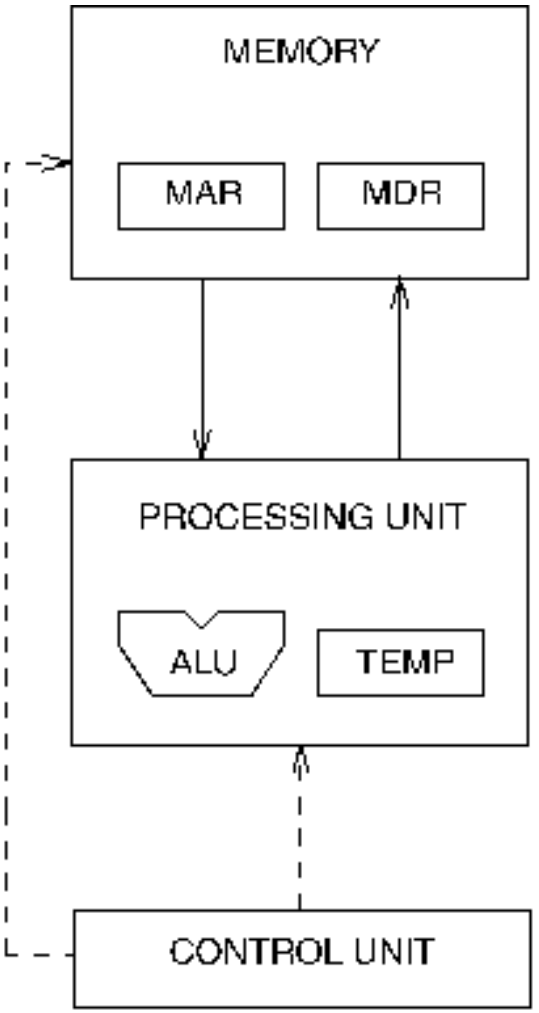
Coordonarea se face cu ajutorul tactului unui **ceas intern** (șir continuu de impulsuri).

(oarecum precum un semafor care coordoneaza traficul)

Structura von Neumann a calculatorului secvential

Comunicarea intre Memorie si Unitatea de Procesare se face cu ajutorul a doi registrii de memorie:

- Memory Address Register (MAR)
- Memory Data Register (MDR)



Structura von Neumann a calculatorului secvential

Principalele **Operatii in Memorie:**

- **Fetch** *[aducere,citire]*(address) returneaza valoarea fara a schimba valoarea stocata la aceea adresa
- **Store** *[stocare*](address, value) scrie noua valoare la adresa data.

Acest tip de memorie este **random-access**, adica **CPU** poate accesa orice valoare la orice moment de timp. Astfel de memorii se numesc **RAM** (random-access memory).

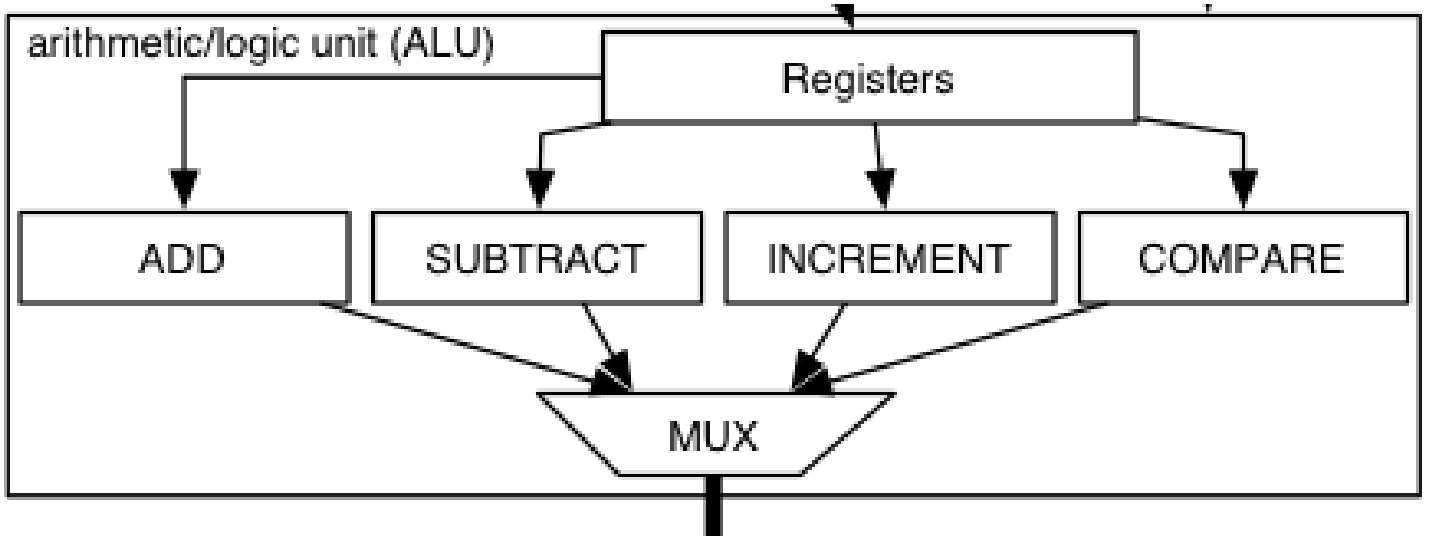
Unele dintre acestea sunt **volatile**, alte **non-volatile** (sau ROM (read-only memory).

