O INTRODUCERE ÎN FILOSOFIA MICROSISTEMELOR DE CALCUL. TRECUT, PREZENT ȘI VIITOR

SCHEMA BLOC A UNUI MICROSISTEM. ROLUL BLOCURILOR COMPONENTE, FUNCŢIONARE DE ANSAMBLU.

*"I think it's fair to say that **personal computers have become the most empowering tool we've ever created**. They're **tools of communication**, they're **tools of creativity**, and they can be **shaped by their user**."*

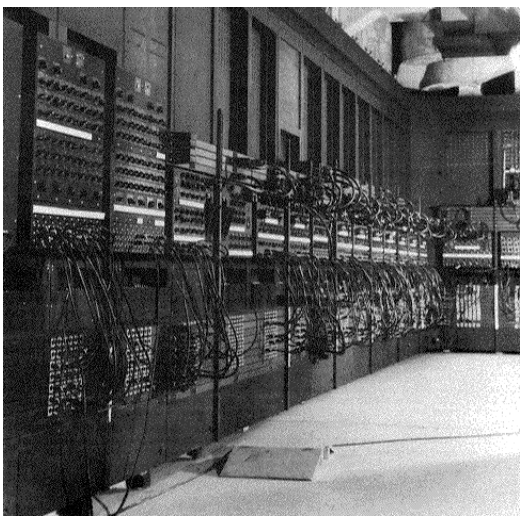
Bill Gates, February 24, 2004

*"I thought [computers] would be a **universally applicable idea, like a book** is. But **I didn't think it would develop as fast as it did**, because I didn't envision we'd be able to get as many parts on a chip as we finally got. The **transistor came along unexpectedly**. **It all happened much faster than we expected**."*

J. Presper Eckert, co-inventor of ENIAC, speaking in 1991

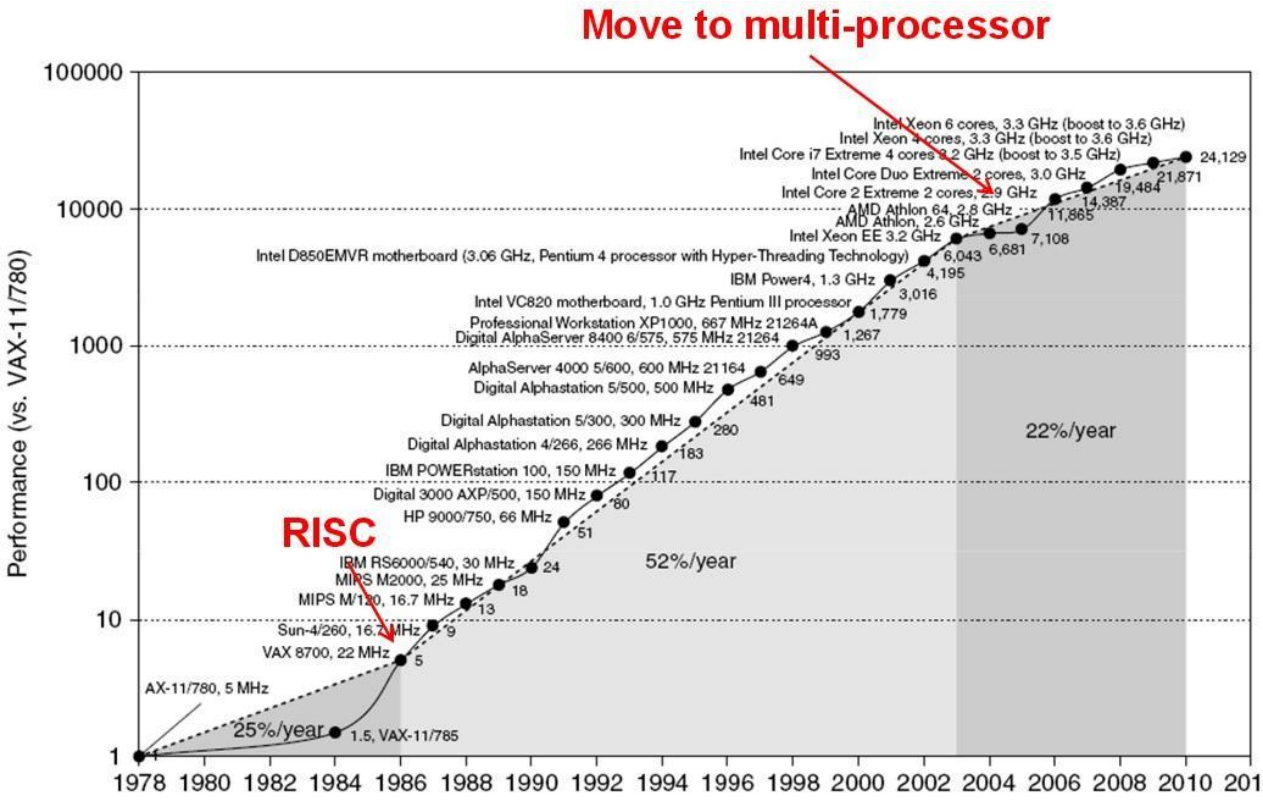
Scurt istoric:

- 12 mai 1941 Berlin, inginerul german Konrad Zuse → calculatorul Z3 (primul calculator **automat, programabil în mod flexibil şi deci universal** (citea programele de pe o bandă de celuloid perforată). Totuşi, **nu avea instrucţiuni de salt condiţionat → nici bucle de program**. Lucra cu numere în virgulă mobilă pe 22 de biţi. Conţinea cca. 2000 de relee electromagnetice şi avea o frecvenţă de tact de **5-10 Hz**. Calculatorul Z3 a fost distrus în bombardamentele din Berlin, în anul 1944. O copie a calculatorului Z3, construită ulterior chiar de către Zuse, se află la Muzeul Tehnicii din Munchen.
- 1943 - 1946, Fizicianul Dr. John Adam Presper Eckert și inginerul John Mauchly (Universitatea din Pennsylvania, Moore School) au iniţiat proiectarea și construcţia primului calculator electronic pe scară largă, de uz general, complet operaţional, numit ENIAC (Electronic Numerical Integrator and Calculator), finanţat de armata SUA, utilizat la calculul traiectoriilor balistice pe timpul celui de-al 2-lea război mondial, proiectarea bombei cu hidrogen etc. La proiect au mai participat şi alţi ingineri străluciţi. Avea cca. 18000 de tuburi electronice și 20 de registre interne, pe 10 digiţi. Avea instrucţiuni de salt condiţionat, **consuma 150 KW**! Când funcţiona, deseori cădea reţeaua electrică a unui orăşel din apropiere. Avea unităţi speciale pentru adunare, înmulţire, împărţire, extragere de radical etc. Era parţial programabil, prin switch-uri manuale, datele se citeau de pe cartele perforate IBM, iar rezultatele se tipăreau. Performanţe: 5000 de adunări pe secundă, 385 înmulţiri pe secundă etc. Primele procesări s-au realizat încă din 1944.



- 1957, ing. Victor Toma (n. 1922, viitor membru de onoare al Academiei Române) creează primul calculator electronic digital din România (numit CIFA-1, cca. 1500 de tuburi electronice şi cilindru magnetic de memorie, realizat la Institutul de Fizică al Academiei, Măgurele, lângă Bucureşti). România este a 8 a ţara din lume care construieşte un asemenea calculator şi a 3-a dintre fostele ţări socialiste, după fosta URSS (3 calculatoare) și fosta Cehoslovacia (unul).

  - În tehnologia calculatoarelor s-au realizat progrese incredibile în ultimii 76 de ani de la apariția primului calculator electronic comercial de success (ENIAC 1946).
  - În ziua de azi, cu mai puțin de 500$ se poate cumpăra un calculator mai performant, cu mai multă memorie principală (internă) și cu dispozitive de stocare (memorie externă) de o capacitate mai mare decât a sistemului de calcul care în 1985 valora 1 milion de dolari.
  - Această îmbunătăţire rapidă s-a datorat atât **progreselor pe linie tehnologică** (vezi **Legea lui Moore**) cât şi **inovaţiilor arhitecturale în proiectare**.



Creşterea performanţei procesoarelor de la sfârşitul anilor 1970

Cronologie privind creșterea performanţei microprocesoarelor:

- **1978 – 1986** creșterea performanței procesoarelor relativ la sistemul VAX 11/780 a fost de **aprox. 25%/pe an**, în mare parte datorată **progreselor tehnologice**
- **1986 – 2003** creșterea relativă a performanței a fost de **52%/pe an** și s-a datorat în principal **inovațiilor arhitecturale și ideilor de organizare / proiectare** (cercetorii susţin că cca. **65% din această creştere se datorează**

îmbunătăţirilor arhitecturale şi doar cca. 35% celor de natură tehnologică).

- **Începând cu anul 2003**, creșterea densității de putere consumată / cm^2 în cip, limitarea paralelismului la nivelul instrucțiunii, și creșterea decalajului de viteză dintre procesor și sistemul ierarhic de memorie

  (**Power Wall + Memory Wall + ILP Wall = Brick Wall**)

  au încetinit performanța sistemelor uniprocesor la cel mult **22%/pe an** conducând la apariția procesoarelor multicore (cu mai multe nuclee pe același cip).

- "Power Wall + Memory Wall + ILP Wall = Brick Wall"

  - Memory wall: the semantic gap between CPU (quick) and Memory (slow)
  - The Power Wall means faster computers get really hot
  - ILP Wall means a deeper instruction pipeline really means digging a deeper power hole

- Increasing the Performance of Instruction Level Parallelism Processors Through Predictive Methods
  - Conditional & Indirect Branch prediction
  - Instruction and Register value prediction

  ✔ **Uniprocessor performance now 2X / 5(?) yrs**

  ✔ **Sea change in chip design: multiple cores (2X processors per chip / ~ 2 years)**

  ✔ **More simpler processors are more power efficient**

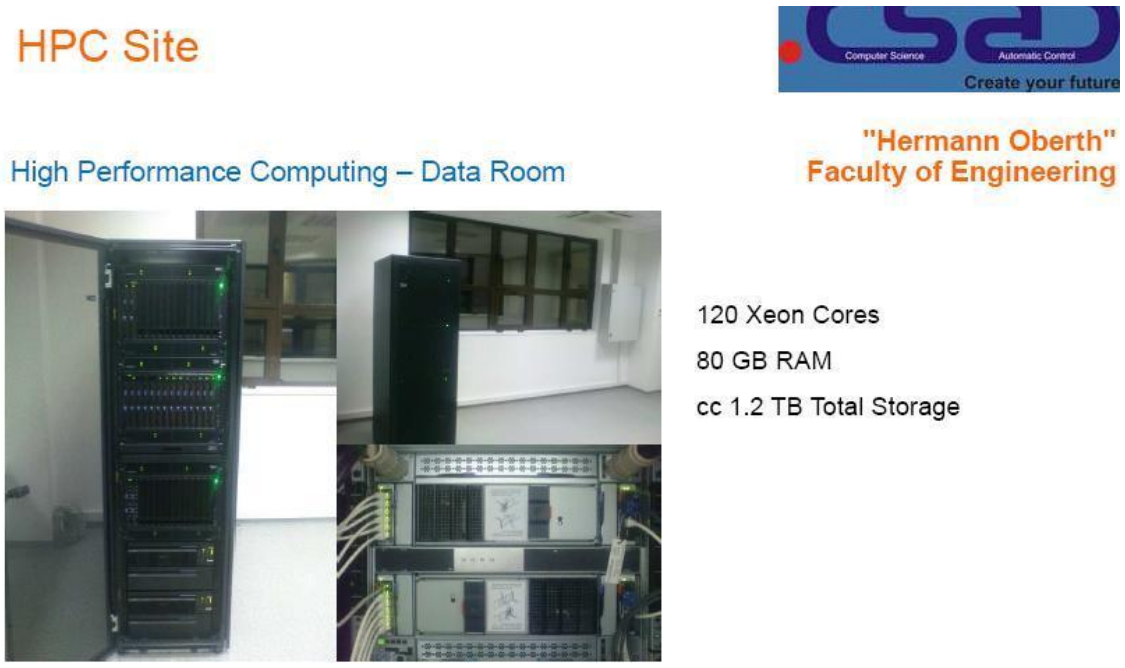
- **Calculatoarele personale** sunt cele mai **populare** şi sunt destinate utilizatorilor celor mai obişnuiţi, toate categoriile profesionale. La nivelul anului 2000 s-au produs cca. 150 milioane de desktop-uri. Exemple: Intel Pentium 4, AMD Athlon XP.

100.8 milioane de PC vândute în **prima jumătate** a anului **2005** (**creştere cu 10.5%** faţa de **2004**). *Perhaps 200 million desktop computers were sold in 2005.*
14.2 milioane de PC vândute în Japonia în 2006 (cu 3.3% mai puţine decât în 2005)
61.1 milioane de PC vândute în sfertul al doilea Q2 (aprilie-iunie) a anului 2007. Estimarea la nivelul anului **2007** era de **255.7 milioane** de PC vândute.
350 milioane de PC vândute în anul 2010.
Numărul utilizatorilor în 2003 era de **670 milioane de PC**, iar tendinţa pentru anul **2010** era de **1 miliard** de utilizatori.
Worldwide combined shipments of devices (**PCs, tablets, ultramobiles and mobile phones**) are projected to reach **2.5 billion units in 2014**, a **7.6 percent increase from 2013**, according to Gartner, Inc. Among the operating system (OS) market, **Android is on pace to surpass one billion users across all devices in 2014. By 2017, over 75 percent of Android's volumes will come** from emerging markets. http://www.gartner.com/newsroom/id/2645115 (Ianuarie 2014)

- **Serverele** sunt destinate să ofere **servicii tot mai sofisticate de reţea**, inclusiv ca noduri de Internet. Serverele absorb cca. 4 milioane de microprocesoare pe an (2000). Vânzările la nivel global de servere în primul sfert al anului 2007 (Q1) au fost de **2.1 milioane** (**creştere cu 6%** faţa de **2006**).
  - *Perhaps 10 million servers were sold in 2005.*
  - **20 milioane** de servere vândute în anul 2010.
  - Aplicabilitate:
    - în domeniul **comercial**: ***Database, transaction processing, search engines*****.** Exemple: Sun Fire 15K, IBM p690, Google Cluster.
    - în domeniul **ştiinţific: calcul ADN** (1994)**, prognoze meteorologice, analiza seismica, climaterica la nivel global, Bioinformatică, calcule şi simulări diferite necesare domeniului militar, sisteme radar si sonar, sisteme control trafic feroviar si aviatic,**

modelarea rezervoarelor de petrol. Exemple: IBM DeepBlue, IBM CELL BEA[1], BlueGene, Cray T3E.

- Caracteristicile lor cele mai importante se focalizează pe:
  - **Viteză de procesare:** *the overall performance of the server–in terms of transactions per minute or web pages served per second–is what is crucial*.
  - **Fiabilitate (**probabilitatea ca, în condiţii de mediu specificate, obiectul să funcţioneze adecvat, menţinându-şi parametrii prestabiliţi în intervalul de timp**):** ***Server downtime*** *could cost a brokerage company more than* ***$6,450,000 / hour***.
  - **Disponibilitate:** *Consider the servers running Yahoo!, taking orders for Cisco, or running auctions (*licitaţii*) on EBay. Obviously such systems must be operating seven days a week, 24 hours a day. Failure of such a server system is far more catastrophic than failure of a single desktop.*
  - **Scalabilitate:** abilitatea de a mări corespunzător capacitatea (puterea) de calcul, memoria internă / externă şi numărul de porturi de I/O (perifericele ce pot fi ataşate unui server).



- **Sistemele dedicate (*Embedded Systems*)** au **dezvoltarea cea mai dinamică**, s-au produs cca. 300 milioane (2001) respectiv 3 miliarde (2005) de astfel de sisteme pe an. Ele acoperă aplicaţiile cele mai frecvente (aparate foto şi camere video, comandă aparate electrocasnice, telefoane mobile, imprimante, comenzi auto, jocuri electronice, switch-uri pentru reţele etc.) şi au costuri cuprinse între 10$ şi 100.000$. Multe dintre aceste sisteme au **softurile scrise de producător** având un **grad de programabilitate relativ redus**. Performanţa se focalizează pe **îndeplinirea cerinţelor de timp real ale aplicaţiei**. Au **consumuri reduse de putere** şi **memorii de capacităţi relativ reduse**. Aceste sisteme sunt / vor fi integrate practic în toate dispozitivele folosite de om şi interconectate prin reţeaua globală de tip Internet, conducând la conceptul de calculator omniprezent ("***ubiquitous computing***"). *According to Sun Microsystems [Gos07], there were more than 5 billion java enabled devices (desktops, phones, cards, settop boxes, etc), from among 2.1 billions are phone handsets or PDA.*

[1] **MareNostrum Blade centers** has 31 racks dedicated to calculate. These racks have a total of **10240 processors** PowerPC 970 with a frequency of 2,3 GHz and 20TB of total memory. Each rack is formed by 6 Blade Centers. In total, each rack has a total of 336 processors and 672 Gb of memory. Each one has a rough peak performance of 3.1 Tflops.

- **A number of standard communication and data exchange protocols had been developed** and refined through widespread deployment (6 June 2012 - *World **IPv6**[2] Launch Unites Industry Leaders to Redefine the Global Internet*).
- **Digital telephony** was firmly established, and **many people were carrying lightweight, network-connected computers in the form of mobile phones**.

