02.10.2024

Arhitectura calculatoarelor

(curs 1-S1)

Flavius Opritoiu flavius.opriţoiu@upt.ro

marti ⟶ 12^{00} - 14^{00} ⟶ B420

Capitolul 0. Introducere

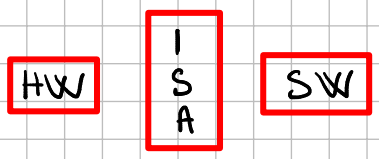
Cele 5 componente clasice ale unui sistem de calcul

- input
- output
- memory
- datapath
- control

Datapath duce greul --> realizeaza operatii aritmetice, instructiuni de virtualizare, lucreaza cu siruri de caractere.

Control --> coordoneaza restul, indicand operatiile de efectuat, in concordanta cu instructiunile programului executat de microprocesor

Instruction Set Architecture (ISA)



- cunoscut ca Architectura unui calculator
- reprezintă interfața între componentele hardware și software ale calculatorului
- include tot ceea ce un programator trebuie să cunoască pentru a putea construi un program în limbaj mașină care să fie executat corect de calculator
  - instrucțiuni, dispozitive Input/Output (I/O), ierarhii de memorie,

ISA vede tot în mod unitar

ISA: permite descrierea funcționalității unui Central Processing Unit (CPU) într-o maniera independentă de hardware-ul din interiorul CPU-ului

Exemplu: ceas digital

x86_64 ⟶ Intel ⟶ 8086

ISA: ascunde detaliile complexe de construcție ale calculatorului care implementează respectivul ISA

- faciliteaza inovația la nivelul componentei hardware fără modificarea arhitecturii

Exemplu:

- atât 8086 cât și Pentium IV implementează aceeași arhitectură *x86*
  - 8086 conține ≈ 29 mii de tranzistori, având o performanță de 0.33 Millions of Instructions Per Second (MIPS)
  - Pentium IV conține ≈ 44 milioane de tranzistori, având o performanță de aprox. 5000 MIPS

Evoluția ISA

La inceputul anilor 1960, International Business Machines(IBM) Corporation avea 4 ISA diferite

- 701 → 7094: destinat calculului științific
- 702 → 7080: destinat marilor corporații
- 650 → 7074: sisteme de calcul în timp real
- 1401 → 7010: destinat micilor afaceri

Fiecare linie de produse avea propriile: seturi de programe, dispozitive I/O și piață de desfacere ⇒ efort mare de dezvoltare SW

Soluția: unificarea celor 4 ISA ⇒ *IBM System/360 ISA*

- datapath: poate acomoda ușor cuvinte de date înguste sau late
- hardware de control: dificil de proiectat, atât atunci cât și acum

Proiectarea hardware-ului de control

Control microprogramat (Maurice Wilkes)

- inspirat de programarea SW
- controlul este specificat printr-un *control store*
  - tabel bidimensional
  - mai multe elemente de controlat ⇒ mai multe coloane
  - instrucțiunile CPU-ului: formate din secvențe de μinstrucțiuni
    - fiecare μinstrucțiune ocupa o linie în control store
  - instrucțiuni complexe ⇒ mai multe linii în control store

Control store:

- implementat utilizând memorii
  - soluție mai ieftină comparativ cu utilizarea porților logice

IBM a dominat piața prin familia System/360

- System/360 a fost lansat în 1964
- descendenții acestei familii încă aduc profit de miliarde de dolari

Primul calculator personal: *Alto*, creat în 1973

- este un Complex Instruction Set Computer (CISC)
- construit de Xerox Palo Alto Research Center
- primul calculator cu display bit-mapped
- primul calculator care utilizează Ethernet
- controller-ele pentru display și rețea sunt programe în control store-ul de 4K x 32b

"The next big ISA":

- în anii 1970 microprocesoarele sunt pe 8 biți (Intel 8080)
- Gordon Moore: următorul ISA al Intel va dăinui *a la longue*
  - asamblează o echipă în Portland pentru construcția lui
  - noul ISA, numit inițial 8800, ulterior redenumit "iAPX-432"
  - este un proiect ambițios: demarat în 1975, fără a fi însă materializat până în 1981, doar pentru a fi retras în 1986

Urmarea insuccesului lui iAPX-432, Intel demarează un plan de avarie:

- să aibă un microprocesor pe 16 biți până în 1979
- o echipă în Santa Clara: în 52 de săptămâni va dezvolta ISA-ul "8086", va proiecta chip-ul și îl va construi
- ISA-ul 8086 a fost dezvoltat în 3 săptămâni extinzând arhitectura 8080 la 16 biți
- CPU-ul a fost terminat la termen, fără prea mult fast

Oportunitatea lui Intel:

- IBM dezvolta un calculator personal pentru a concura cu Apple II și are nevoie de un CPU pe 16 biți
- IBM era interesat de Motorola 68000
  - 68000 avea un ISA similar cu cel al System/360
  - dar, 68000 nu ține pasul cu planul rapid de dezvoltare al lui IBM
- ca urmare, IBM alege CPU-ul 8086 de la Intel

Calculatorul Personal:

- IBM îl anunță în 12 August 1981
  - IBM speră să vândă 250 mii de unități până în 1986
  - în schimb vinde 100 milioane de unități
- acest deznodământ asigură un viitor luminos arhitecturii de avarie a lui Intel: 8086

În 1985 Intel extinde microprocesorul 8086 pe 16 biți construind microprocesorul 80386 pe 32 de biți.

Prezicerea lui Gordon Moore că următorul ISA va dăinui se împlinește!

- viitorul a aparținut lui 8086, cunoscut și ca arhitectura *x86*
- succesul nu a fost de partea ambițiosului iAPX-432 sau arhitecturii Motorola 68000
  - ambele CPU-uri au "învățat" o lecție dură: *piața nu are răbdare*

CISC - complex instruction set computer

RISC - reduced instruction set computer

De la CISC la Reduced Instruction Set Computer (RISC):

- la începutul anilor 1980 apar unele schimbări de perspectivă:
  - sunt folosite limbaje de nivel înalt pentru dezvolatarea Operating System (OS)
  - întrebarea "ce limbaj de asamblare să folosesc" devine "ce instrucțiuni să utilizeze compilatorul"

Grupul lui John Coke de la IBM analizeaza arhitectura System/360:

- compilatorul folosește doar instrucțiuni simple (cele complexe sunt evitate)
- rezultatul: programele pot fi făcute să ruleze de *3 ori* mai repede folosind instrucțiuni simple
- această cercetare stă la baza tranziției de la CISC la RISC

RISC:

- instrucțiunile unui RISC: la fel de simple ca μinstrucțiunilor unui CISC
  - $\Rightarrow$ hardware-ul de control devine mai puțin complex
- pentru că nu utilizeaza control store, această memorie rapidă preia rolul de cache al instrucțiunilor microprocesorului

Cache: mediu de stocare de dimensiuni mici, rapid care păstrează instrucțiunilor executate recent, acestea fiind cel mai probabil necesare în viitorul apropiat

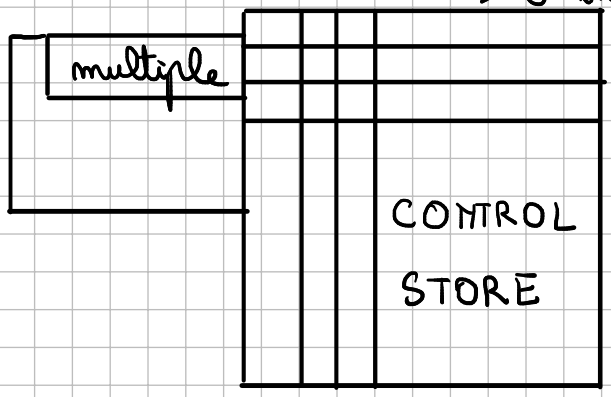
Întrebare De ce sunt necesare instrucțiunile tocmai executate în viitorul apropiat?
Răspuns Considerați fragmentul de cod următor:

```
1  int a = 1;
2  int b = N;
3  do {
4      a = a * b;
5      b = b - 1;
6  } while (b != 0);
```

Memorie largă $\longrightarrow$ control store

nr. coloane = nr. de elemente de controlat per linie

——> o instructiune la o adresă



↗ complex ——> mai multe instructiuni

memoria ——> parte din CPU

CISC TO RISC

bit-mapped ——> grayscale

x*16 ——> x<<4

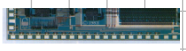
x/16 ——> x>>4

complex reduced

3 * viteza

din control store ——> cache L_1, L_2, L_3

Microprocesoare RISC:



5

- RISC-I dezvoltat la Berkely în 1982 de o echipa ce îl include pe D. Patterson
- MIPS (Microprocessor without Interlocked Pipeline Stages) dezvoltat la Stanford în 1983 de o echipă condusă de J. Hennessy

Arhitecturile RISC au dominat performanța CPU-urilor mai bine de 15 ani

Very Long Instruction Word (VLIW) și arhitectura Explicitlly Parallel Instruction Computer (EPIC):

- previzionate a depăși în performanță RISC și CISC
- EPIC - un efort comun al Hewlett-Packard și Intel
- o instrucțiune de tip wide unește mai multe operații independente
  - 2 operații de acces la memorie
  - 2 operații cu întregi
  - 2 operații cu numere de virgulă mobila
- s-a sperat că tehnologia de compilare va optimiza selecția operațiilor pentru instrucțiunile de tip wide
  - ca și CPU-urile RISC, efortul este transferat dinspre HW către compilator

7

Intel Itanium -> pentru aplicatii de tip digital-signal processing
-> program mic
-> mai putine branch-uri
-> fara cache

EPIC promitea sa înlocuiasca arhitectura x86 pe 32 de biți:

- primul CPU EPIC a fost Itanium, pe 64-biți
  - performanțe ridicate pentru programe în virgulă mobilă structurate
  - performanțe slabe pentru cache miss-uri/branch-uri puțin predictibile
  - Knuth nota: compilatoarele "așteptate" erau imposibil de construit
  - rebotezat de unii în "Itanic"

Încă odată, piața nu are răbdare și alege versiunea pe 64-biți a arhitecturii x86 ca noul ISA

VLIW este relevant pentru aplicații de tip Digital Signal Processing, caracterizate de:

- programe scurte
- instrucțiuni condiționale simple
- lipsă cache

RISC vs CISC în era PC

- CPU-urile CISC reduc diferența de performanță față de RISC
  - echipe de dezvoltare mari (Intel și AMD)
  - beneficiază de viteza crescută a RISC
    - transformă intern, on-the-fly, instrucțiunile în μinstrucțiuni RISC
    - $\Rightarrow$ tehnicile de creștere a performanței specifice RISC sunt acum aplicabile și microprocesoarelor CISC

CISC curent => transforma instructiuni complexe in mai multe microinstructiuni

Vârful erei PC atins în 2011:

- 350 milioane de microprocesoare x86 vândute anual
  - volum mare + profit redus $\Rightarrow$ preț mai mic pentru x86 decât RISC
- produsele SW pentru PC crează o piață imensă
  - piața SW pentru Unix este mult mai diversă, oferind produse pentru diverse arhitecturi RISC (Alpha, HP-PA, MIPS, Power, SPARK)
  - programele pentru PC, comparativ, ofera aplicații "împachetate" compatibile doar cu arhitectura x86
  - $\Rightarrow$ PC a dominat piața calculatoarelor de birou și serverelor de mici dimensiuni ai anilor 2000

Era post-PC:

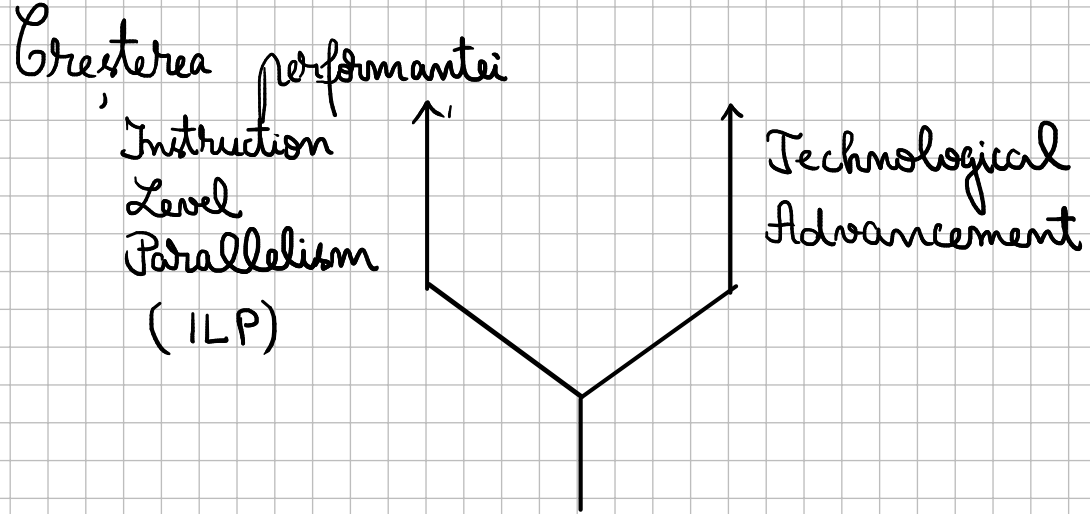
- deschisă de Apple prin lansarea iPhone-ului în 2007
  - în loc să cumpere un microprocesor, Apple dezvoltă propriul System on Chip (SoC) folosind arhitecturi ale altor companii
- proiectanții de dispozitive mobile incep să valorifice reducerea dimensiunii și a puterii consumate
- $\Rightarrow$ CPU-urile CISC sunt dezavantajate