Structura von Neumann a calculatorului secvential

ALU, unitatea de prelucrare

- **Unitatea de prelucrare** este hardware-ul care implementeaza operatiile **Aritmetice** si **Logice** (ADD, SUBTRACT, AND, OR, si NOT).
- Lungimea input-ului la nivel ALU este adesea referita ca si lungimea cuvantului computerului (in prezent procesoarele au lungimea cuvantului de 32 si 64 bit.

- "Word size" refera numarul de biti prelucrati de procesor la un singur pas



Structura von Neumann a calculatorului secvential

UCC Unitatea de comanda si control (Control Unit):

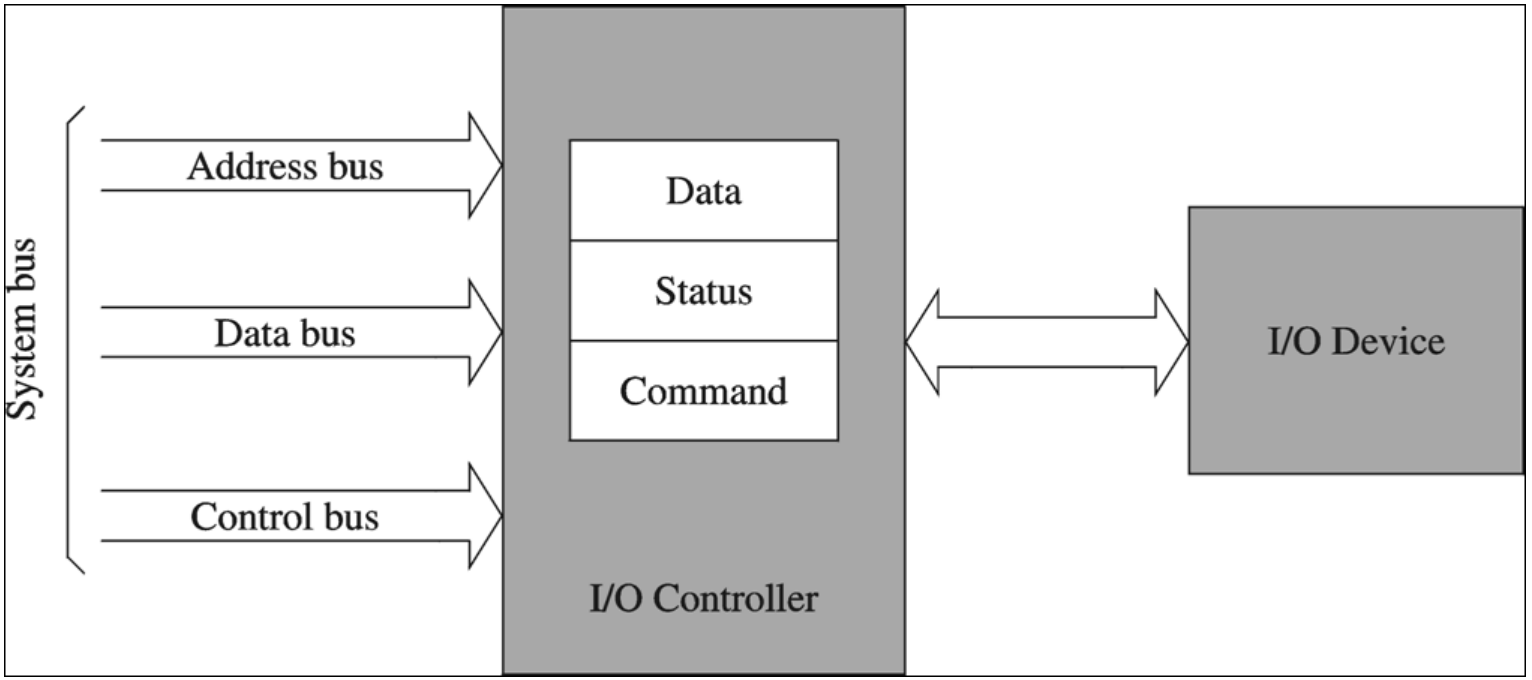
- **Coordoneaza unitatea de procesare ALU**
- **Implementat ca si FSM (finite state machine)**
- **FSM coordoneaza toate activitatile**
- **Procesarea este bazata pe un ceas intern**

Nota: finite-state machine (**FSM**) sau **finite-state automaton**, este un *model matematic* pentru calcule folosit pentru a proiecta programele de calculator precum si circuitele secventiale logice. In fact, este o masina abstracta care se poate afla intr-un numar finit de stari. *(detalii la cursul de Limbaje Formale, sem.II)*

Structura von Neumann a calculatorului secvential

Input/output

- Controlerul I/O furnizeaza interfata necesara pentru dispozitivele I/O
- Furnizeaza semnalele electrice necesare interfetelor



Structura von Neumann a calculatorului secvential

Din punct de vedere al clasificarii *Flynnschen* (Michael J. Flynn) structura *von Neumann* face parte din clasa arhitecturilor **SISD (Single Instruction, Single Data),** in contrast cu procesarea paralela.