"Today, **MIPS Technologies (din 2013 Imagination Technologies)** powers some of the world's most popular products for the **digital entertainment**, **home networking**, **wireless**, and **portable media markets**-including **broadband devices** from **Linksys**, **DTVs and digital consumer devices from Sony**, **DVD recordable devices from Pioneer**, **digital set-top boxes from Motorola**, **network routers from Cisco**, 32-bit **microcontrollers** from **Microchip Technology** and **laser printers from Hewlett-Packard"**.

- **Dispozitive mobile cu caracter personal / multimedia** (PMD – telefoane celulare, tablete de tip IPad)
  - **1.8 miliarde** de PMD (90% fiind telefoane mobile) s-au vândut în anul 2010.

Sistemele dedicate și aplicațiile software specifice lor au început să domine modul de interacțiune dintre om și calculator (sistem de calcul) în viața de zi cu zi. Computers are no longer isolated entities sitting on our desks. Instead, they are nicely woven and integrated into our everyday lives via the gadgets we directly or indirectly use—mobile phones, washing machines, microwaves, automotive control, and flight control. Indeed, embedded systems are so pervasive, that they perform the bulk of the computation today - putting forward "embedded computing" as a new paradigm to study. Not all embedded systems are safety-critical. On one hand, there are the safety critical embedded systems such as automobiles, transportation (train) control, flight control, nuclear power plants, and medical devices. On the other hand, there are the more vanilla, or less safety-critical, embedded systems such as mobile phones, HDTV, controllers for household devices (such as washing machines, microwaves, and air conditioners), smart shirts, and so on. Irrespective of whether an embedded system is safety-critical or not, the need for integrating validation into every stage of the design flow is clearly paramount. Of course, for safety-critical embedded systems, there is need for more stringent validation—so much so that formal analysis methods, which give mathematical guarantees about functionality/timing properties of the system, may be called for at least in certain stages of the design.

Caracteristici	Personal mobile devices (PMD)	Desktop	Server	Clusters/warehouse - scale computers	Embedded
Preţul sistemului	$100 ÷ $1,000	$300 ÷ $2,500	$5,000 ÷ $10,000,000	$100,000 ÷ $200,000,000	$10 ÷ $100,000 (including network routers at the high end)
Preţul unui microprocesor component al sistemului	$10 ÷ $100	$50 ÷ $500 (per processor)	$200 ÷ $2,000 (per processor)	$50 ÷ $250 (per processor)	$0.01 ÷ $100 (per processor)
Microprocessors sold per year (estimates for 2000)		150,000,000	4,000,000		300,000,000 (32-bit and 64-bit processors only)

[2] Protocol de nivel internet TCP/IP (respectiv nivel rețea din Modelul OSI)

Critical system design issues	Cost, energy, media performance, responsiveness[3]	Price-performance, energy, graphics performance	Throughput, availability, scalability, energy	Price-performance, throughput, energy proportionality	Price, power consumption, energy, application-specific performance

Figura 1.0. *Un rezumat al principalelor cinci clase de sisteme de calcul şi caracteristicile lor*

Note the ***wide range in system price for servers and embedded systems***, which go from USB keys to network routers. *For servers, this range arises from the need for very large-scale multiprocessor systems for high-end transaction processing and Web server applications.* ***The total number of embedded processors sold in 2005 is estimated to exceed 3 billion if you include 8-bit and 16-bit microprocessors.***

More than **10 billion** embedded processor have been sold in **2008** and more than **10.75 billion** in **2009**. The total number of **embedded processors sold** was **nearly 19 billion (6.1 billion ARM-technology based chips** were shipped in 2010**).**

Înţelegerea tendinţelor tehnologice şi a celor arhitecturale.

Prin arhitectura unui microprocesor se înţelege nu numai *setul de instrucţiuni* şi *modurile de adresare* (ISA – *Instruction Set Arhitecture*) ci şi *structura organizatorică a procesorului*, respectiv *implementarea hardware*. **Arhitectura Setului de Instrucţiuni** (**ISA**) – reprezintă interfaţa dintre software (programele de aplicaţie / sistem de operare) şi hardware-ul care îl execută. ISA specifică *modul de organizare a memoriei* (zonă de date statice şi dinamice, de cod, de stivă, zonă rezervată nucleului sistemului de operare), *setul de regiştri*, setul de instrucţiuni, *formatul instrucţiunii*, *tipurile de date* utilizate şi modurile de adresare (mecanismul prin care calculatorul / procesorul localizează operanzii).

Tendinţe tehnologice:

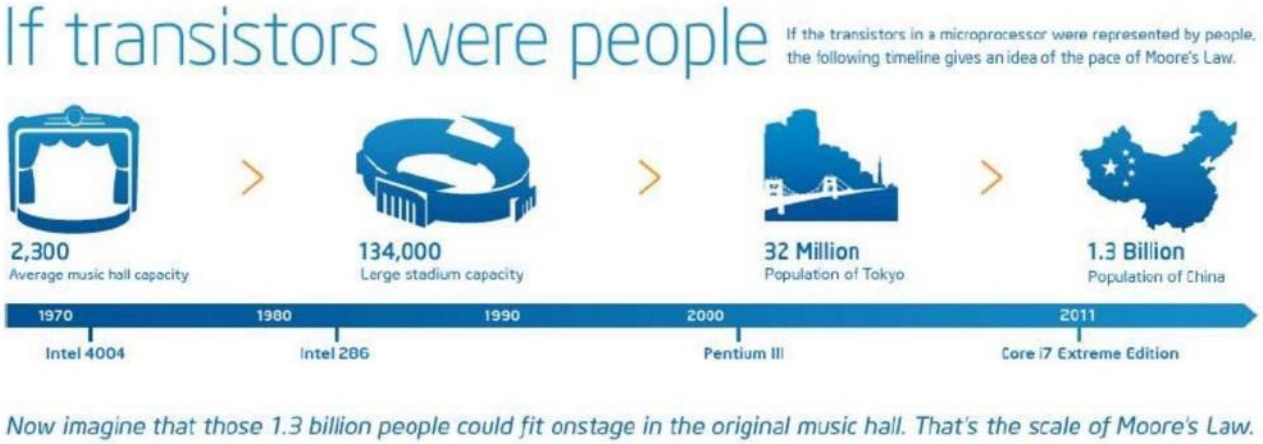
- În cazul microprocesoarelor, **gradul de integrare al tranzistorilor pe cip creşte** cu cca. 55% pe an, **dimensiunea tehnologiei reducându**-se de la 10 microni (1971) la 0,18 microni (2001), respectiv 0,032μm în 2008. Frecvenţa ceasului creşte si ea cu cca. 50% pe an. **Corelaţie „tehnologie de integrare – număr de tranzistori integraţi – frecvenţa procesor" (vezi tabelul** Procesorul Intel – tendinţe tehnologice 1971÷2008**).**

Din punct de vedere al complexităţii microprocesoarelor se poate spune că **numărul tranzistorilor integraţi într-un cip se dublează la fiecare 18 luni**. Afirmaţia aparţine lui **Gordon Moore** şi datează de la mijlocul anilor ’60. După cum se poate observa din următorul tabel afirmaţia este pe deplin justificată de implementările comerciale.

[3] receptivitate

Gordon Moore, one of the founders of Intel

- In 1965 he predicted the doubling of the number of transistors per chip every couple of years for the next ten years
- http://www.intel.com/research/silicon/mooreslaw.htm



Anul lansării pe piaţă	Denumire procesor	Număr de tranzistori încorporaţi / Tehnologia de integrare	Frecvenţa procesorului	Magistrala de date
1971	Intel 4004	2300 / 10µm	108KHz	4-biţi
1972	Intel 8008	3500 / 10µm	200KHz	8-biţi
1974	Intel 8080	6000 / 6µm	2MHz	8-biţi
1978	Intel 8086 / 8088	29000 / 3µm	4.77-10MHz	16-biţi. Implementat în primul calculator personal IBM.
1982	Intel 80286	134000 / 1.5µm	12.5-20MHz	16-biţi
1985	Intel 80386DX	275,000 / 1µm	33MHz	32-biţi
1989	Intel 80486DX	1200000 / 0.8µm	25-100MHz	32-biţi. Primul procesor care încorporează o memorie de tip cache.
1993	Intel Pentium	3100000/ 0.5µm	60-133MHz	32-biţi
1995	Intel Pentium Pro	5500000 / 0.35µm	200MHz	32-biţi
1997	Intel Pentium II	7500000 / 0.35µm	233-333MHz	32-biţi
2000	Intel Pentium III	28000000 / 0.18µm	733MHz-1.2GHz	32-biţi
2001	Intel Pentium 4 ver. Northwood	42000000 / 0.18µm	1.6GHz	32-biţi (din 2000 apare versiunea Itanium pe 64 de biţi)

2003	Intel Pentium 4 ver. Northwood Hyperthread	55000000 / 0.13µm	2 – 3.4GHz	32-biţi
2004	Intel Pentium 4 ver. Extrem Edition Hyperthread	178000000 / 0.09µm	2.8 – 3.4GHz	32-biţi
The **second half of 2007**	Intel® Core2 Penryn	400.000.000 dual-core 800.000.000 quad-core 0.045µm	3.4 – 4GHz	32-64 de biţi
Tukwila is targeted for production towards the **end of 2008**.	**Intel Itanium Tukwila / Intel® Core™ i7-980X desktop processor Extreme Edition**	2mld / 0.032µm 6 cores	4GHz	32-64 de biţi

Tabelul 1.1 Procesorul Intel – tendinţe tehnologice

Cum sunt întrebuinţaţi aceşti tranzistori?

• Mai multă funcționalitate pe un singur cip

– Începuturile anilor 1980 – microprocesoare pe 32 de biți (paralelismul la nivel de date)

– Sfârşitul anilor 1980 – Integrarea cache-urilor de nivel 1 în cipul de microprocesor

– Începutul şi mijlocul anilor 1990 – microprocesoare pe 64 de biți, superscalare (ILP)

– Sfârşitul anilor 1990 – Integrarea şi a cache-urilor de nivel 2 în cipul de microprocesor

– Începutul şi mijlocul anilor 2000 – Mai multe nuclee de procesare pe acelaşi cip, Integrarea cache-urilor de nivel 3 în cipul de microprocesor

• Ce va urma?

– Cât de mult cache se va putea integra într-un singur cip ? Memorii inteligente (procesoare cu rol de memorie) – IRAM

– Câte nuclee de procesare se vor putea integra într-un singur cip ? procesoare – Piranha, etc

– Ce s-ar mai putea integra pe cip ?

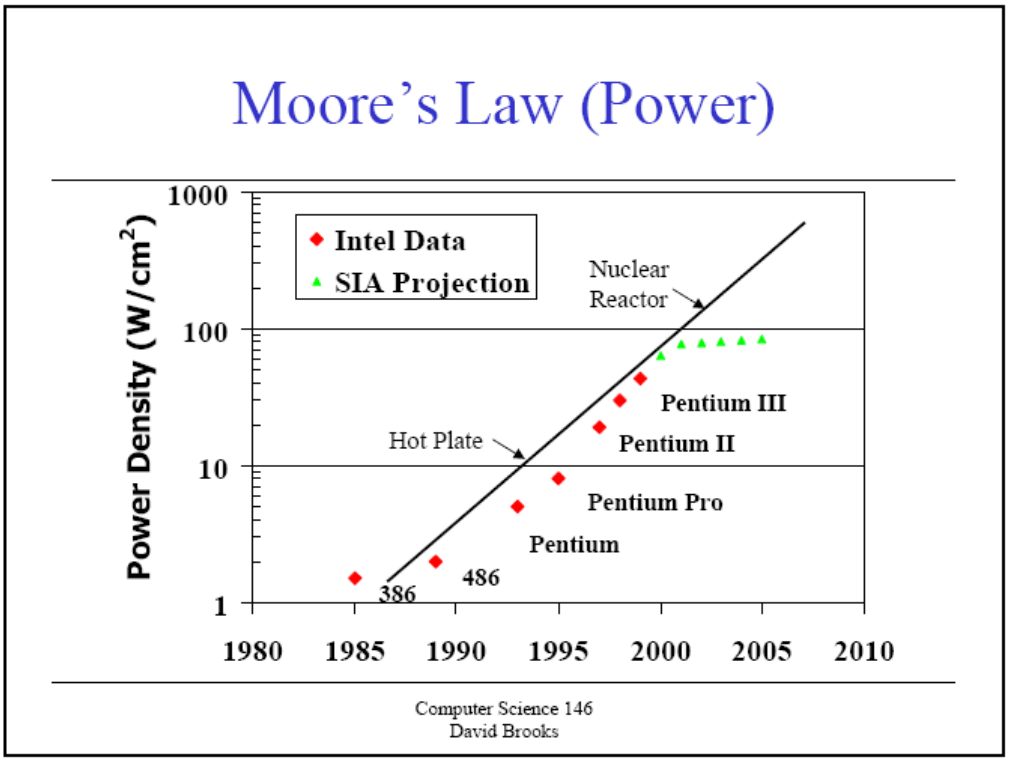
In 2020 such a CMP would contain 625 cores, and when running on the maximum possible frequency of 73 GHz would consume 7 kW, while the power budget is predicted to be 198 W.

"Can soon put more transistors on a chip than can afford to turn on!" - D. Patterson 2007

Nuclear Reactor – 200W/cm^2
Rocket Nozzle – 1000W/cm^2

Primele microprocesoare consumau zeci de watts în timp ce un Intel **Pentium 4 Extreme Edition la 3.2 GHz consumă 135 watts –** . $P_d = C \cdot V_{dd}^2 \cdot a \cdot f$

Dacă f ⇒ P_d ⇒ densitatea de putere din chip (sute de W/cm^2) ⇒ disipatie termică foarte mare (>>83°C – fierbe Siliciul) ⇒ scade durata de viață μP ⇒ cresterea frecventei de tact nu mai este posibilă, perfomanţa putând creşte doar prin inovaţii arhitecturale.



- În cazul memoriilor **DRAM**, densitatea de integrare creşte cu 40-60 % pe an, în timp ce timpul de acces aferent scade cu 33% pe decadă. **$T_{accesDRAM}$ = 5÷40ns (2008)** – vezi referinţe mai jos în document – la calculul timpului de execuţie al unei instrucţiuni.

CA:AQA Edition	Year	DRAM growth rate	Characterization of impact on DRAM capacity
1	1990	60%/year	Quadrupling every 3 years
2	1996	60%/year	Quadrupling every 3 years
3	2003	40%–60%/year	Quadrupling every 3 to 4 years
4	2007	40%/year	Doubling every 2 years
5	2011	25%–40%/year	Doubling every 2 to 3 years

Variaţia în timp a ratei de îmbunătăţire a capacităţii memoriei principale DRAM **[Hen11]**

- În cazul memoriilor ***Flash***, densitatea de integrare creşte cu 50-60 % pe an
  - Între 15-20 de ori mai ieftin preţul per bit memorat decât în cazul memorie DRAM

- În ultima perioada (> 2010) densitatea informaţiei scrise pe **hard-disc-uri** creşte cu cca. 40% pe an, în timp ce timpul de acces aferent scade cu cca. 33% pe decadă. Anterior s-au atins și rate de creștere de 100%.
  - Între 15-25 de ori mai ieftin preţul per bit memorat decât în cazul memoriei Flash
  - Între 300-500 de ori mai ieftin preţul per bit memorat decât în cazul memorie DRAM
- Costurile scad simţitor în timp (la aceeaşi putere de calcul ori capacitate de memorare).

Tendinţe arhitecturale :

- Exploatarea paralelismului la nivelul instrucţiunilor şi firelor de execuţie, atât prin tehnici statice (soft), cât şi dinamice (hard) sau chiar hibride (cazul arhitecturii IA-64, procesorul *Intel Ithanium*)

- Ierarhizare a sistemului de memorie, prin utilizarea unor arhitecturi evoluate de memorie tip cache

- Reducerea latenţei căii critice de program, inclusiv prin tehnici de reutilizare dinamică a instrucţiunilor şi predicţie a valorilor instrucţiunilor

- Utilizarea multiprocesoarelor (*shared memory*), în cadrul arhitecturilor serverelor şi staţiilor grafice. Dezvoltarea sistemelor distribuite de procesare a informaţiei (*message passing*)

- **Since 2002, processor performance improvement has dropped to about 20% per year due to the triple hurdles of maximum power dissipation of air-cooled chips, little instruction-level parallelism left to exploit efficiently, and almost unchanged memory latency**. *Indeed, in 2004 Intel canceled its high-performance uniprocessor projects and joined IBM and Sun in declaring that* ***the road to higher performance would be via multiple processors per chip rather than via faster uniprocessors***. *Whereas the compiler and hardware conspire to exploit ILP implicitly without the programmer's attention, TLP and DLP are explicitly parallel,* **requiring the programmer to write parallel code to gain performance**.