performanță ↘

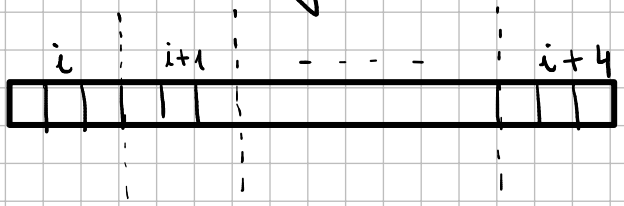
eficiență →

Astazi:

- vanzările x86 au scăzut anual cu 10% începând cu 2011
- vânzările CPU-urilor RISC au explodat la 20 miliarde de unități
  - 99% din CPU-urile pe 32 și 64 de biți sunt RISC



Dezavantaj



Provocări actuale pentru Arhitectura Calculatoarelor

De regulă, producătorul implementeaza un ISA nu construiește unul nou:

- MOS: tehnologia prevalentă de implementare a microprocesoarelor începând cu a doua jumătate a anilor 1970
  - inițial în tehnologie nMOS
  - ulterior în tehnologie CMOS
- evoluția tehnologiei CMOS a cunoscut salturi spectaculoase: legea lui Moore
  - v1: 1965: dublarea anuală a numărului de tranzistori ⇒ rata de creștere anuală de 100%
  - v2: 1975: dublarea la 2 ani a numărului de tranzistori ⇒ rata de creștere anuală de 41%

Întrebare Cum este evaluată o rată de creștere de 40% ?
Răspuns O rată de creștere de 41% este foarte mare!
Exemple de rate anuale de creștere: culturi de porumb - 2%, eficiența generatoarelor electrice - 1.5%, eficiența sistemelor de iluminat - 2.6%, viteza călătoriilor intercontinentale - 5.6%, eficiența consumului de conbustibil pentru autoturismelor - 2.5%

I. ciclu de tact □□□ i

II. ciclu de tact □□□ i+1

- - - - - - - - - - - - -

□□□

41%

(1,41% ; 1,41%) -> la 2ani

=> 2 x tranzistori

Legea lui Moore încetinește în jurul anilor 2000

- în anul 2018 decalaj între nr. de tranzistori previzionat și cel actual a crescut de 15 ori
- decalajul acesta se va adânci mai mult în viitor

Legea lui Robert Dennard (1974)

- efectele miniaturizării tranzistorului
  - tranzistorul devine mai eficient energetic (scade puterea consumată)
  - tranzistorul devine mai rapid (tehnologie CMOS mai performantă)
- trendul de miniaturizare a tranzistorului a încetinit în jurul anului 2007
  - până în 2012 acest trend aproape a dispărut

Creșterea performanței CPU-urilor între 1986 și 2002

- facilitată de următorii factori:
  - tehnologia de integrare mai performantă
  - Instruction Level Parallelism (ILP)
- ⇒ creșterea performanței microprocesoarelor cu aproape 50%

Miniaturizarea tranzistorilor

↗eficiență ↗viteză

41% Tech. Adv.

9% ILP

=> 50% creșterea performanței

Diminuarea efectului legii lui Dennard:

- sunt necesare alte mijloace de creștere a performanței microprocesoarelor

Se declanșază era multi-core!