Conform clasificarii Flynnschen avem

(clasif. include calc. secventiale si paralele):

SISD (Single Instruction, Single Data)

SIMD (Single Instruction, Multiple Data)

MISD (Multiple Instruction, Single Data)

MIMD (Multiple Instruction, Multiple Data)

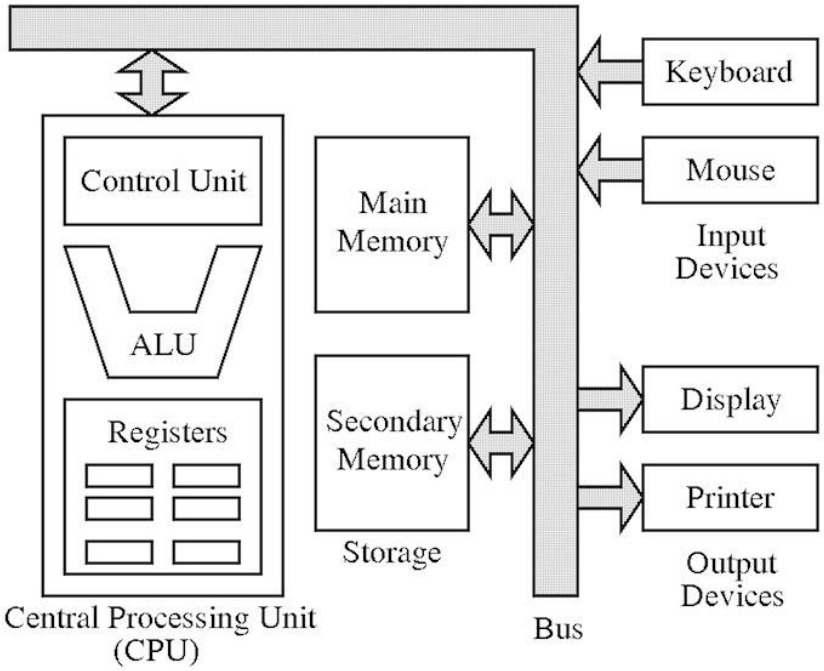
Structura von Neumann a calculatorului secvential

Clasificarea Flynnschen:

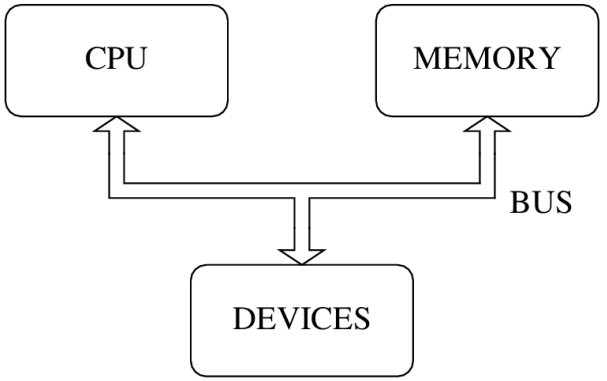
- **SISD:** Flux de instrucțiuni singular, flux de date singular - un computer secvențial (scalar) – la calculatoarele clasice cu structura von Neumann-
- **MISD :** fluxuri de instrucțiuni multiple, flux de date singular - paralelism redundant- la avioane si centrale atomice unde câteva procesoare care prelucreaza simultan aceleasi date: în cazul în care unul se defectează nu e nevoie să se întrerupă rularea - rezultate diferite, decide majoritatea !!!!!!!!!!!!!!!
- **SIMD:** Flux de instrucțiuni singular, fluxuri de date multiple – aceiasi operatie pe date diferite.
- **MIMD**: Fluxuri multiple de instrucțiuni, fluxuri multiple de date

Structura von Neumann a calculatorului secvential

Cum este evidentiata in prezent, arhitectura von Neumann:



simplificat =>



Magistrala unica folosita atat de date cat si de program este dezavantajul major al VNA, problema cunoscuta sub numele de “strangulare Neumann” ("**von Neumann bottleneck**“)

Structura von Neumann a calculatorului secvential

Avantaje:

- Nu sunt necesare magistrale suplimentare (hardware mai putin -> pret mai mic)
- Structura computerului este independenta de program

Probleme:

- Procesarea seriala a instructiunilor nu permite prelucrarea acestora in paralel (mult mai tarziu un astfel de proces a fost doar simulat)
- Date si instructiuni la un loc, pot conduce la rescrierea accidentala a instructiunilor datorita unei erori de program/programare
- Single BUS => o latență care este inevitabilă.
- "von Neumann bottleneck“: doar o informatie poate fi accesata la un moment dat

Nota: In ultimii ani:

- a crescut viteza procesoarelor
- a crescut capacitatea memoriilor mult mai mult decat rata de transfer a datelor

=>

- a crescut timpul de inactivitate a procesoarelor (idle time) asteptand ca datele sa fie preluate din memorie ⇔ nu conteaza cat de rapid este un procesor, el va avea tot mai mult timp de inactivitate

Structura von Neumann a calculatorului secvential –solutii la probleme-

- Caching : stocarea datelor mai frecvent utilizate intr-o zona speciala (cache memory).

- Prefetching: mutarea unor date in memoria cache inainte ca ele sa fi fost solicitate de catre procesor pentru a creste viteza de acces in cazul unei solicitari.
-
- Multithreading: gestionarea mai multor solicitari simultane in fire de executie (threads) separate.