- Complexitatea din ce in ce mai greu de testat/stapanit (2Mld tranz) a condus la apariţia inclusiv pe piaţă a **arhitecturilor cu mai multe procesoare integrate pe aceeaşi pastilă** de Si. Pot fi procesoare identice (Intel Core 2 Duo, Intel, AMD dual/quad/six core) sau diferite (IBM Cell – 9 procesoare: 1 de uz general+8 specializate) **dar neexploatate încă eficient.**

"*Cell BE is the best high-performance embedded processor of 2005 because of its innovative design and future potential.*"
"*Cell could power hundreds of new apps, create a new videoprocessing industry and fuel a multibillion-dollar build out of tech hardware over ten years.*" – Forbes
"*It was originally conceived as the microprocessor to power Sony's but it is expected to find a home in lots of other broadbandconnected consumer items and in servers too.*" -- IEEE Spectrum

Tratarea complexităţii microarhitecturale prin Abstractizare

- Ca şi architect de microprocesoare scopul principal îl constituie realizarea unor **compromisuri** între următoarele: Performanţă de procesare, Putere disipată, Arie de integrare, Complexitate, Suportul oferit aplicaţiei, Funcţionalitate, Compatibilitate între modele aparţinând aceleaşi clase, Fiabilitate, etc.
- Tendinţe tehnologice şi la nivelul aplicaţiilor software dezvoltate. *Cum vor fi implementate noile arhitecturi pentru a respecta sus-amintitele compromisuri* ?

  - bazându-ne pe Abstractizare şi focalizarea pe următoarele metricile: Performanţă de procesare, Cost, Putere disipată, Disponibilitate

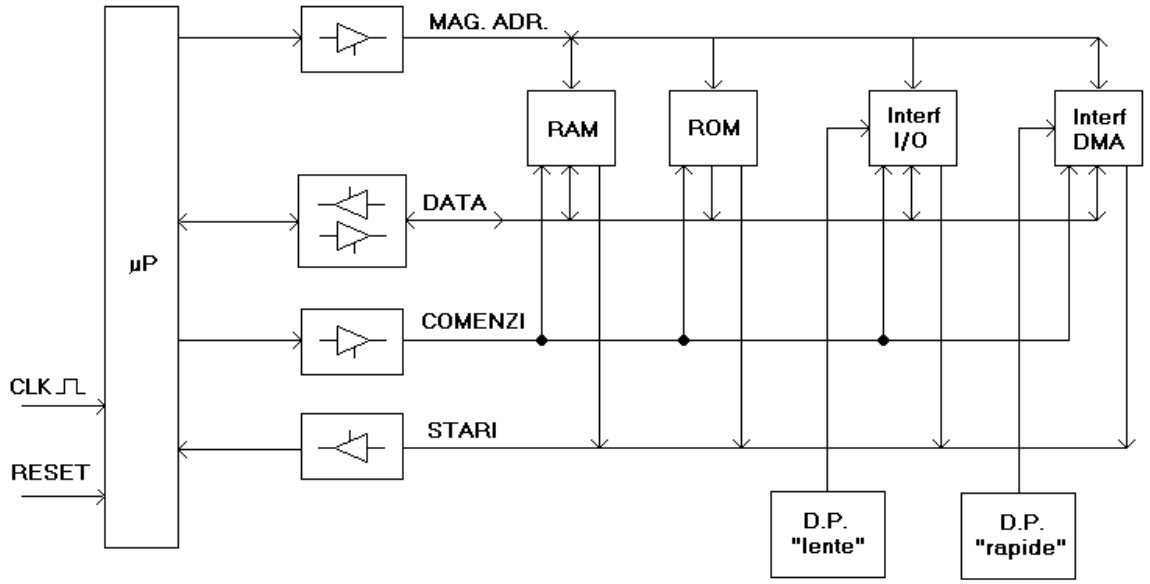


Figura 1.1. ***Schema bloc a unui microsistem*** (Microprocesor, amplificatoare de magistrale, magistrale de adrese, date comenzi şi stări, module memorie ROM şi RAM, porturi I/O lente, porturi I/O rapide – interfeţe DMA, program incarcator - POST, programe BIOS)

Microprocesorul (CPU - central processing unit) este elementul central al structurii, responsabil cu:

- aducerea din memorie, decodificarea şi execuţia instrucţiunilor maşină, codificate binar. În conformitate cu aceste instrucţiuni, microprocesorul generează secvenţiat toate semnalele (adrese, date, comenzi) necesare memoriilor şi interfeţelor pe care le gestionează.
- Dispune de o memorie proprie, foarte mică şi foarte rapidă (Regiştri – 32, 64 de uz general, PC, SP, RA, de stare / flaguri).
- Conţine unităţi de execuţie, o unitate de comandă cablată sau microprogramată, bus-uri interne de interconectare etc. În general este integrat pe un singur circuit. În sfera comercială, **primul microprocesor, pe doar 4 biţi (bus date), instrucţiuni pe 8 biţi şi un bus de adrese pe 12 biţi, s-a realizat în anul 1971 la compania *Intel*** şi a fost proiectat de către inginerul *Tedd Hoff*. S-a numit **Intel 4004**. (În domeniul militar existau pare-se mai demult asemenea circuite integrate complexe; spre ex. în comanda avioanelor militare americane F14A a existat un microsistem pe 20 de biţi, cu procesare *pipeline* a instrucţiunilor – v. http://www.microcomputerhistory.com/).
- Execută:
  - Operaţii aritmetice (add/sub/mul/div/deplasări/rotiri)
  - Operaţii logice (and/or/xor/not)
  - Teste de comparaţie / salturi condiţionate
  - Repetat anumite secvenţe de instrucţiuni
- Comandă citirea / scrierea din / în memorie, porturi I/O
- Comandă trecerea în stare nedeterminată (*tristate*) a busului de date între procesor şi memorie în cazul DMA.
- Este caracterizat de viteză mare de procesare
- Ceasul procesorului – circuit electronic ce conţine un *cristal de cuarţ*. Acesta generează impulsuri la intervale regulate stabilindu-se astfel tactul de lucru al procesorului
  - **f=100 MHz ⇒ într-o secundă (1s) sunt generate 100 de milioane de impulsuri de tact.**

Bus (magistrală) – **interconectează CPU cu memoria şi dispozitivele periferice**. Este formată dintr-o mulţime de fire paralele prin care sunt transmise adrese, date, semnale de comandă şi control (**Read / Write, INTerupt Acknowledge**), stări ale memoriei sau ale perifericelor (Cereri de întrerupere – semnalul **INT**, cereri de transfer **DMA**, **Ready / Busy** – dacă este activ semnifică faptul că Memoria / Dispozitivele de Intrare-iEşire sunt pregătite / sau nu pentru transferul de date cu CPU). **Magistrala reprezintă practic un set de reguli şi mijloace de a realiza transferul într-un sistem de calcul**. Constituie **o cale de a transporta informaţii între două dispozitive** / echipamente numite sursă şi destinaţie. Din punct de vedere al **dialogului pe magistrală** (al **coordonării transferului de informaţii**) modulele implicate se pot afla într-una din următoarele stări:

- *Master*
- *Slave*

Bus-ul de adrese este **unidirecţional de tip *tree state* (TS)**. Prin intermediul acestui bus microprocesorul pune adresa de acces la memorie sau la porturile I/O (*Input/Output*). Lumea externă a **microprocesorului** este constituită exclusiv din memorie şi interfeţele de intrare – ieşire. Acestea sunt resursele care pot fi accesate (scrise respectiv citite) de către microprocesor. Aşadar, acesta **nu "vede" în mod direct perifericele ci doar prin intermediul interfeţelor de I/O.**

Bus-ul de date este de tip **bidirecţional TS**. Prin intermediul acestui bus microprocesorul aduce din memorie instrucţiunea, respectiv citeşte data (operandul) din memorie sau dintr-un port de intrare (arhitectura *Princeton* de memorie):

- La **scriere microprocesorul plasează pe bus-ul de date rezultatul pe care doreşte să-l scrie în memorie** sau într-un port de ieşire.
- La **citire**, **rezultatul este preluat prin intermediul acestui bus din memorie** sau dintr-un port de intrare.

În ambele cazuri, **microprocesorul activează adresa respectivă pe bus-ul de adrese împreună cu semnalele de comandă aferente (*Read, Write*, Memorie, Interfaţă etc.) pe bus-ul de comenzi**. Pe **bus-ul de stări**, dispozitivele *slave* (memorii, interfeţe) comunică **informaţii referitoare la modul de desfăşurare al transferului** (Ex. semnalul "aşteaptă - **Wait**", emis spre microprocesor, cu semnificaţia că transferul de date comandat nu este încă încheiat).

Dispozitivele periferice – intermediază comunicaţia calculatorului cu mediul înconjurător. Dispozitivele periferice sunt caracterizate de: **comportament** şi respectiv **rată de transfer**. În ce priveşte comportamentul se disting următoarele clase:

a) **Dispozitive de Intrare**– tastatură, mouse, scanner, microfon, motion detector (sensor, cameră de luat vederi), interfaţă de reţea, dispozitive biometrice, interfeţe haptice, tactile.

b) **Dispozitive de Ieşire** – monitor, imprimantă, plotter, difuzor, interfaţă de reţea.

c) **Memorii externe** – cu caracter nevolatil: hard-disk, floppy disk, CD-ROM, CD-RW. *Sunt mai lente decât memoria internă însă sunt mai ieftine per bit memorat*.

Referitor la rata de transfer dispozitivele periferice sunt:

a) **Dispozitive lente** – tastatură (**100 bytes/sec**).

b) **Dispozitive rapide**[4] –disc (≥**30 MB/s**), interfaţă de reţea (**1 Mb/s - 1 Gb/s**).

[4] La ora actuală dispozitivele **USB 3.0** ating rate de transfer de **5 Gbit/s (≈640MB/s)** iar magistralele seriale bidirecționale de tip **PCI-Express 3.0** ating o viteză de transfer de 8 Gbit/s (**≈1GB/s**) [EDA11].
De exemplu, a doua generație de stick-uri **Kingston** (DataTraveler Ultimate **3.0 G2 USB Flash Drive** – care folosește un controller **USB 3.0)** având capacități de stocare de **16, 32 și 64GB**, permit o citire a datelor la o viteză de până la **100MB/s** și o scriere cu o viteză de cea până la **70 MB/s.**

Referitor la **interfaţa de programare a dispozitivelor periferice** se pun următoarele întrebări:

Q1: Cum sunt identificaţi regiştrii dispozitivelor periferice?

- mapaţi în memorie (*memory-mapped*) vs. instrucţiuni speciale.

Q2: Cum este gestionat transferul datelor între periferic şi CPU din punct de vedere al apariţiei sale în timp?

- asincron vs. sincron

Q3: Cine controlează transferul?

- Procesorul (interogare – *polling*) vs. dispozitivul periferic (întrerupere)

Q4: Tipuri de intreruperi:

- Hardware vs. software

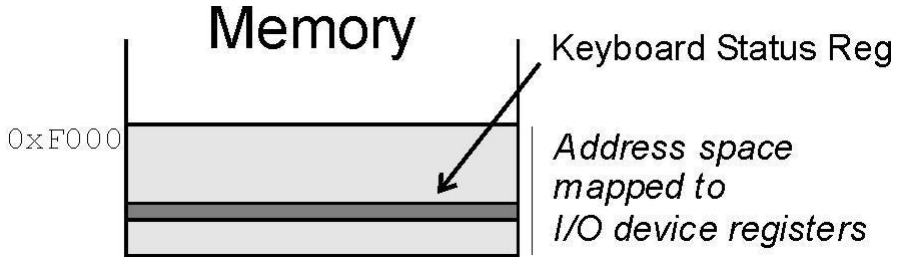
Regiştri mapaţi în memorie (*memory-mapped*) vs. instrucţiuni speciale (I/O)

Instrucţiunile speciale:

- sunt proiectate în ISA fiecărui procesor asignându-se un cod de operaţie pentru fiecare întrerupere
- regiştrii şi codul operaţiei sunt codificate în instrucţiune

Regiştri mapaţi în memorie:

- fiecare registru aferent dispozitivelor periferice (tastatură, monitor) are asignată o adresă de memorie (unică). De regulă sunt utilizate două locaţii care caracterizează starea perifericului şi data transferată spre / dinspre periferic.
- transferul datelor este controlat prin intermediul instrucţiunilor cu referire la memorie (LD/ST)



Gestionarea transferului datelor între periferic şi CPU din punct de vedere al apariţiei sale în timp

Evenimentele (cererile de întrerupere) generate de dispozitivele periferice (I/O) apar în general cu o frecvenţă mult mai mică decât frecvenţa procesorului (sporadic în timp faţă de ceea ce execută procesorul – ciclu de ciclu).

Transferul datelor se realizează în mod **sincron** dacă:

- datele sunt transmise la un interval de timp fix (viteză de transfer predictibilă / cunoscută)
- CPU citeşte/scrie la fiecare X ciclii de tact processor

Transferul datelor se realizează în mod a**sincron** dacă:

- viteză de transfer nu este predictibilă / cunoscută (nu se cunoaşte exact intervalul de timp până la viitorul transfer)
- CPU trebuie să se *sincronizeze* cu dispozitivul periferic pentru a nu pierde date sau pentru a nu le scrie prea repede (şi redundant)

Cine controlează transferul? Cine determină când apare următorul transfer de date?

În cazul **INTEROGĂRII** (*polling*) **CPU** verifică permanent starea perifericului **până când procesorul recepţionează o nouă dată** sau **până când perifericul este disponibil pentru primirea unei noi date**. *"Are we there yet? Are we there yet? Are we there yet?"*

În cazul **ÎNTRERUPERII perifericul** trimite un semnal special spre procesor (cerere de întrerupere) **când perifericul doreşte un nou transfer de date.**

Prin **întrerupere** se înţelege oprirea programului în curs de execuţie şi transferul controlului la o nouă adresă de program. La această adresă se află rutina de tratare a întreruperii, dedicată soluţionării cererii de întrerupere. Mecanismul de realizare a transferului este de tipul *apel funcţie* / *revenire*, astfel încât ultima instrucţiune din rutina de tratare trebuie să fie una de revenire (*return*) care să faciliteze întoarcerea în programul principal (cel aflat în execuţie în momentul apariţiei întreruperii), în general[5] pe prima instrucţiune de după cea pe care a apărut întreruperea.

Între două cereri succesive de transfer CPU poate executa propriile sale sarcini (instrucţiuni neimplicate în vreun transfer). "*Wake me when we get there.*"

Tipuri de întreruperi:
Din punct de vedere al generării lor, întreruperile se clasifică în **hardware** şi **software**.

A. **Întreruperile** iniţiate **hardware** apar ca răspuns la un semnal extern, fiind de două tipuri:

- **mascabile** (**ele pot fi dezactivate sau activate în mod selectiv**, prin setarea sau resetarea biţilor corespunzători din registrul IMR *Interrupt Mask Register* al interfetei (controlerului de mintrerupere). Sunt mai puţin prioritare decat întreruperile nemascabile. La microcontrolerul Philips 80C51 validarea sau invalidarea surselor de întrerupere poate fi făcută individual prin setarea sau ştergerea unui bit în registrul IE din SFR (spatiul registrilor cu functii speciale). Registrul conţine un bit de dezactivare globală IE.7, care şters, poate dezactiva toate sursele de întrerupere. **Un exemplu de intrerupere care poate fi mascabila este intreruperea de Timer**).
- **nemascabile** (sau nedezactivabile)

Întreruperile hardware sunt frecvent folosite **în calculul de timp real** din sistemele multitasking (preluarea controlului de către procesor în anumite **situaţii critice**. Pentru astfel de situatii este recomandat ca intreruperile sa fie **nemascabile**. De ex. *intreruperea la scaderea tensiunii de alimentare ACLOW, apariţia unei erori la memorie, o excepţie de operare*). Cererea de întrerupere nemascabilă este formulată pe un terminal dedicat numit în general **NMI**. Are prioritatea imediat următoare cererii DMA (vezi mai jos subcapitolul 1.2.3). Dupa terminarea instrucţiunii în curs de desfăşurare se „sare" la rutina de deservire aferenta intreruperii nemascabile care are o localizare prestabilită, definită de fabricant, răspunsul fiind unic indiferent de perifericul care solicită această întrerupere.

[5] În general revenirea din rutina de tratare a întreruperii se face pe instrucţiunea imediat următoare celei care a cauzat întreruperea. Pot însă apărea instrucţiuni de genul *Load Adresă* care să cauzeze o excepţie de tip *Page Fault*. În această situaţie va fi tratată excepţia după care se va relua execuţia programului cu aceeaşi instrucţiune de acces la memorie (*Load Adresa*).

B. **Întreruperile software** sunt de regulă implementate ca şi instrucţiuni în setul de instrucţiuni al fiecărui procesor. Se cunosc două tipuri de întreruperi software:

- **Excepţii**, deoarece întreruperea apare numai dacă există o condiţie de eroare, care nu permite execuţia corespunzătoare a unei instrucţiuni (împărţire cu zero, depăşire de domeniu, etc). Un tip special de excepţie îl reprezintă *excepţia de depanare*, care permite execuţia unui program instrucţiune cu instrucţiune („*pas cu pas"*).
- **Întreruperi generate la fiecare execuţie a instrucţiunii TRAP *n*** (unde $n \in (0 \div 255)$).

Din punct de vedere al sincronizării cu ceasul procesorului, **întreruperile *software* sunt evenimente sincrone**, reprezentând răspunsuri ale procesorului la anumite evenimente detectate în timpul execuţiei unei instrucţiuni, în timp ce **întreruperile *hardware* sunt evenimente asincrone**, fiind generate de dispozitive externe. Întreruperile software sunt întotdeauna reproductibile prin reexecuţia programului în aceleaşi condiţii de intrare, în timp ce întreruperile hardware sunt de obicei independente de execuţia procesului curent.