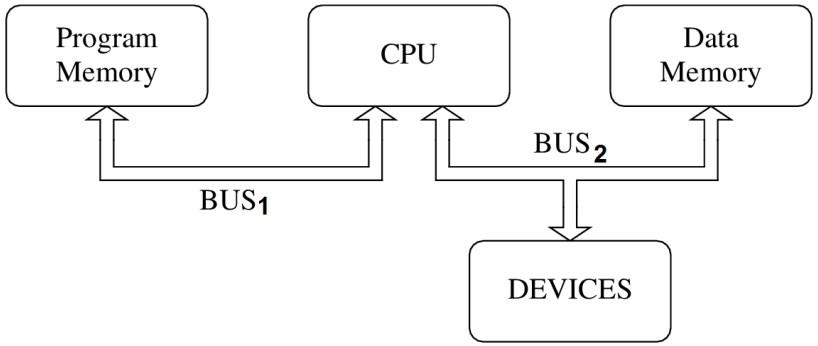
- Tipuri noi de memorie RAM (random access memory) cu performante ridicate: DDR SDRAM

- RAMBUS: un subsistem de memorie alcatuit din RAM, un controller de RAM si un nou bus care conecteaza RAM-ul la microprocesor si celelalte dispozitive

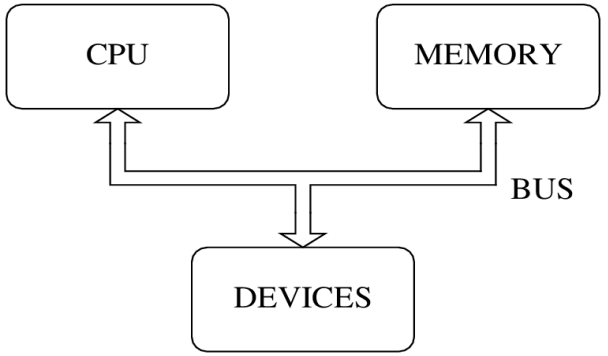
- Processing in memory (PIM): integrarea unui procesor si a memoriei RAM intr-un singur microcip.

Arhitectura Harvard

Mark II Harvard University 1947: o separare fizica intre instructiuni si date, fiecare avand propriul bus independent unul fata de celalalt.



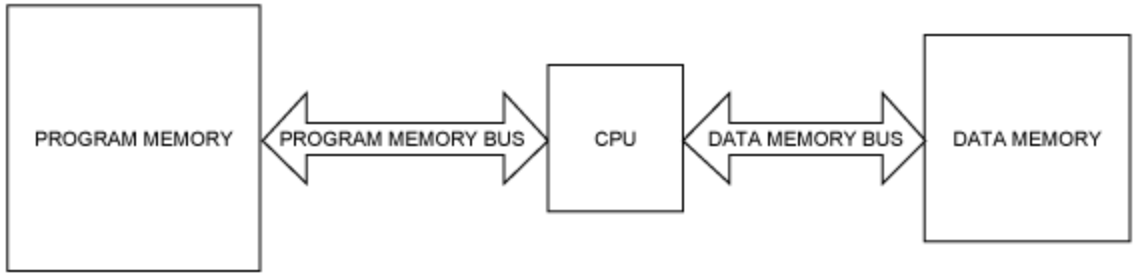
Harvard



Neumann

Avantaj major: instrucțiunile și datele se pot încărca sau pot fi scrise în același timp, potentiala “gâtuire” putand fi astfel evitata.

Potential dezavantaj: executia non-deterministica a unui program.

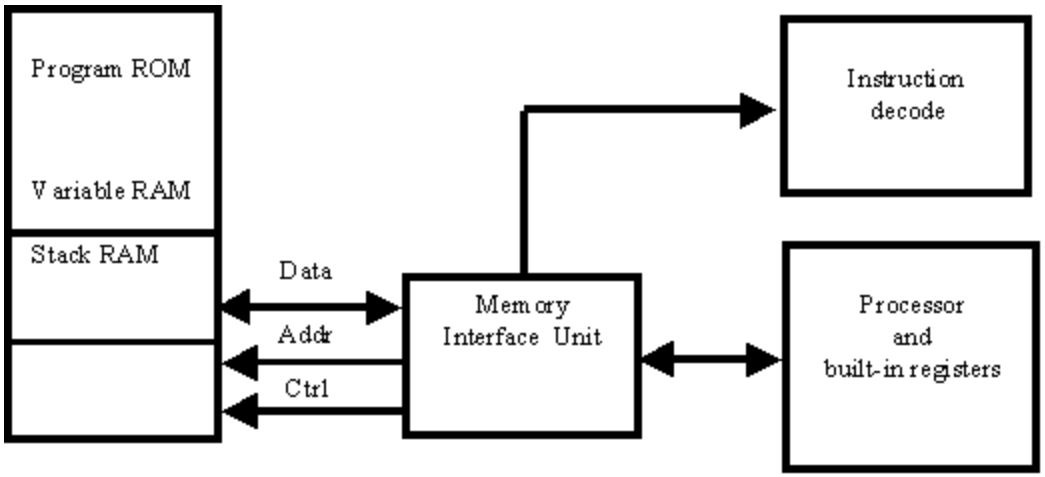


Arhitectura Harvard vs arhitectura von Neumann

Cum a aparut competitia intre cele doua arhitecturi?

La inceputul anilor “40, guvernul US a solicitat celor doua universitati (Princeton si Harvard) sa dezvolte un computer pentru efectuarea in principal a calculelor de artilerie pentru armata US.

- **raspunsul Princeton:** arhitectura von Neumann

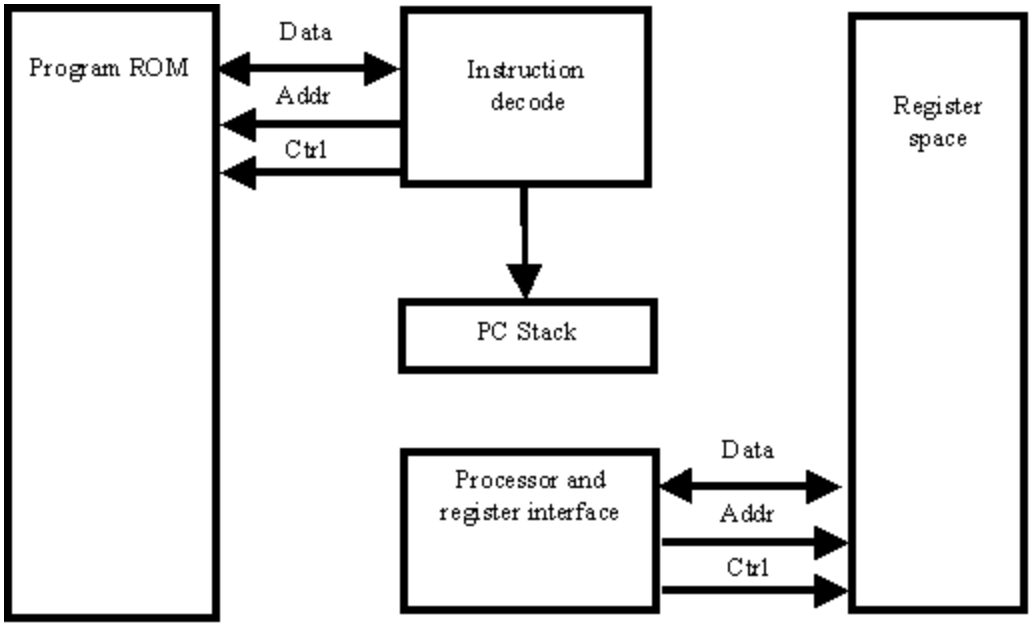


Princeton "Von Neumann" block diagram.

Arhitectura Harvard vs arhitectura von Neumann

Cum a aparut competitia intre cele doua arhitecturi?

- **raspunsul Harvard:** un proiect care folosea doua blocuri de memorie



Harvard architecture block diagram

Arhitectura Harvard vs arhitectura von Neumann

Si cum s-a sfarsit competitia sau mai bine zis

Rezultatul de moment al competitiei:

Arhitectura Princeton a castigat, deoarece era mai adecvata pentru tehnologia de la aceea data (anii 1940) si datorita faptului ca o singura memorie avea mai putine componente si drept urmare erau mai putine sanse ca ceva sa mearga prost.

Arhitectura Harvard a fost in mare ignorata pana la sfarsitul anilor 1970 cand producatorii de microcontrolere au realizat ca aceasta arhitectura are avantaje pentru dispozitivele pe care ei le proiectau la vremea aceea.

De fapt competitia nu s-a sfarsit!!!!!!!!!!!!!

Arhitectura Harvard vs arhitectura von Neumann + -uri si - -uri

von Neumann

+ programatorii organizeaza continutul memoriei si ei pot utiliza intreaga memorie instalata in computer;
+ un singur bus/magistrala este mai simplu pentru proiectarea UCC. Si mai ieftin!
+ realizarea unitatii de control este mai ieftina si ma rapida..
+ Datele si instructiunile sunt accesate in acelasi mod unitar..
- Un singur bus (pentru date, instructiuni si dispozitive) genereaza “bottleneck”.
- Erorile din programe pot rescrie instructiunile si duce la blocarea executiei acestora (programelor).

Harvard

+ Doua memorii cu doua bus-uri permit acces paralel la date si instructiuni adica executia poate fi de 2x mai rapida.
+ Ambele memorii pot fi produse folosind tehnologii diferite (Flash/EEPROM, SRAM/DRAM).
+ Ambele memorii pot utiliza dimensiuni diferite pentru blocuri (cell sizes).
+ Programele nu se mai pot “rescrie” datorita erorilor de programare..
- Unitatea de Control pentru doua bus-uri este mai complicata si mai scumpa.
- Zonele libere de memorie nu pot fi itilizate de instructiuni si vice-versa.

EGALITATE! Adica e imposibil a decide care arhitectura e mai buna. Ambele sunt inca folosite.