Tipuri de interfețe (porturi)

1. Porturi (interfețe) paralele

- Facilitează transferul de date între memoria microprocesorului sau a microcontrolerului (calculator implementat pe un singur cip: conține CPU – *Central Processing Unit*, memorie și interfețe I/O; este destinat controlului în timp real al unor procese) și dispozitivele periferice, prin intermediul unor magistrale paralele (**transfer simultan al mai multor biți de date**). Sens transfer: *input* sau *output.*
- Transferul se face de obicei prin intermediul unor protocoale asincrone de tip *hand-shake* (întrebare-răspuns similar modului de lucru prin interogare).

- Exemplu generic protocol *hand-shake*:
  - Activare Adresă Port In (de către microprocesor-MP),
  - Activare Comandă Read – tot de la microprocesor (MP) și eventuale Stări Wait,
  - Activare Read_Ack – de la interfață (periferic) + Activare și Date de la interfață,
  - Citire date de către MP,
  - Dezactivare Comandă Read de către MP,
  - Dezactivare Read_Ack de către interfață,
  - Dezactivare Bus Date (interfață), Dezactivare Adresă Port In (MP).
- O magistrală este caracterizată de protocolul logic de transfer aferent (scriere/citire/întreruperi/DMA etc.)

2. Porturi seriale

- Transfer serial asincron (*Universal Asynchronous Receiver Transmitter* – UART)
- START (1 bit activ pe 0 logic), DATE (5-8 biți), PARITATE (pară/impară, calculată prin SAU EXCLUSIV), STOP (1-2 biți activi pe 1).
- Dacă paritatea emisă este diferită de cea calculată la receptor, se consider că datele recepționate sunt alterate.
- Este necesar ca durata unui bit să fie aceeași la Emițător și la Receptor (exemplu 600, 1200, 2400, 4800, 9600 etc. biți pe secundă)
- **Emisie**: buffer emisie gol (empty) → se poate înscrie un nou cuvânt de date (paralel), care va fi serializat de către UART
- **Recepție**: buffer recepție plin (full) → trebuie citit cuvântul asamblat de UART din buffer (conține doar datele)
- UART – cuvintele se transmit-recepționează asincron, chiar dacă biții din cadrul unui cuvant sunt, evident, sincroni.

- Există și interfețe seriale sincrone, în care cuvintele se transmit / recepționează în mod sincron, prin protocoale specifice (nivelul fizic de rețea de calculatoare). Avantajul față de cele asincrone este dat de eficiența transferului, pentru că informația de control nu este la nivelul fiecărui octet în parte, ci la nivelul unui cadru conținând mai mulți octeți (sincroni).

3. Timere
- Oferă **funcţii de timp (real)**. Practic, este **obligatoriu la microcontroler** (MC).
- Măsoară timpul (exemplu bucle de întârziere) și generează semnale de diferite frecvențe / durate (spre exemplu, pe post de întreruperi de timp real – comutare între aplicații, startare secvență periodică de program etc.)
- Spre exemplu, **programatorul setează un registru timer cu o anumită valoare. Acesta este decrementat cu un semnal de frecvență cunoscută. Când conținutul acestui registru este zero, se generează o întrerupere** etc.
- Alt exemplu: măsoară perioada unui semnal (nr. impulsuri de timer între două fronturi crescătoare succesive ale semnalului).
- Funcție *watchdog* – un numărător intern al MC, care este activat prin software. Când ajunge la valoarea maximă resetează sistemul. Ca să nu se întâmple asta, programatorul trebuie să-l încarce înainte de resetare. Util în depanare (detecție eroare de program).

4. Module PWM (*Pulse Width Modulation*)
- Folosite la comanda motoarelor de curent continuu, comanda surselor de alimentare etc.
- Generat periodic, fără intervenția CPU. Perioada (T) și factorul de umplere se pot modifica în mod controlat, prin software.

5. Module conversie A/D și D/A
- Practic **obligatorii la microcontroler** (MC).
- Timpi de conversie între 8-20 microsecunde. Semnalul analogic trebuie menținut constant pe durata conversiei (circuite de eșantionare/memorare).
- Rezoluții 8-12 biți. Tensiunea de referință (valoarea maximă convertită), GND (masă).
- Declanșarea și terminarea conversiei sunt semnalizate prin biți de control. Rezultatul conversiei este memorat într-un registru de date. Se pot genera întreruperi la finele procesului.
- Declanșarea conversiei poate fi internă (prin soft) sau externă (exemplu prin timer)

6. Controler de întreruperi
- Arbitrează cererile (priorități fixe – PF, rotitoare – *Round Robin*)
- Generează INTACK și vectori de întrerupere pentru fiecare nivel de cerere (*Interrupt Request Level*)
- Poate masca anumite cereri. Demascarea se face prin software. Spre exemplu, în modul de lucru cu PF, întreruperea de un anumit nivel poate bloca toate cererile mai puțin prioritare → la finele rutinei de tratare a întreruperii (RTI), trebuie demascate (prin scrierea într-un registru special al controlerului).
- În general, RTI aferentă unei întreruperi poate fi întreruptă de o cerere mai prioritară. A nu se uita însă demascarea întreruperii mai puțin prioritare la finele rutinei celei mai prioritare.

Memoria internă – realizează stocarea programelor şi datelor necesare acestora, de capacitate mai mare decât regiştrii procesorului dar mai lentă decât aceştia.
- Unitatea de măsură elementară a memoriei este **bit**ul (***Bi**nary Digi**t***). Definiţia bitului: *probabilitatea unui element cu două stări să ia una dintre ele*. Un bit[6] este cantitatea de

[6] Un computer obişnuit stochează un bit de informaţie în aproximativ 1 milion de atomi. Cercetătorii IBM au reuşit să stocheze un bit în doar 12 atomi! Avantaj – reducerea dimensiunii circuitelor şi proprietăţi superioare circuitelor din

informaţie necesară reducerii la jumatăte a incertitudinii. O succesiune de biţi din memoria internă poate corespunde unei instrucţiuni, unei date reprezentând valori numerice, text, diverse coduri (imagini, sunet, text).

- Este organizată în diviziuni (locaţii sau cuvinte) de o mărime egală cu cea a numărului de biţi ce poate fi procesată simultan de procesor. O **locaţie de memorie** este caracterizată de **adresă** şi de **conţinut**. Un cuvânt de adresă pe *m* biţi poate indexa un spaţiu de 2^m locaţii de memorie.
- Poate fi statică sau dinamică caz în care un condensator va reprezenta celula iniţială de memorare, necesitând reîncărcarea sa periodică (regenerare).

Memoria poate fi văzută intr-o primă abordare ca o **stivă de locaţii binare** (cuvinte), fiecare cuvânt fiind caracterizat de o **adresă binară unică**.

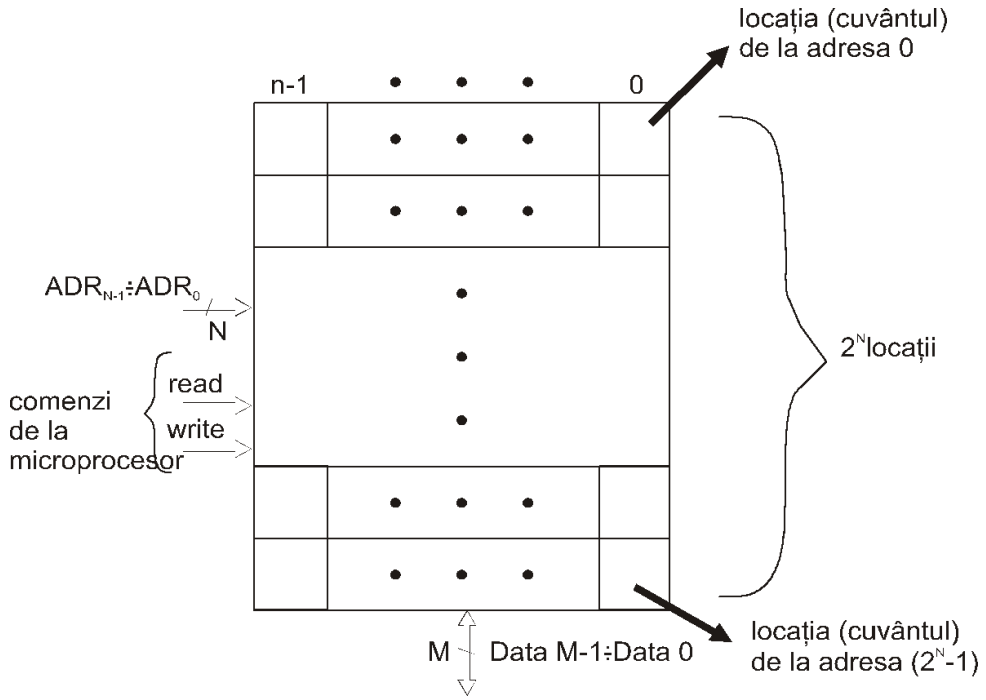


Figura 1.2. Schemă generică de memorie

<u>În general M=8,16,32,64<=> **largime bus date microprocesor** (microsistem pe M biti)</u>

Memoria este caracterizatã prin **2 parametri de bazã**:

- **capacitatea** (nr. de locaţii pe care le conţine)
- latenţa (**timpul de acces**) care este o caracteristicã intrinsecã a circuitului de memorie si reprezintã in principiu timpul scurs din momentul furnizării adresei de către microprocesor până in momentul in care memoria a încheiat operaţia comandată (citire sau scriere).

Fireste, se *doresc* **capacităţi cât mai mari** si **latente cât mai mici**, cerinţe în general contradictorii.

Siliciu. Dezavantaj: Dezvoltat prin magnetizare (la temperaturi apropiate de 0 absolut), o radiaţie (o schimbare de temperatură) poate transforma un 0 în 1 şi invers **pierderea informaţiei** (de exemplu: durata de viaţă a unui disk magnetic este de 100 de ani, cea a unui stick USB de aproximativ 7-10 ani).

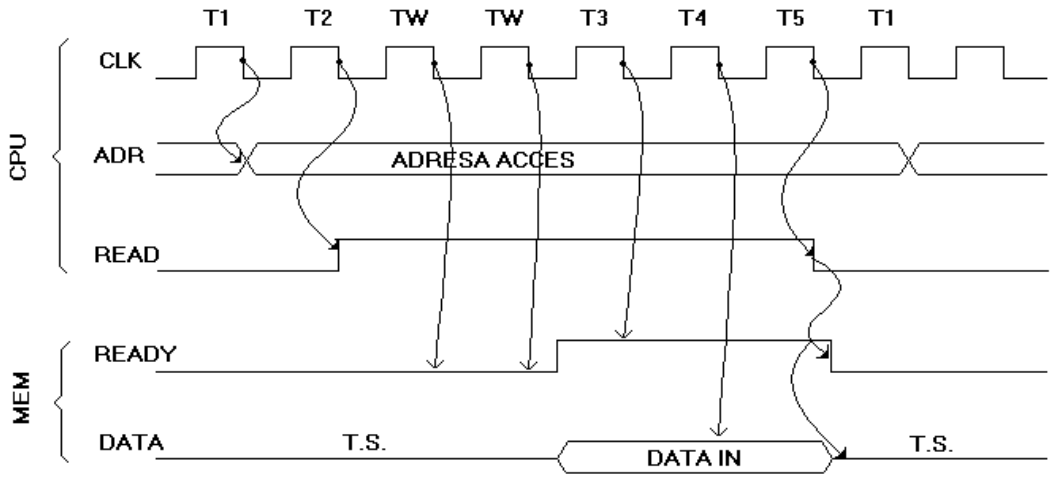


Figura 1.3. Un ciclu (fază) extern de citire din memorie

Între busul de adrese si memorie exista un **<u>decodificator</u>** N:2^N ca în figură:

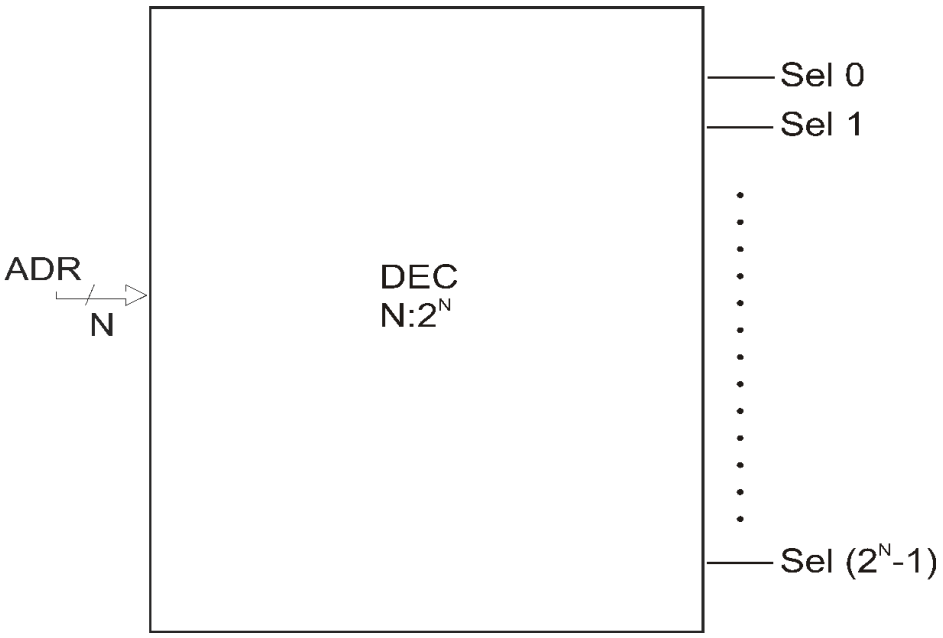


Figura 1.4. Decodificator N:2^N **(a)**

$$\begin{cases} Sel0 = NADR0 * NADR1 * ... * NADR(2^N - 1) \\ . \\ . \\ Sel(2^N - 2) = ADR0 * ADR1 * ... * ADR(2^N - 2) * NADR(2^N - 1) \\ Sel(2^N - 1) = ADR0 * ADR1 * ... * ADR(2^N - 1) \end{cases}$$

Cu 10 biţi de adrese => 2^{10} cuvinte = 1k cuvânt (kilo)
Cu 20 biţi de adrese => 2^{20} cuvinte = 2^{10} k cuvânt = 1M cuvânt (mega)
Cu 30 biţi de adrese => 2^{30} cuvinte = 2^{10} M cuvânt = 1G cuvânt (giga)
Cu 40 biţi de adrese => 2^{40} cuvinte = 2^{10} G cuvânt = 1T cuvânt (terra)
M = 8 => 1 cuvânt = 1 octet

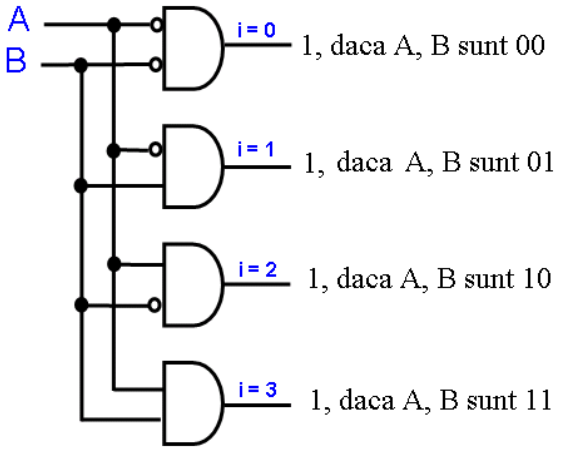


Figura 1.4. Decodificatorul - reprezentare cu porţi logice **(b)**

În general au *n* intrări şi 2^n ieşiri. Decodificatorul furnizează la o singură ieşire 1 iar la restul 0. Ieşirea „*i*" va avea valoarea logică 1 dacă *numărul reprezentat în binar pe intrare* este chiar 1. Funcţia decodificatorului este de interpretare a unui *pattern* de biţi. Utilitatea sa se regăseşte în faza de *decodificare* a instrucţiunilor (vezi modelul *von Neumann*) când câmpul *opcode* – parte componentă a oricărei instrucţiuni – va specifica ce operaţie trebuie efectuată (adunare / scădere / înmulţire / împărţire / acces la memorie / ramificaţie a programului). Decodificatorul este utilizat şi pentru identificarea regiştrilor sursă / destinaţie sau a modurilor de adresare în cazul instrucţiunilor cu referire la memorie tot pe baza *pattern*-urilor de biţi din corpul instrucţiunilor. De asemenea, este folosit la **selecţia adreselor de memorie**.