Arhitectura Harvard vs arhitectura von Neumann

- In ultimii ani, cateva arhitecturi diferite au fost propuse: transputers, biologically inspired computers, distributed computing, etc.
- Dar, Neumann si Harvard sunt inca folosite intens in productia de calculatoare si microcontrolere:

- Harvard a fost utilizata la inceput pentru mici computere integrate(microcontrolere) si procesarea semnalelor (DSP data signal processing)

- Von Neumann este mai potrivita pentru desktop computers, laptops, workstations

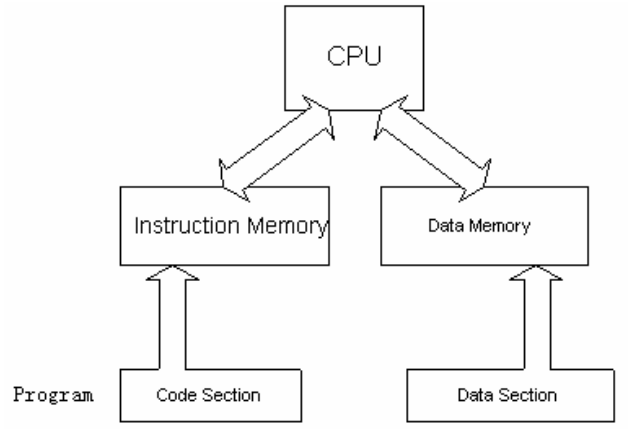
Nota: Unele computere folosec avantajele ambelor arhitecturi, adica folosesc doua memorii separate: prima este folosita de catre programe iar cea de a doua pentru date. Exemplu: telefoanele mobile.

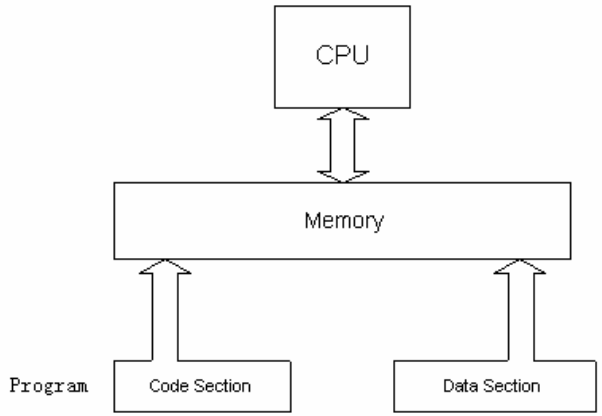
Arhitectura Harvard vs arhitectura von Neumann - la lucru-

In arhitecturile Harvard instructiunile se reprezinta pe mai multi biti => acestea suporta mai multe instructiuni cu cerinte hardware mai mici.

De exemplu, la procesorul ARM9, lungimea instructiunilor de 24-bit, ceea ce permite un numar de 2^24=16.777.216 instructiuni, ceea ce este mai mult decat in cazul unui procesor pe 16 bit (65536). Drept urmare, intr-o arhitectura Neumann ce are un singur bus, procesorul necesita mai mult hardware daca se doresc instructiuni de 24 bit lungime. In al doilea rand, accesul memoriei prin doua bus-uri ofera un castig de timp procesorului in cazul Harvard, in timp ce procesorul Neumann executa o comanda in 2 pasi (citeste instructiunea si apoi citeste datele cerute de instructiune).

Urmarea: Harvard este mult mai puternica in procesarea semnalelor digitale (DSP) deoarece astfel de dispozitive necesita un numar mare de accese la memoria de date, iar folosirea a doua bus-uri genereaza disponibilitati de timp pentru CPU.





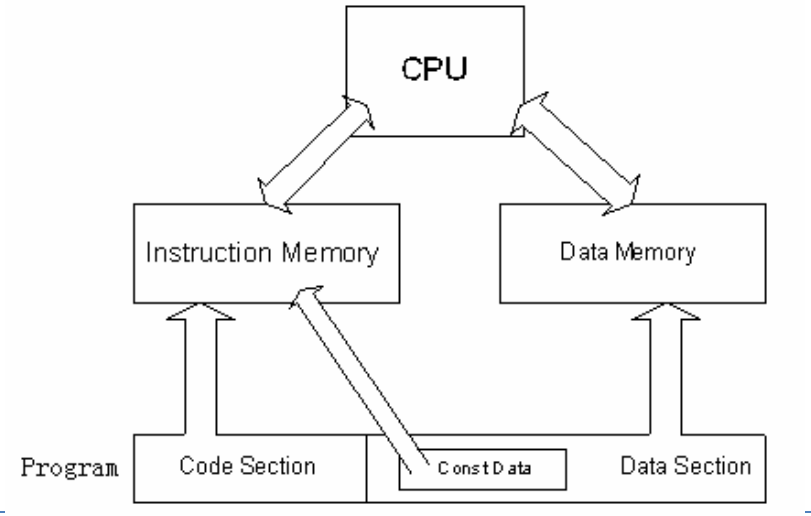
Arhitectura Harvard modificata (modified/updated)

In fact, o cale suplimentara de comunicare intre CPU si Instruction Memory (Memoria program).

Asta permite tratarea instructiunilor ca si ““read-only data”, astfel incat o data constanta precum un sir (string) poate fi citita in *Instruction memory* in loc de *Data memory*. Această metodă păstrează mai multă memorie de date (*Data memory*) pentru variabilele citite sau scrise.

Nota1: standard ANSII C nu suporta arhitectura Harvard ceea ce impune adaugarea de cod in assembler in programul C

Nota2: adaugarea de memorie cache la von Neumann diminueaza importanta arh,. Harvard



Arhitectura Harvard - programarea

Intrucat limbajul C nu a fost proiectat pentru arhitecturi Harvard, compilatoarele pentru Harvard trebuie sa aibe capacitatea de a opera cu spatii de adresa distincte.

Exemple mediu de dezvoltare: M**PLAB** (dezvoltat de catre Microchip Technology)

- In 1996, Atmel a dezvoltat o arhitectura Harvard pe 8-bit RISC, un procesor microcontroller, unele din primele microcontrolere ce au utilizat *on-chip flash memory* pentru stocarea programelor spre deosebire de celelalte microcontrolere de la acea vreme ce utilizau *One-Time Programmable ROM, EPROM,* sau *EEPROM.*

- Arhitectura ARM (Advanced RISC Machine, respectiv Acorn RISC Machine (ulterior)) este un procesor RISC pe 32 sau 64 bit dezvoltat si produs de catre ARM Limited Co., folosit intr-un numar foarte mare de sisteme integrate (embedded systems). Datorita facilitatilor de economisire a energiei (power saving features) procesoarele sunt dominante pe piata dispozitivelor mobile, unde consumul de energie electrica este un punct critic.

RISC - Reduced instruction set computing, este o strategie de proiectare a CPU bazata in principal pe un set redus de instructiuni procesor, fapt care genereaza performante sporite atunci cand este combinat cu o arhitectura de procesor capabila sa execute aceste instructiuni folosind un numar redus de cicluri procesor per secunda. Un calculator bazat pe aceasta strategie se numeste uneori *RISC computer*.

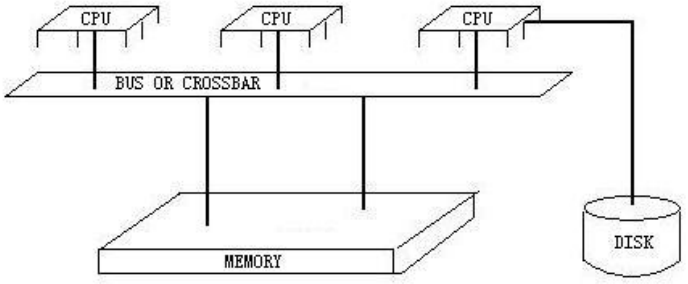
In partea opusa se afla arhitectura CISC (complex instruction set computing).

Arhitectura Harvard - caracteristici de baza ale procesoarelor ARM

- **MMU**: memory management unit (MMU), numita uneori Paged Memory Management Unit (PMMU), este o componenta hardware responsabila de accesarea memoriei solicitata de catre CPU. In fact translateaza adresele virtuale in adrese fizice, controleaza cache-ul, arbitreaza fluxul pe bus, respectiv asigura o protectie a memoriei.

- **FPU**: floating point unit este o parte a computerului proiectata special pentru a efectua operatiile cu numere in virgula mobila (Intel 286-287, 386-377 math coprocessor). In structurile moderne , FPU este integrata in CPU

- **SMP** (Symmetric multiprocessing) este o arhitectura multiprocesor in care doua sau mai multe procesoare sunt conectate la o singura memorie principala partajata. Cele mai mlte dintre procesoarele zilelor noastre folosesc arhitectura SMP. In cazul procesoarelor multi-core, arhitectura SMP aplicata la cores, le trateaza ca si procesoare separate. Cu suport din partea sistemului de operare, sistemele SMP pot usor sa comute taskurile intre procesoare pentru a eficientiza procesul de incarcare al acestora.



Arhitectura Dataflow.

Topic de interes la inceputul anilor ‘’80

-Dataflow este o arhitectura in contrast cu von Neumann care este de tip *control flow*. Dataflow nu are un contor de program (program counter) adica executia instructiunilor este exclusiv determinata de disponibilitatea datelor de intrare pentru instructiuni, adica ordinea de executie a instructiunilor este impredictibila, adica comportamentul este nederministic.

Exemplu:……..

Note:

- Chiar daca nu a avut un succes comercial in constructia de calculatoare , arhitectura a fost implementata in hardware specializat precum DSP (digital signal processing), NR (network routing), GP (graphics processing) si multe altele.
- Deasemenea, in arhitecturi software precum medii de lucru pentru calcul paralel, fapt ce constituie o provocare pentru programatori in ceea ce priveste incarcarea echilibrata a procesoarelor si sincronizarea acestora.
- Mai mult, in terminologia calculului paralel exista un subdomeniu al programarii calculatoarelor paralele numit dataflow programming.

Arhitectura Dataflow.

O *secventa* specifica care instructiune va urma a fi executata de catre procesor.

Regula de executie: o instructiune va fi executata cand cea precedenta a fost executata.

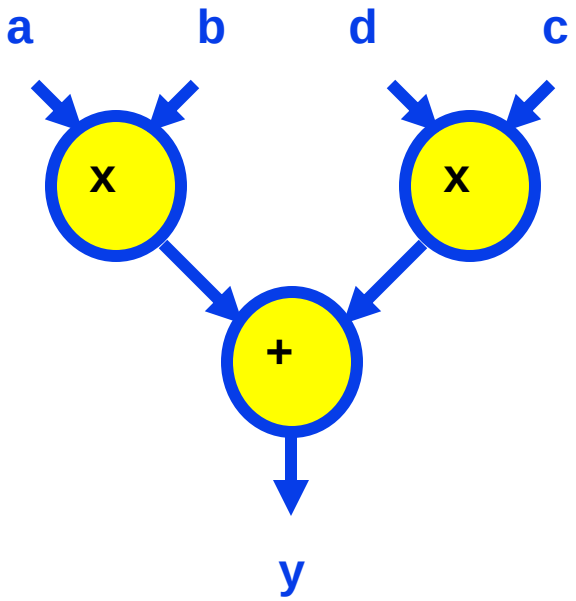
```
s1:     r = a*b;
s2:     s = c*d;
s3:     y = r + s;
```

s2 se executa
cand s1 este gata (?)
s3 se executa
cand s1 si s2 sunt gata (!)