Din punct de vedere tehnologic se cunosc următoarele **tipuri de memorii**:

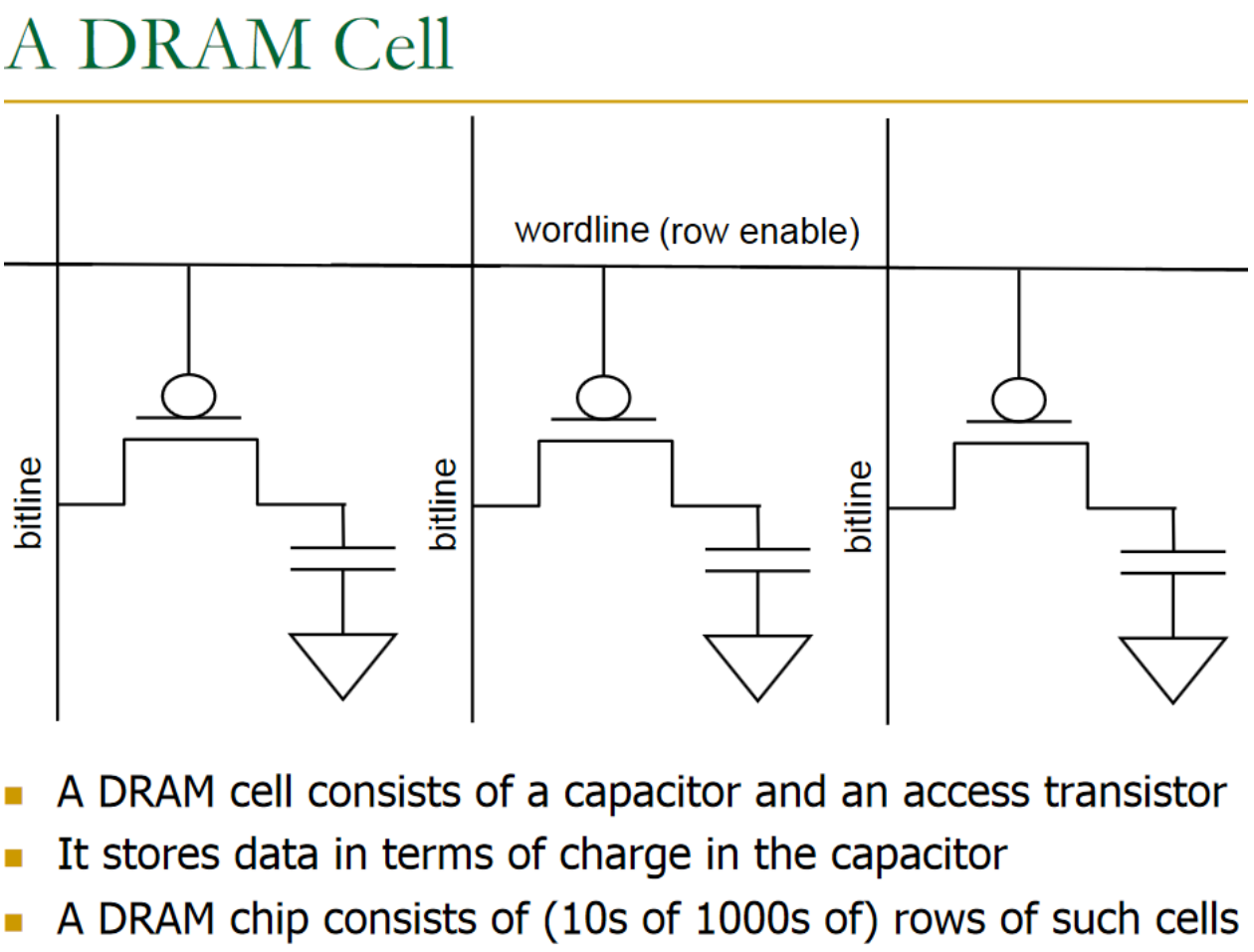
- **ROM** (memorie cu **caracter permanent** – folosită **doar în citire**). Memorie fixă, nevolatilă, în care informaţia se păstrează şi după întreruperea alimentării. Circuitele ROM sunt circuite pur combinaţionale, celula de memorie fiind un tranzistor programat sau nu, amplasat la intersecţia unei linii cu a unei coloane, din matricea de memorie. În ele se înscrie softul de bază al unui sistem de calcul: *sistemul de operare*, **BIOS** (rutina de iniţializare a sistemului de calcul imediat după alimentarea cu energie – teste de memorie, verificare periferice). Memoriile ROM pot fi doar citite nu scrise (scrierea se face o singură dată în momentul fabricării).
  - **PROM (*programmable ROM*) –** poate fi programat de producător sau utilizator la prima folosire a dispozitivului.
  - **UVEPROM (Ultraviolet Erasable PROM) –** sunt identice cu memoriile PROM cu deosebirea că oferă posibilitatea de ştergere şi reprogramare prin expunerea la raze ultraviolete. Necesită un echipament special pentru programare.
  - **EEPROM (Electrical Erasable PROM) –** sunt identice cu memoriile UVEPROM cu deosebirea că ştergerea se face prin aplicarea unor semnale electrice speciale de tensiune ridicată (30V).
  - **FLASH ROM –** este o memorie specială de tip EEPROM care poate fi ştearsă sau programată în timpul funcţionării în circuit (aplicaţii cu microcontrollere în care se încarcă un cod obiect gata de execuţie). Odată programat conţinutul rămâne nemodificat chiar şi după eventuale anomalii datorate căderii tensiunii de alimentare.

Memoriile **EPROM** sunt **reprogramabile** în sensul în care **pot fi şterse prin expunere la raze ultraviolete şi reînscrise** pe baza unui dispozitiv special numit programator de EPROM –uri. EPROM-urile păstrează aşa numitul ***program monitor* înscris de către fabricant care este primul program procesat de către sistem imediat după alimentarea (resetarea) sa**. Acest lucru este absolut necesar întrucât **conţinutul memoriilor RAM este neprecizabil imediat după alimentare**. Prin urmare, imediat după RESET conţinutul PC-ului este iniţializat şi va pointa spre prima

instrucţiune din programul monitor rezident în EPROM. **Rolul programului monitor** este de a efectua o **testare sumară a microprocesorului şi a celorlalte componente ale microsistemului**, după care va **iniţia încărcarea sistemului de operare** *(Linux, Windows* etc.) **de pe disc în memoria RAM**. După aceasta programul monitor **dă controlul sistemului de operare rezident acum în RAM**.

- **RAM** (**memorie cu acces aleator**) - are caracter dinamic **volatil**. Este o memorie care poate fi scrisă sau citită (atâta timp cât este alimentată cu energie). Informaţia se pierde după oprirea alimentării. Accesul la memorie se permite numai în anumite momente de timp validate de un semnal de tip acces la memorie (*memory request*). Există două tipuri majore de memorii RAM:
  - **SRAM (Static RAM) –** sunt memorii **deosebit de rapide**, timp de acces de t_0 = 1 ns ÷ 4 ns (2008), având **capacitate de integrare redusă** (circuite de memorie având capacităţi între 1Mbit şi 16Mbiţi). Mai rapide, mai fiabile dar mai scumpe per unitate de octet memorat. Uzual sunt folosite la implementarea memoriilor ***cache***. De asemenea, consumă mai multă energie. **Folosesc tranzistorii pentru memorarea biţilor de informaţie (între 4 şi 6 tranzistori per bit)**.
  - **DRAM (Dynamic RAM) –** lentă şi necesită regenerare (**celula de memorie o reprezintă un condensator care se descarcă în timp; memoria DRAM trebuie regenerată la fiecare aproximativ 2-8 ms); ieftină** per unitate de octet memorat. Memoria principală este de tip DRAM. Din punct de vedere al **evoluţiei în timp** a memoriilor DRAM se cunosc următoarele etape:
    - ☞ EDO (*Extended Data Out RAM*) – cu 10% până la 20% mai rapidă decât primele memorii DRAM.
    - ☞ SDRAM (*Synchronous DRAM*) – mai rapide cu aproape 25% decât memoriile EDO RAM.
    - ☞ DDR sau SDRAM II (*Double Data Rate SDRAM*) – de două ori mai rapide decât memoriile SDRAM.
    - ☞ RDRAM (*Rambus DRAM*) – dezvoltate de către firma Rambus Inc., sunt de aproape zece ori mai rapide decât memoriile DRAM.
    - ☞ SLDRAM (*Synclink DRAM*) – este principalul competitor din punct de vedere tehnologic al memoriilor RDRAM.
    - ☞ NVRAM (nonvolatile RAM – **FLASH**) – reţine informaţiile şi în absenţa alimentării cu energie (**Avantaj faţă de memoriile cache de tip SRAM**). Se cunosc două alternative de proiectare:
      - 1) cu porţi logice fundamentale **NOR**, utilizat în principal pentru **cod** deoarece oferă un **timp foarte bun pentru citiri (100ns)** şi un timp mai puţin bun pentru scrieri (10µs), repectiv
      - 2) cu porţi logice fundamentale **NAND**, utilizat în principal pentru **date** întrucât oferă un **timp echilibrat atât pentru citiri cât şi pentru scrieri** (1µs). Referitor la comparatia cu memoria cache se observă **Dezavantajul memoriilor Flash care realizează operaţia de scriere într-un timp mai îndelungat decât o scriere în cache**. Se cunoaşte o plajă largă de aplicaţii bazate pe acest tip de memorii: playere MP3, MP4, aparate foto, camere video statice, şi probabil în curând vor înlocui harddisk-urile.

DRAM: constituie peste **90 % din memoria oricărui sistem de calcul** datorită faptului că oferă densităţi mari de integrare (64 Mbiţi –256 Mbiţi / chip) şi timpi de acces "relativ rezonabili" t_0=30 ns ÷ 60 ns. Totuşi, **decalajul între timpul de acces ridicat al acestor memorii şi viteza mare de execuţie a microprocesorului, constituie una dintre marile probleme tehnologice şi arhitecturale în ingineria calculatoarelor (MEMORY WALL)**.



Fiind realizate în tehnologie CMOS puterea absorbită este redusă. Din păcate au **2 mari dezavantaje**:

1. Accesare (citire / scriere) complicată. Circuitele DRAM sunt organizate sub o formă matricială, pe linii şi coloane. Bitul ce se doreşte a fi accesat se află la intersecţia liniei cu coloana selectată.

Un circuit DRAM are următoarele terminale (pini):

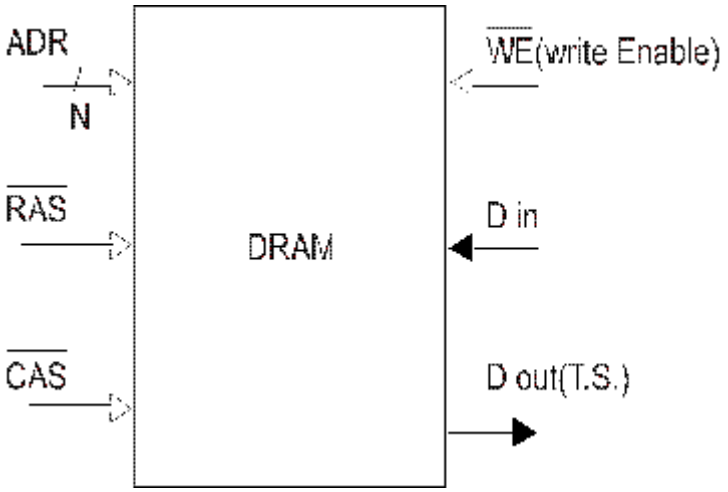


Figura 1.5. Circuit DRAM

RAS (*Row Address Strobe* – strob adresă rând), CAS (*Column Address Strobe* – strob adresă coloană)

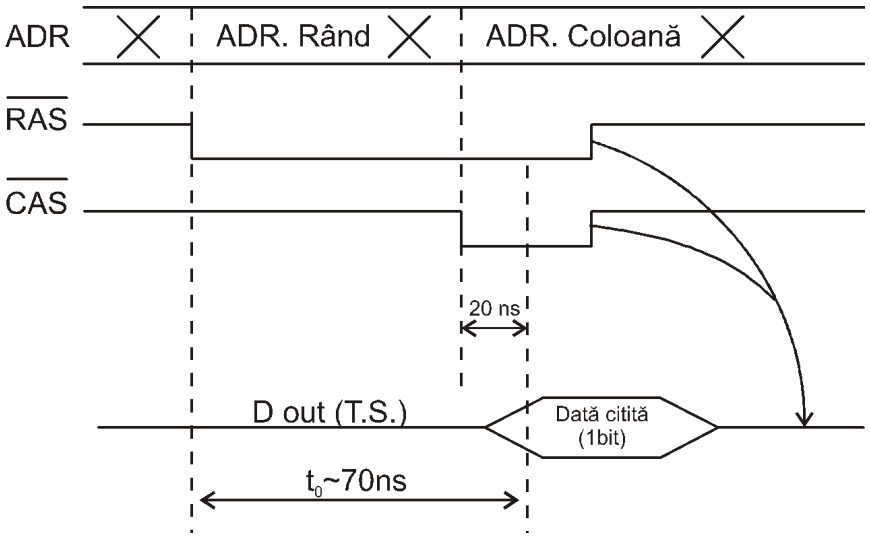


Figura 1.6. Ciclul de citire din DRAM

Pentru o citire corectă este necesar ca frontul căzător al lui RAS să strobeze perfect adresa de rând (o memorează într-un registru intern al circuitului) iar frontul căzător al lui CAS să strobeze perfect adresa de coloană. În momentul memorării adresei de coloană (căzătorul lui CAS) memoria are toate elementele necesare furnizării bitului de ieşire Dout, după un timp precizat în catalog (de cca. 20 ns la memoriile 4164). Rezultă deci că interfaţa între microprocesor şi DRAM este complicată întrucât din semnalele pe care le furnizează microprocesorul (adresă de memorie şi comandă), interfaţa trebuie "să fabrice" protocolul mai sus expus (secvenţa ADR, RAS, CAS…).

2. Necesitatea regenerării memoriei DRAM.

DRAM Refresh

- DRAM capacitor charge leaks over time
- The memory controller needs to refresh each row periodically to restore charge
  - Activate each row every N ms
  - Typical N = 64 ms
- Downsides of refresh
  -- Energy consumption: Each refresh consumes energy
  -- Performance degradation: DRAM rank/bank unavailable while refreshed
  -- QoS/predictability impact: (Long) pause times during refresh
  -- Refresh rate limits DRAM capacity scaling

Bitul de informaţie DRAM este implementat sub forma unui condensator. Din păcate acest condensator **se descarcă în timp şi prin urmare cu timpul poate să piardă informaţia pe care o memorează** => deci că **periodic** el trebuie reîncărcat (***refresh***, regenerare).

Regenerarea se face pe întreg rândul din matricea de memorare. Conform catalogului un anumit rând "ţine minte" informaţia circa 2 ms. După acest interval, întreg rândul trebuie regenerat. **Algoritmul de regenerare va trebui să fie unul de tip secvenţial care să regenereze rând după rând în mod ordonat.**

Rezultă că **rata de regenerare a 2 rânduri succesive *i* respectiv (*i+1*) trebuie să se facă la un interval de maxim 2 ms/N, unde N=nr. de rânduri al circuitului de memorie DRAM**.

De exemplu, considerând N=128 rânduri (DRAM 4116, 4164)

=> rata de regenerare 2 ms/128 ~ 15,6 µs.

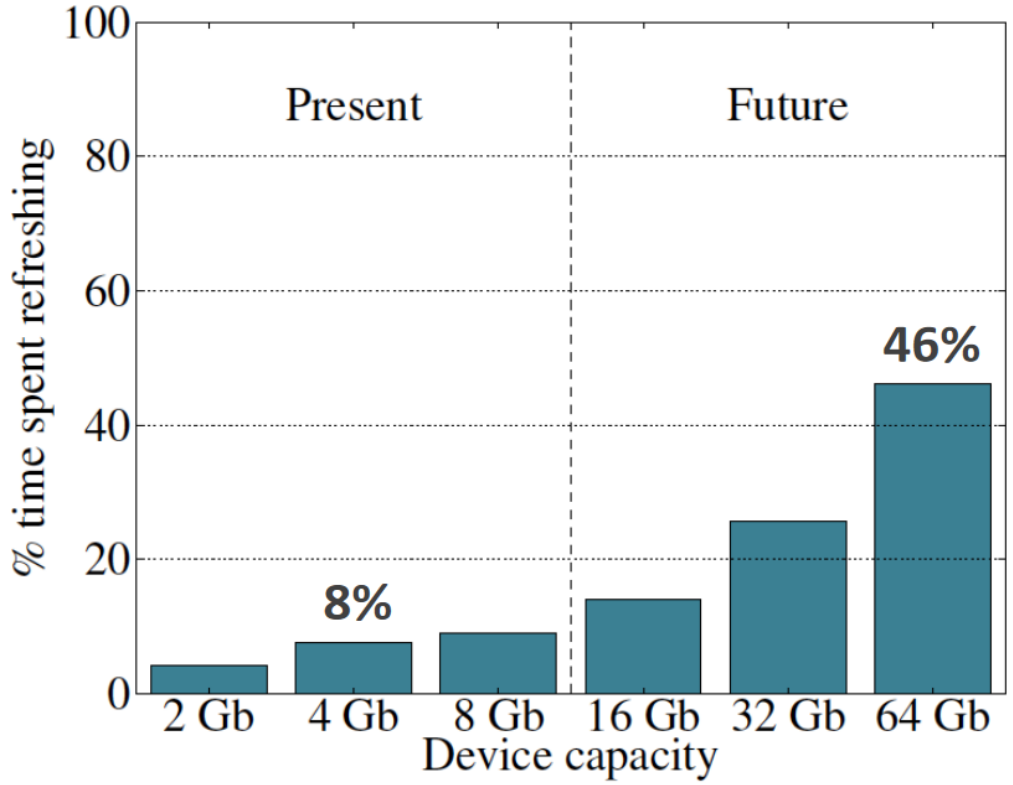
Prin urmare vom avea **accese la DRAM din 2 părţi**:

- **din partea µprocesorului care citeşte / scrie conform programului pe care îl execută**
- **din partea unui automat de regenerare care regenerează periodic, rând după rând, memoria DRAM**.

Posibilele conflicte la memoria DRAM între microprocesor şi automatul de regenerare vor trebui gestionate corespunzător, eventual acceptând chiar prin blocaje reciproce, rezultând scăderea performanţei. Ar fi utilă implementarea unei "regenerari transparente" (care sa nu blocheze deloc procesorul).

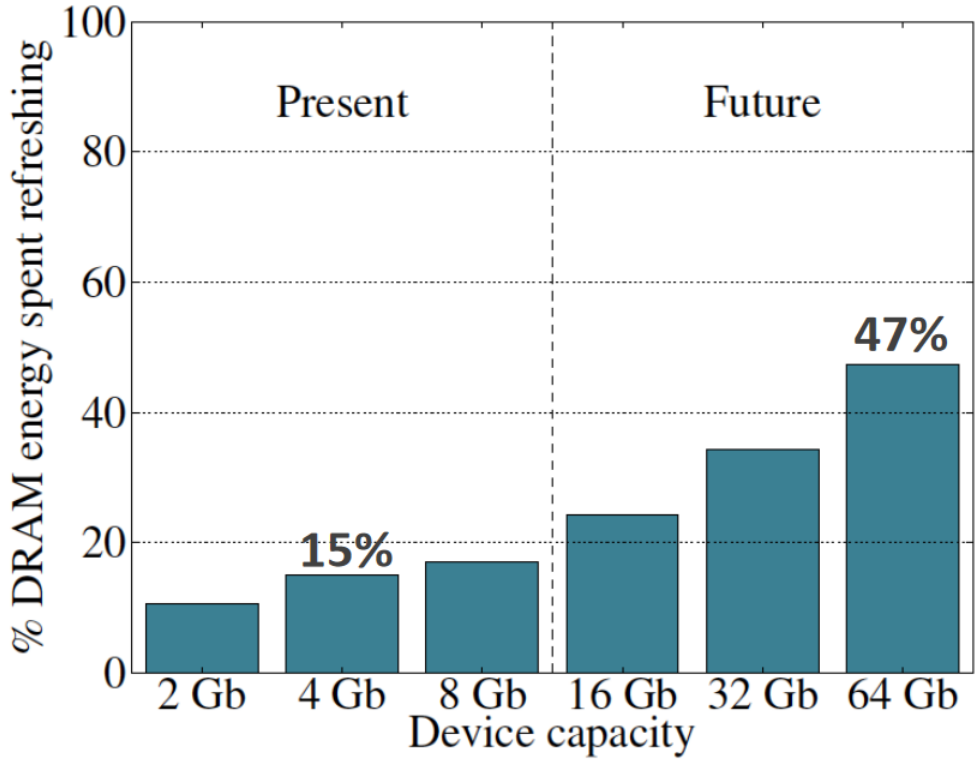
Standardizarea bus-urilor şi a protocoalelor aferente a condus la conceptul de **microsistem de dezvoltare**. Într-un asemenea microsistem, prin simpla conectare la bus-ul comun (date, adrese, comenzi, stări) standard a unor noi module compatibile (memorii, interfeţe) se permite o dezvoltare în timp a microsistemului iniţial.

Refresh Overhead: Performance



SAFARI Liu et al., "RAIDR: Retention-Aware Intelligent DRAM Refresh," ISCA 2012.

Refresh Overhead: Energy

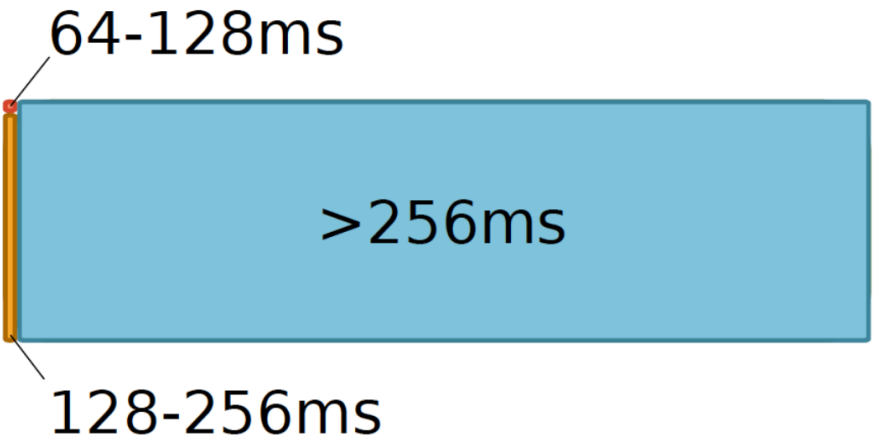


SAFARI Liu et al., "RAIDR: Retention-Aware Intelligent DRAM Refresh," ISCA 2012.

How Do We Solve the Problem?

- Observation: All DRAM rows are refreshed every 64ms.
- Critical thinking: Do we need to refresh all rows every 64ms?
- What if we knew what happened underneath and exposed that information to upper layers?

Underneath: Retention Time Profile of DRAM

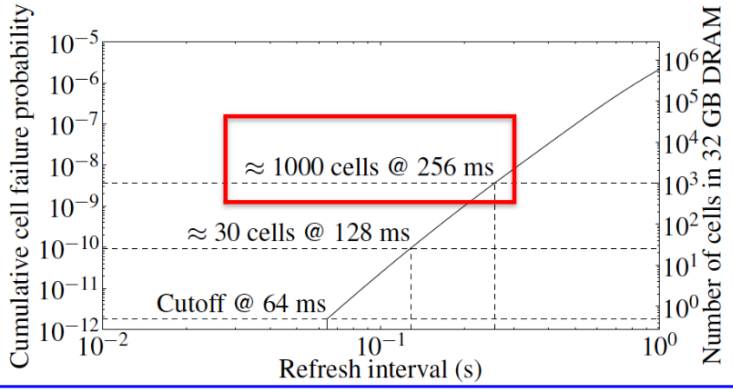


Aside: Why Do We Have Such a Profile?

- Answer: Manufacturing is not perfect
- Not all DRAM cells are exactly the same
- Some are more leaky than others
- This is called Manufacturing Process Variation

Retention Time of DRAM Rows

- Observation: Overwhelming majority of DRAM rows can be refreshed much less often without losing data



Key Idea of RAIDR: Refresh weak rows more frequently, all other rows less frequently

Cam cât timp durează procesarea unei instrucţiuni ?

Exemplu: instrucţiunea **ADD reg1,(reg2)200**, codificată pe un singur cuvânt egal cu lărgimea de bandă a interfeţei procesor – memorie.

reg1 (reg1) + cuv. mem| adr.(reg2)+200

$$\begin{cases} f_{CLK} = 2\,\text{GHz} \Leftrightarrow T_{CLK} = \dfrac{1}{f_{CLK}} = 500\,\text{ps} = 0.5\,\text{ns} \\ \text{Timpul de acces DRAM}: T_{aDRAM} = 10\,\text{ns} \end{cases}$$

Specificaţiile legate de timpul de acces la DRAM au fost preluate din următoarele două referinţe, în care se consideră identic (dar în mod independent unul de celălalt) că Ta_{DRAM}=5ns÷40ns pentru circuite de memorie având capacităţi între 64 Mbiţi şi 1Gbit.

- Nikkei Electronics Asia – July 2008 (*IBM Develops Ultra-Dense Magnetic Memory*, http://techon.nikkeibp.co.jp/article/HONSHI/20080625/153878/)

- Department of Electrical Engineering, Stanford University (*Lecture 5 Main Memory Systems: EE282 – Fall2008,* http://eeclass.stanford.edu/ee282**)**

Această instrucţiune se procesează, conform modelului secvenţial clasic (atribuit marelui matematician american *John von Neumann* – autorul primului document referitor la un calculator electronic numeric pe deplin funcţional), în faze succesive, după cum urmează:

Faza **IF** (*instruction fetch*): ciclu de citire din memorie de la adresa dată de PC (*program counter*) durează aproximativ, spre ex., 20 T_{CLK} (conform protocolului unui ciclu mediu de acces la memorie) = 10 ns.

Faza **ID** (*instruction decode*): în această fază microprocesorul decodifică instrucţiunea ("înţelege" ce trebuie să facă în continuare) şi ca urmare aduce într-un registru temporar intern, situat la intrarea ALU, operandul din reg1. Decodificarea instrucţiunii consumă uzual $1T_{CLK}$=0.5 ns.

Calcul adresă memorie a celui de-al doilea operand (reg2) + 200 $ADR_{(BUS)}$ "atacă" memoria: ~ 1 T_{CLK}=0.5 ns.

Declanşare ciclu de aducere a operandului din memorie de la adresa anterior calculată. Durează, ca şi IF, ~ 20 T_{CLK} = 10 ns.

Faza **EX**: execuţia propriu-zisă a instrucţiunii, adunarea celor 2 operanzi, durează ~ 1 T_{CLK} = 0.5 ns **=> timpul de procesare aferent acestei instrucţiuni T~21.5 ns => nr. de instrucţiuni pe secundă:~ 46511628 instrucţiuni (**performanţa **46.5 MIPS)** (Pentium I ~ 15 MIPS)

De remarcat faptul că **o instrucţiune se procesează sub forma unei înlănţuiri de cicli maşină (faze)**. Un ciclu maşină reprezintă o înlănţuire de **acţiuni sincronizate cu un impuls de tact**. **Ciclul maşină este unitatea atomică de procesare, cea care nu poate fi întreruptă de nici o cerere externă**. **CICLUL INSTRUCŢIUNII** (totalitatea fazelor de procesare care compun instrucţiunea) este neinteruptibil de către cererile de întrerupere hardware (externe), însă el poate fi întrerupt la nivelul unei faze de procesare (la finele acesteia – Fetch, Decode, ...) de către o cerere DMA (acces direct la memorie) din partea unui periferic foarte rapid.

Obs.: Utilitatea modurilor de adresare indirecte prin registru şi indexate (adresa operand din memorie = R+index) este dată de facilitatea scrierii compacte a programelor (bucle), adresării elegante a unor structuri de date de tip tablou situate în memorie etc. Astfel de structuri de date se întâlnesc implementate atât în cadrul limbajelor de nivel înalt (ex. în limbajul C, stiva de date asociată unei funcţii şi care trebuie accesată în vederea transmiterii de parametri respectiv revenire dintr-un apel) cât şi al aplicaţiilor scrise în aceste limbaje.

Creşterea decalajului dintre viteza procesoarelor şi timpul de acces la memorie a impus introducerea unui sistem ierarhic de memorie (vezi figura 1.7), **pentru a nu face simţită la nivelul performanţei globale a sistemului încetineala cu care se accesează memoria**. Practic, **cu cât capacitatea memoriei creşte, cu atât creşte timpul de acces la memorie şi se ieftineşte preţul per unitate de octet memorat**. Memoria **cache** este o **memorie situată din punct de vedere logic între CPU (unitatea centrală de procesare) şi memoria principală, mai mică, mai rapidă şi mai scumpă (per byte) decât aceasta** şi gestionată – în general prin hardware – astfel încât să existe o cât mai mare probabilitate statistică de găsire a datei accesate de către CPU, în cache. Aşadar, cache-ul este adresat de către CPU în paralel cu memoria principală (MP): dacă data dorită a fi accesată se găseşte în cache, accesul la MP se abortează, dacă nu, se accesează MP cu penalizările de timp impuse de latenţa mai mare a acesteia, relativ ridicată în comparaţie cu frecvenţa de tact a CPU. Oricum, data accesată din MP se va introduce şi în cache.

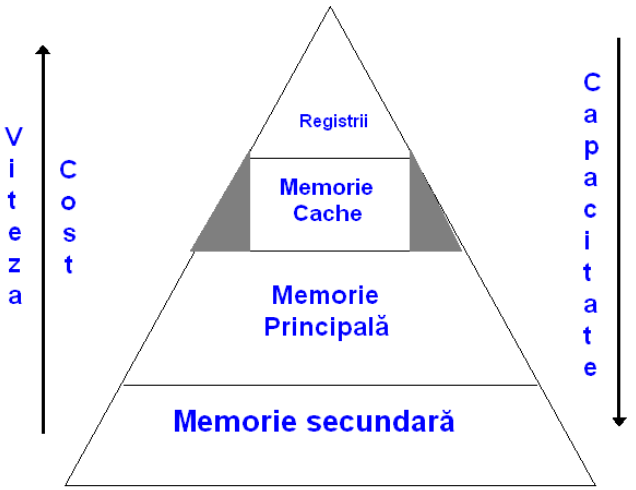
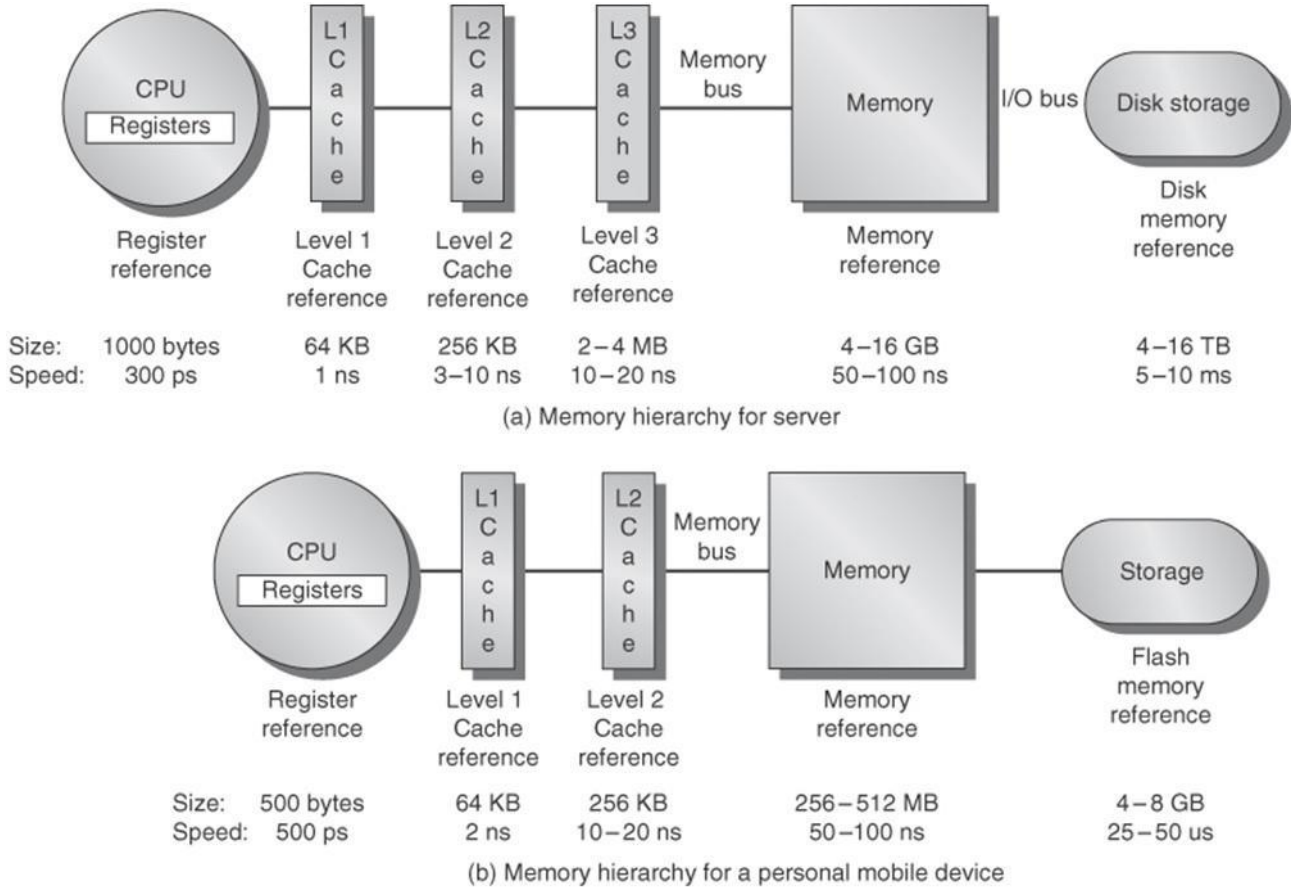


Figura 1.7. Ierarhizarea memoriei într-un sistem de calcul



*Caracteristicile de **viteză și capacitate de stocare** aferente fiecărui nivel din ierarhia de memorie*[Hen12]