Arhitectura Dataflow

Consideram calculul expresiei

- y = a*b + c*d ce poate fi reprezentata ca si un graf:
- Nodurile reprezinta calcule
- Datele “curg” de-a lungul arcelor;
- O instructiune este executata cand datele sunt disponibile



Data Flow - EM4

Japonia 1992 PE (EM-Y)

- **CMOS Gate Array**
- **80k gates / 1.0µ**
- **f = 20MHz**

EM-4 : o structura hibrida (flow-arh. + VNA)

Performante procesare: 1 GIPS

Performante retea: 14.63 GBps.(Bps : ?)

EM-4 was 21-times faster than the CRAY/XMP and 94-times faster than the SUN Sparc-330

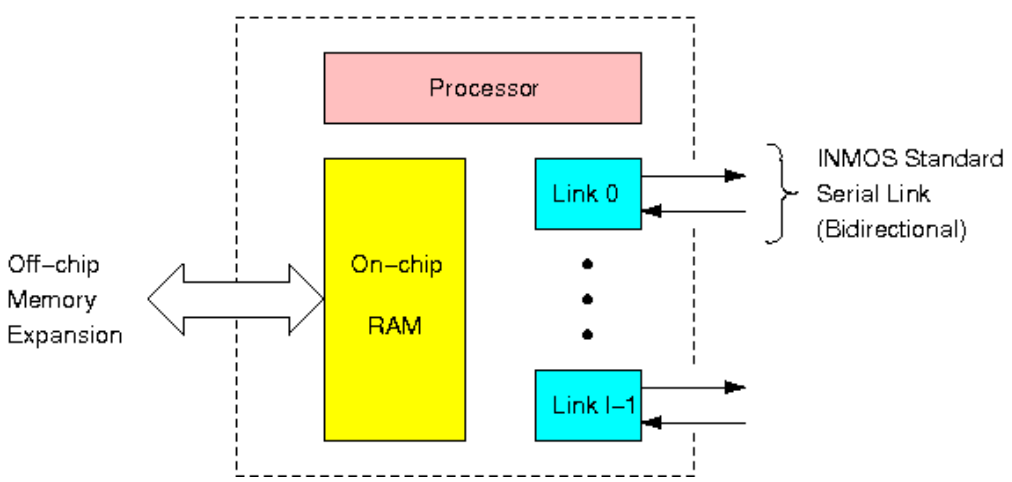
Arhitectura Transputer.

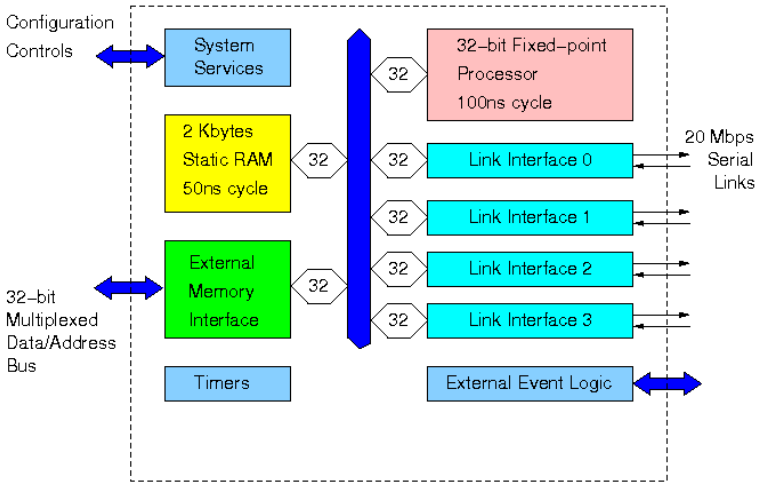
Arhitectura ***Inmos Transputer*** a fost introdusa in 1985 ca si o arhitectura de microcomputer realizata pe un singur cip, optimizat pentru calcul paralel in configuratii MIMD (Multiple Instruction, Multiple Data).

Transputer = procesor + memorie + I/O + ceas intern

- Arhitectura a dat rezultate atat in ceea ce priveste comunicatiile interprocesor cat si in ceea ce priveste capacitatile de calcul.
- Transputerele ofera capabilitatea de a implementa sisteme scalabile (ce pot fi extinse in functie de necesitatile de calcul).
- Filozofia generala a unui transputer este aceea de a furniza o familie de componente compatibile care sunt capabile sa comunice intre ele cu un minim de elemente externe, folosind legaturi de tip *point-to-point.* Acest tip de legaturi sunt implementate folosind un protocol *asynchronous bit-serial.* Fiecare transputer are un numar fix (4) de legaturi bidirectionale.

Arhitectura Transputer.





Exemplu transputer: T414

- T414 is a 32-bit microprocessor implementation of the general transputer structure.
- 2 Kbytes of on-chip RAM
- four standard INMOS full duplex, serial links.
- The external 32-bit memory interface is capable of addressing up to 4 Gbytes and has a peak data transfer rate of 25 Mbytes per second.