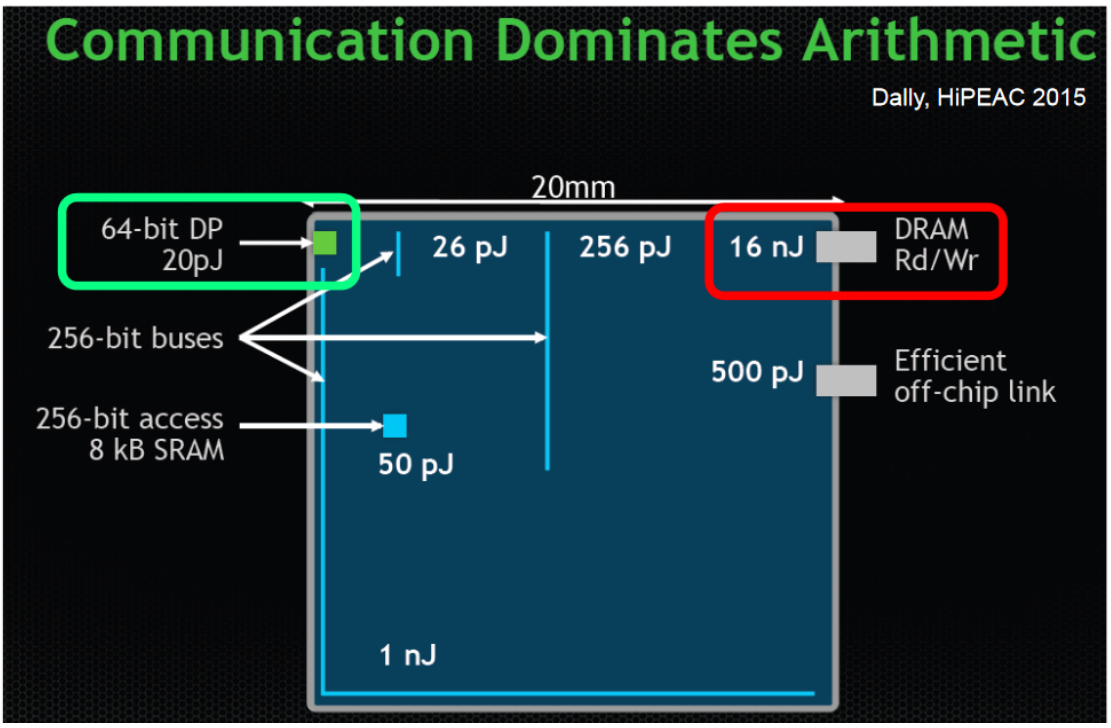
Some Takeaways

- It is an exciting time to be understanding and designing computing platforms
- Many challenging and exciting problems in platform design
  - That noone has tackled (or thought about) before
  - That can have huge impact on the world's future
- Driven by huge hunger for data (Big Data), new applications, ever-greater realism, …
  - We can easily collect more data than we can analyze/understand
- Driven by significant difficulties in keeping up with that hunger at the technology layer
  - Three walls: Energy, reliability, complexity

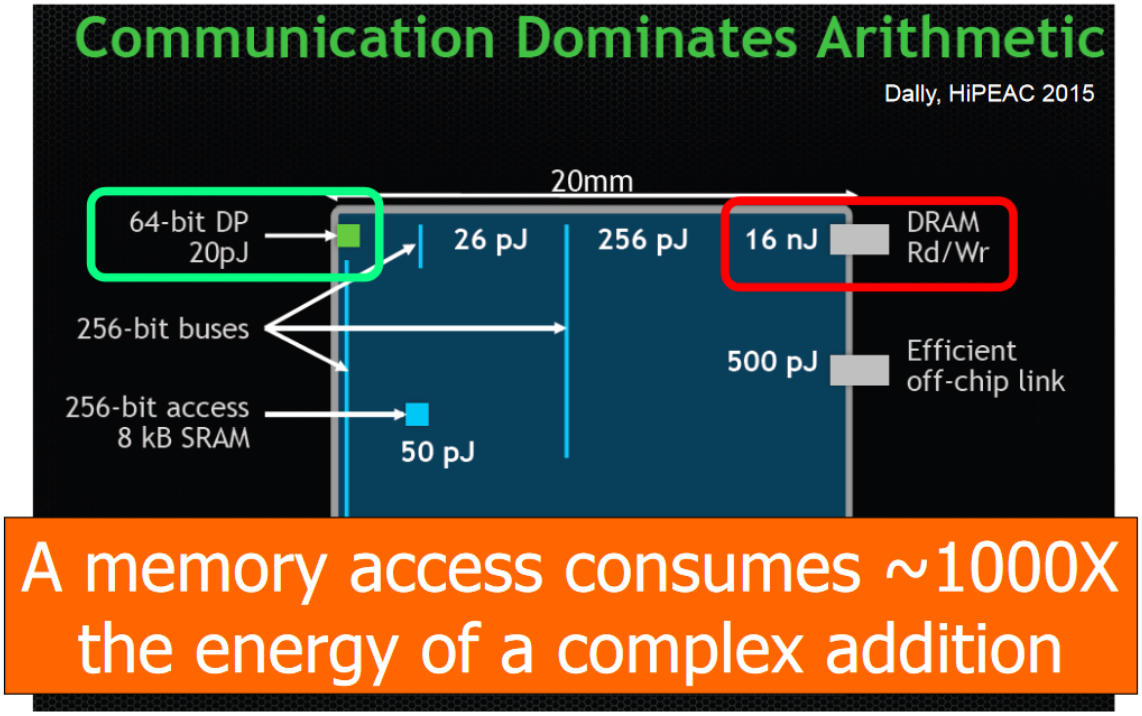
Increasingly Demanding Applications

- Dream, and they will come

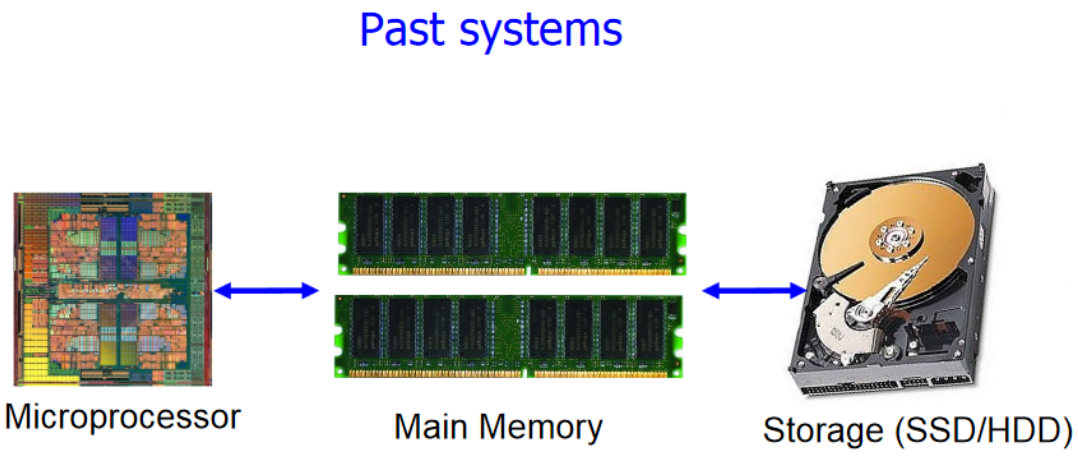
Increasingly Diverging/Complex Tradeoffs

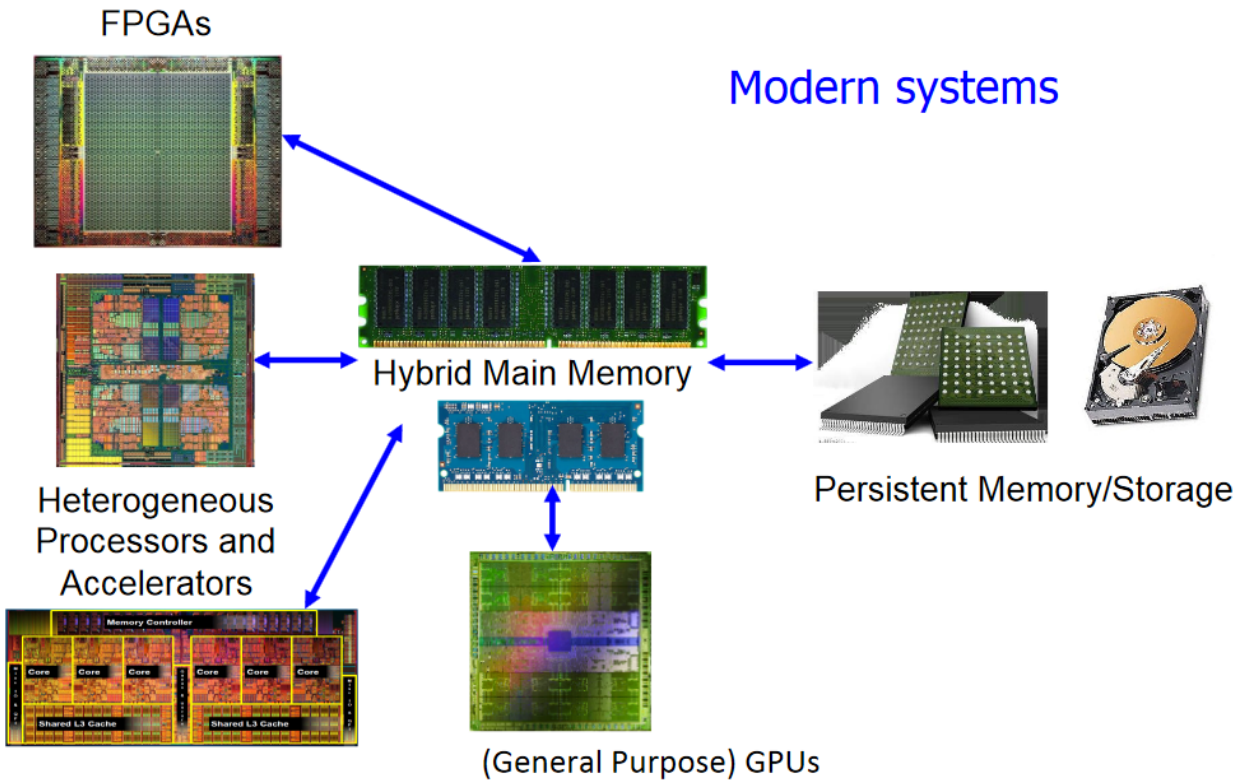


Increasingly Diverging/Complex Tradeoffs



Increasingly Complex Systems

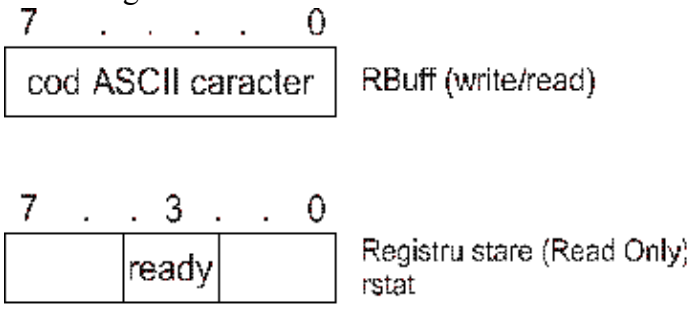




1.2. MODURI DE LUCRU ÎNTRE MICROPROCESOR ŞI INTERFEŢELE I/O

1.2.1. MODUL DE LUCRU PRIN INTEROGARE ("*POLLING*")

Se bazează pe **testarea de către microprocesor a unui bit de stare asociat dispozitivului periferic**. Microprocesorul nu **va iniţializa transferul cu perifericul** decât **în momentul în care bitul de stare semnifică faptul că perifericul este pregătit pentru transfer** (nu lucrează la un transfer iniţiat anterior). Să considerăm de exemplu **interfaţa cu o tastatură**. Această interfaţă trebuie să conţină minimum 2 registre.

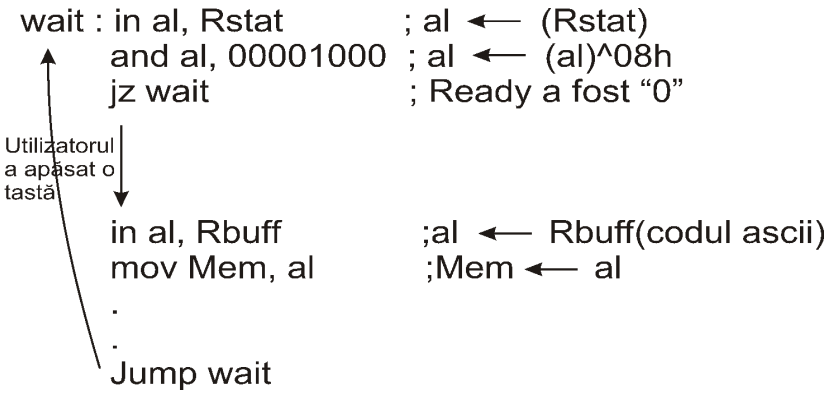


RBuff va memora un octet care reprezintă codul ASCII (Unicode) al tastei apăsate de către utilizator.

Exemple:

“A” = 41h ⇔0100.0001
“a” = 61h ⇔0110.0001
“0” = 30h
“ ” = 20h

Bitul *Ready* din registrul de stare este un bit de tip *Read Only* cu următoarea **semnificaţie**: **dacă registrul RBuff se încarcă cu un octet (utilizatorul a apăsat o tastă) atunci *Ready* se pune automat pe “1” arătând microprocesorului că poate să preia codul din RBuff. Bitul *Ready* se va reseta automat odată cu preluarea codului din Rbuff de către microprocesor.** Un program - absolut principial - de gestiune a tastaturii s-ar scrie ca mai jos:



Q5: Credeţi că modul de lucru prin interogare reprezintă o bună abordare pentru celelalte periferice (mai rapide) – cum ar fi memoriile externe (disk) sau interfeţele de reţea?

Dezavantajul acestei metode constă în faptul că microprocesorul aşteaptă un timp T, neacceptabil de mare la nivelul vitezei sale, pentru a inspecta dacă perifericul este sau nu este pregătit. Considerând că utilizatorul apasă **o tastă la interval de 250 ms** şi că **o instrucţiune a microprocesorului durează cca. 21.5 ns => că “pierde” 250 ms / 21.5 ns ≈ 11.6 milioane instrucţiuni în bucla de aşteptare în loc să execute instrucţiuni utile**. Acest dezavantaj este eliminat de metoda următoare de comunicare procesor-interfaţă.

1.2.2. MODUL DE LUCRU PRIN ÎNTRERUPERI HARDWARE (MASCABILE)

Se bazează pe generarea unui semnal de întrerupere INT de la interfaţă (port) spre microprocesor ori de câte ori acesta doreşte un serviciu de la microprocesor. **Ca urmare a recepţionării semnalului INT microprocesorul va abandona programul principal (PP) urmând să intre intr-o aşa numită rutină tratare a întreruperii în care va satisface cererea interfeţei. La finele rutinei de tratare a întreruperii printr-o instrucţiune de tip RETURN, microprocesorul va reveni in PP, in general dar nu întotdeauna, pe instrucţiunea imediat următoare ultimei instrucţiuni din PP executate**. In cazul exemplului cu tastatura anterior considerat, interfaţa va genera întreruperea INT ori de câte ori sutilizatorul a apăsat o tastă, adică registrul RBuff este “plin”, deci conţine codul (ASCII, Unicode etc.) al caracterului tastat.

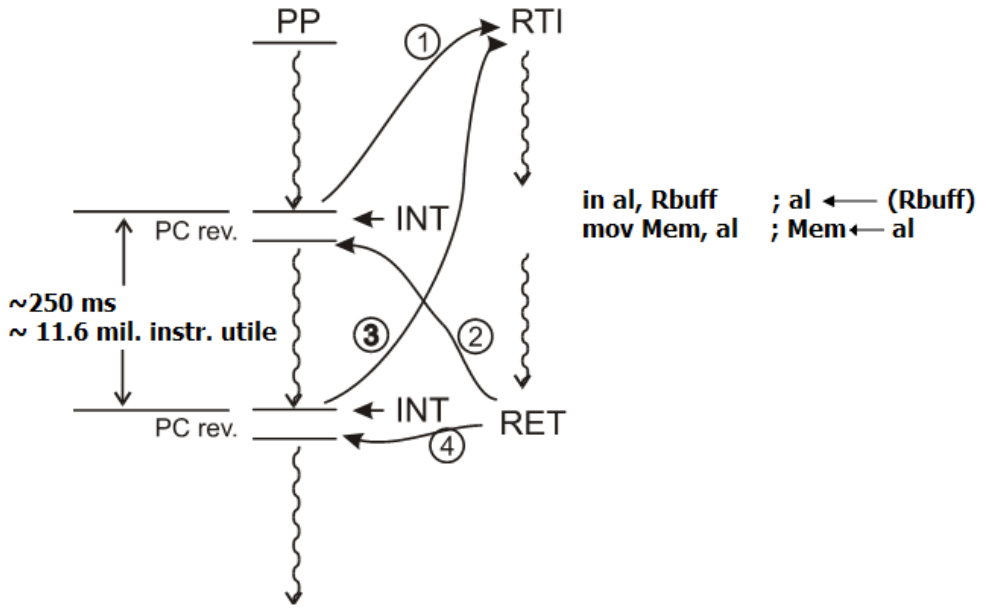


Figura 1.8. Modelul de lucru prin întreruperi

Aşadar RTI după ce execută serviciul necesar perifericului (în cazul acesta preluare şi depozitare caracter în memorie) revine în PP, unde până când perifericul cere un nou serviciu (de ex. se apasă din nou o tastă), microprocesorul va executa instrucţiuni utile din PP (sistem de operare, program utilizator etc.) si deci nu mai este necesar să mai aştepte inutil ca in cazul 1.

Totalitatea acţiunilor executate de către microprocesor din momentul apariţiei semnalului de întrerupere INT până în momentul procesării primei instrucţiuni din RTI formează aşa numitul **protocol hardware de acceptare a întreruperii** (săgeţile 1 şi 3 din figură). În principiu acest protocol se desfăşoară în următoarele etape succesive:

1.) Odată sesizată întreruperea INT de către microprocesor acesta îşi va termina instrucţiunea in curs de execuţie după care, dacă anumite condiţii sunt îndeplinite (nu există activată o cerere de întrerupere sau de bus mai prioritare etc.), va trece la pasul 2. În general, **microprocesoarele examinează activarea întreruperilor la finele ultimului ciclu aferent instrucţiunii în curs de execuţie**.

2.) Recunoaşterea întreruperii: microprocesorul va iniţializa aşa numitul **ciclu de achitare a întreruperii**. Pe parcursul acestui ciclu extern va genera un semnal de răspuns (achitare) a întreruperii INTACK (*interrupt acknowledge*) spre toate interfeţele de intrare - ieşire. Ca urmare a recepţionării INTACK interfaţa care a întrerupt va furniza microprocesorului prin intermediul bus-ului de date un aşa numit octet **vector de întrerupere** (**VI**). Acest VI este diferit pentru fiecare periferic în parte, individualizându-l deci într-un mod unic. Pe baza acestui VI şi conform unui algoritm care diferă de la microprocesor la microprocesor, acesta va determina adresa de început a RTI, adresă ce va urma să o introducă in PC. Firește, la VI diferiţi vor corespunde adrese de început diferite.

3.) Microprocesorul va salva într-o zonă specială de program numită **memorie stivă**, PC-ul aferent instrucţiunii imediat următoare instrucţiunii executate de către microprocesor din PP (PCrev), pentru a putea şti la finele RTI unde să revină exact în PP.

Memoria stivă este o zonă de memorie RAM caracterizată la un moment dat de aşa numitul vârf al stivei **adică de ultima locaţie ocupată din stivă**.

Acest vârf al stivei este pointat (adresat) permanent de conţinutul unui registru special dedicat, existent în orice microprocesor modern, numit SP (*stack pointer*).

În memoria stivă sunt posibile 2 tipuri de operaţii:

- operaţia **PUSH Reg** care se desfăşoară astfel:

$$\begin{cases} SP \leftarrow (SP) - 1 \,(cuvânt = octet) \\ (Reg) \rightarrow Mem \,|\, adr.SP \end{cases}$$

operaţia **POP Reg**:

$$\begin{cases} (Reg) \leftarrow Mem\,|\,adr.SP \\ SP \rightarrow (SP) + 1 \end{cases}$$

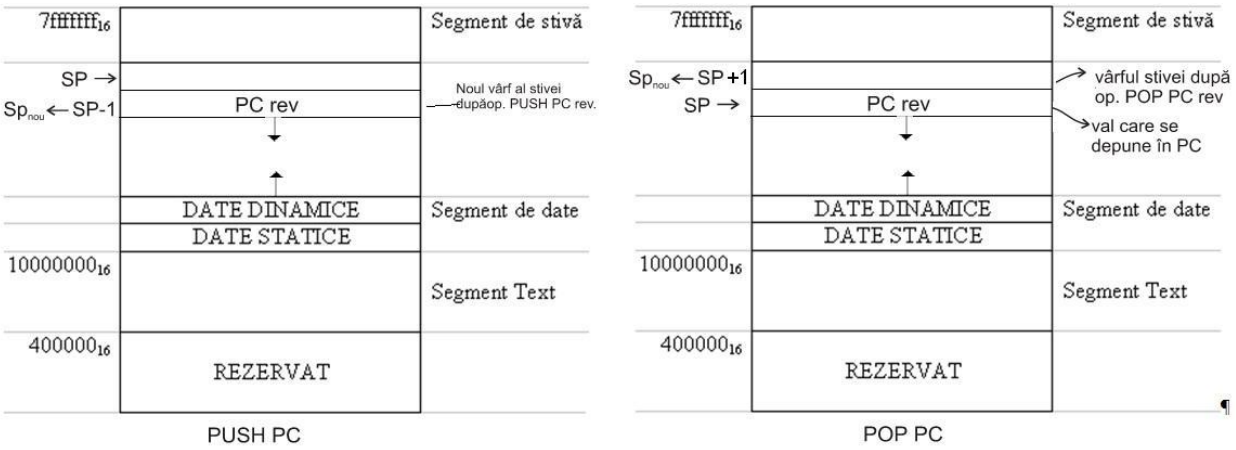


Figura 1.9. Modul de lucru al stivei la microprocesorul MIPS

Stiva este o memorie de tip LIFO (*last in first out*) si care spre deosebire de PC în procesarea secvenţială, "creşte" (PUSH) de obicei înspre adrese descrescătoare evitându-se astfel suprapunerea zonelor de program (cod) cu cele de stiva.

4.) Intrarea în RTI se face simplu prin introducerea adresei de început a RTI calculată in pasul 2, in registrul PC. Normal în continuare microprocesorul va aduce şi executa prima instrucţiune din RTI protocolul de tratare fiind în acest moment încheiat şi controlul fiind preluat de RTI a perifericului care a fost întrerupt.

După cum s-a observat protocolul de tratare salvează in stiva doar PC-ul de revenire (la anumite microprocesoare se mai salvează registrul de stări - *flags*). Acest fapt se poate dovedi insuficient având în vedere că in cadrul RTI pot fi alteraţi anumiţi regiştri interni ai microprocesorului. Această alterare a regiştrilor poate fi chiar catastrofală la revenirea în PP. Din acest motiv cade in sarcina celui care scrie RTI să salveze (instrucţiuni PUSH) respectiv să returneze corespunzător (instrucţiuni POP) aceşti regiştri.

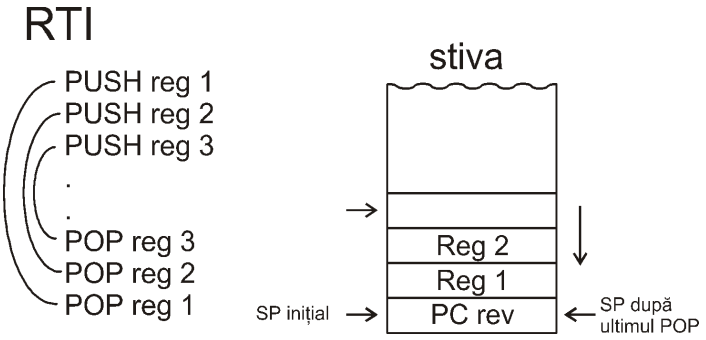


Figura 1.10. Efectul RTI asupra stivei

Acţiunea instrucţiunii RETURN este echivalentă cu o operaţie de tip POP PC.

$$RET: \begin{cases} a) PC \leftarrow Mem\,|\,adrSP \\ b) SP \leftarrow SP + 1 \end{cases}$$

Acum devine evident de ce instrucţiunea RETURN implementează revenirea în PP pe instrucţiunea imediat următoare celei întrerupte.

Obs.: **Prin mecanismul de stivă se pot gestiona perfect şi întreruperile de tip imbricat** (apariţia unei întreruperi INT în chiar rutina de tratare a altei întreruperi, când este permis).

1.2.3. MODUL DE LUCRU PRIN TRANSFER DMA (*DIRECT MEMORY ACCESS*)

Există dispozitive periferice a căror rată de transfer (octeţi /secundă) este atât de ridicată încât, din motive de *timing*, face imposibil modul de lucru prin întreruperi. Astfel de exemplu discurile magnetice şi interfeţele video, impun **rate de transfer de 20-40 MBytes /s rezultând 1 octet la aproximativ 50 ns până la 25 ns**. Este evident că, fără un buffer FIFO (*First In First Out*) între periferic şi memorie, transferul prin întreruperi este imposibil în acest caz întrucât rata de transfer este **comparabilă (chiar mai rapidă!) cu durata unei instrucţiuni a microprocesorului**. Aşadar, **în aceste cazuri durata RTI ar fi mult mai mare decât rata de transfer a perifericului** (octeţi per secundă). Un monitor video este un alt periferic rapid de vreme ce, pe durata unei curse directe a baleiajului pe orizontală a spotului de câteva zeci de microsecunde, trebuie afişate zeci sau chiar sute de octeţi (caractere). De aceea în aceste cazuri se impune un transfer direct între memorie şi dispozitivul periferic.

Cererea DMA este luată în calcul de procesor la finele unui ciclu maşină (Fetch, Decode sau altul aflat în execuţie).

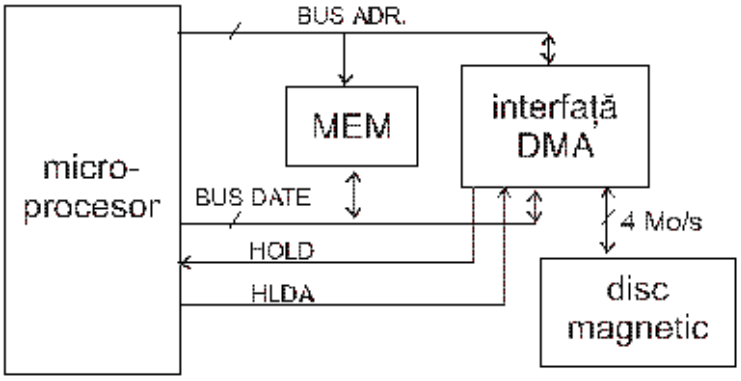


Figura 1.11. Modul de lucru prin transfer DMA

Atunci când se doreşte prin program **transferul unor octeţi din memorie pe disc sau citirea de pe disc în memorie microprocesorul va scrie în interfaţa DMA aferentă** (prin instrucţiuni de tip OUT succesive) **următoarele informaţii**:

- **adresa de început de pe disc** (nr. cilindru, nr. cap, nr. sector). *Header* reprezintă adresa de început sector, deci un identificator al sectorului care se scrie la formatarea fizică a discului

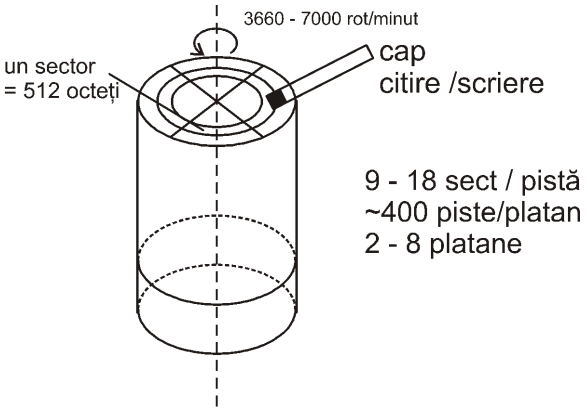
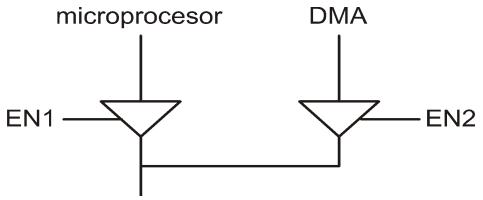


Figura 1.12. Structura discului magnetic

- **adresa de început a zonei de memorie (RAM) utilizată în transfer**
- **nr. octeţi (sectoare) care trebuiesc transferate**
- **sens transfer (*Write* sau *Read* pe / de pe disc)**

În urma recepţionării acestor informaţii **interfaţa DMA va activa un semnal numit cerere de bus (HOLD) spre microprocesor**. Ca urmare a recepţionării semnalului HOLD, **la finele ciclului maşină în curs microprocesorul îşi va pune bus-urile de adrese date şi comenzi in TS permiţând astfel controlul acestora de către DMA** (EN1=1, microprocesor master pe bus, EN2=1, DMA master pe bus).



Simultan microprocesorul va activa semnalul de răspuns la HOLD numit semnal de achitare a cererii (HLDA). Ca urmare a recepţionării achitării HLDA (*Hold Acknowledge*), **DMA-ul va starta transferul efectiv între disc şi memorie având toate informaţiile necesare pentru aceasta**. Spre exemplu, dacă s-a comandat citire de pe disc (scriere în memorie) DMA-ul va adresa memoria pe bus-ul de adrese simultan cu punerea pe bus-ul de date a cuvântului (octetului) scris în memorie. **La finele transferului DMA interfaţa va dezactiva semnalul HOLD, ca urmare microprocesorul, dezactivând şi el semnalul HLDA, îşi va continua activitatea întreruptă prin procesarea următorului ciclu maşină. O cerere de bus (HOLD) este prioritară faţă de o cerere de întrerupere (INT).**

De remarcat că **diferenţa de principiu între transferurile prin interogare - întreruperi şi respectiv transferul DMA constă în faptul că în cazul primelor două transferul se face programat, prin intermediul microprocesorului care serveşte periferic în cadrul rutinei de tratare, pe când în cazul DMA se face fără intervenţia microprocesorului, direct între memorie şi interfaţa DMA**. Pe timpul HLDA=1, **microprocesorul îşi întrerupe orice activitate externă**, master pe bus fiind DMA-ul. **Un sistem de calcul cu DMA este un arhetip al sistemelor multiprocesor**.

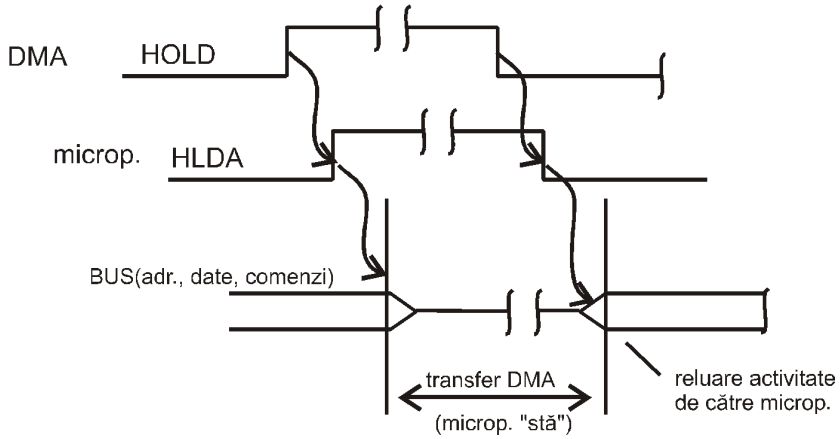


Figura 1.13. Cronograma unui transfer DMA

Modul de lucru prin transfer DMA apare în cadrul procesorului de tip multicore **IBM-CELL** (compus dintr-un element puternic de procesare PowerPC pe 64 de biţi şi 8 procesoare specializate SIMD – single instruction multiple data). Procesoarele SIMD accesează memoria comună şi partajată de către toate cele 9 nuclee de procesare prin comenzi DMA asincrone. Prin DMA se transferă date şi instrucţiuni între memoria comună şi memoriile locale (private fiecărui procesor SIMD). Scopul este de a exploata paralelismul de calcul din sistemul multicore, având în vedere că **timpul petrecut cu setarea parametrilor transferului DMA este incomparabil mai mic decât *memory wall*-ul dintr-un sistem monocore**. În cadrul sistemelor multicore sau multiprocesor, **singurele instrucţiuni care nu pot fi întrerupte de o cerere DMA sunt instrucţiunile atomice** (**read/modify/write**) care ajută la păstrarea coerenţei şi consistenţei datelor din memoria partajată.

Problemă:

1. A given processor requires 1000 cycles to perform a context switch and start an interrupt handler (and the same number of cycles to switch back to the program that was running when the

interrupt occurred), or 500 cycles to poll an I/O device. An I/O device attached to that processor makes 150 requests per second, each of which takes 10,000 cycles to resolve once the handler has been started. By default, the processor polls every 0.5 ms if it is not using interrupts.

a) How many cycles per second does the processor spend handling I/O from the device if interrupts are used?

b) How many cycles per second are spent on I/O if polling is used (include all polling attempts)? Assume the processor only polls during time slices when user programs are not running, so do not include any context-switch time in your calculation.

c) How often would the processor have to poll for polling to take as many cycles per second as interrupts?

Soluţie:

1. a. The device makes 150 requests, each of which require one interrupt. Each interrupt takes 12,000 cycles (1000 to start the handler, 10,000 for the handler, 1000 to switch back to the original program), for a total of 1,800,000 cycles spent handling this device each second.

b. The processor polls every 0.5 ms, or 2000 times/s. Each polling attempt takes 500 cycles, so it spends 1,000,000 cycles/s polling. In 150 of the polling attempts, a request is waiting from the I/O device, each of which takes 10,000 cycles to complete for another 1,500,000 cycles. Therefore, the total time spent on I/O each second is 2,500,000 cycles with polling.

c. In the polling case, the 150 polling attempts that find a request waiting consume 1,500,000 cycles. Therefore, for polling to match the interrupt case, an additional 300,000 cycles must be consumed, or 600 polls, for a rate of 600 polls/s.

Bibliografie

[EDA11] http://eda360insider.wordpress.com/2011/12/06/fast-serial-io-whats-its-future-in-the-embedded-world/

[Hen07] Hennessey J., Patterson D. Elsevier, Computer Architecture: A Quantitative Approach, fourth edition, Elsevier, 2007.

[Hen11] Hennessey J., Patterson D. Elsevier, Computer Architecture: A Quantitative Approach, fifth edition, Elsevier, 2011.

[Vin19] Vinţan L.N., Fundamente ale arhitecturii microprocesoarelor, Editura MatrixRom 2019, ISBN: 978-606-25-0276-8, https://www.matrixrom.ro/produs/fundamente-ale-arhitecturii-microprocesoarelor/ .