- problema exploatării paralelismului (la nivel de instructiuni sau date) este transferată către programator și limbaje de programare
- nu rezolva problema consumului de putere
  - fiecare nucleu activ consumă energie

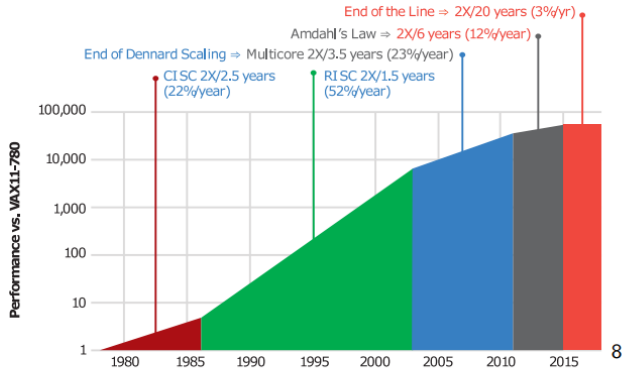
Creșterea numărului de nuclee ⇒ creșterea puterii consumate

- numărul de nuclee este limitat de Thermal Dissipation Power (TDP)
- "dark silicon" era: nucleele inactive nu sunt alimentate

salvează energie

Creșterea performanței CPU-urilor măsurate de SPECint

- Standard Performance Evaluation Company (SPEC)



8

Revenirea la creșterea de performanță a anilor 1980 și 1990:

- abordări arhitecturale noi

Siguranța - tratată superficial

În anii 1970 proiectanții de CPU-uri adaugă măsuri arhitecturale de creștere a securității:

- se consideră că majoritatea defectelor provin din SW
- oferă suport HW pentru detecția lor

Facilitățile de securitate rămân nefolosite de către OS:

- implică costuri de performanță ⇒ sunt eliminate
- în contextul actual, măsurile (modeste) de asigurarea siguranței:
  - suport HW pentru mașini virtuale
  - facilități HW pentru primitive criptografice

Vectorii de atac rezidă acum în HW:

- Procesorul Intel Management Engine (ME):
  - rulează cod pentru mentenanța firmeware-ului având privilegii mai mari decât OS

"Sadly, and most depressing, there is no option for us users to opt-out from having this on our computing devices, whether we want it or not. The author considers this as probably the biggest mistake the PC industry has got itself into she has ever witnessed."
9

Familia de atacuri Spectre:

- vulnerabilitatea se afla în arhitectura microprocesorului, in contrast cu vulnerabilitățile rezidente în HW

Execuția speculativă introduce în multe CPU-uri defecte de securitate nebănuite dar importante:

- Meltdown și Spectre: exploatarea unor vulnerabilități în HW-ul CPU-urilor
  - permite obținerea informațiilor confidențiale cu o viteză de peste 10 Kbit/sec
  - sunt utilizate atacuri de tip "side-channel":
    - informația este "scursă"(leaked) observând durata de execuție a unei sarcini de calcul și convertirea ei în informație utilă
- atacul NetSpectre din 2018:
  - permite obținerea informațiilor la distanță, de la calculatoare conectate într-o rețea locală sau într-un cluster (cloud)

Atacurile "side-channel" nu sunt noi:

- anterior, succesul unui atacator era facilitat de vulnerabilități SW
- Meltdown și Spectre: vulnerabilitatea rezida în implementarea HW
  - ISA nu oferă informații privind efectele "side-channel" ale execuției unei secvențe de instrucțiuni
  - ⇒ regândirea arhitecturii unui calculator

Arhitecturi hardware - oportunități (contin.)

Hardware streamlining:

- favorizeaza calculul paralel
  - aria de Siliciu a tranzistorilor eliminați poate fi folosită pentru adăugarea de noi nuclee ⇒ grad crescut de paralelizare

Localizare: factor esențial în creșterea performanței aplicațiilor:

- abilitatea arhitecturii de a accesa eficient datele [LTEK20]
  - proximitatea localizării în memorie (localizare spațiala)
  - date accesate de curând (localizare temporala)
- aplicațiile cu localizare crescută oferă un grad înalt de paralelism
  - utilizare eficienta a nucleelor CPU-ului

Cum poate fi adresat, în maniera eficientă, paralelismul la nivel de aplicație? [LTEK20]

- simplificarea procesorului
- specializarea de domeniu

Simplificarea procesorului:

- înlocuirea nucleelor complexe cu unele simple
  - eliminarea facilităților costisitoare (destinate accelerării calculului serial)
- aplicația trebuie să suporte un grad ridicat de paralelizare

Specializare de domeniu:

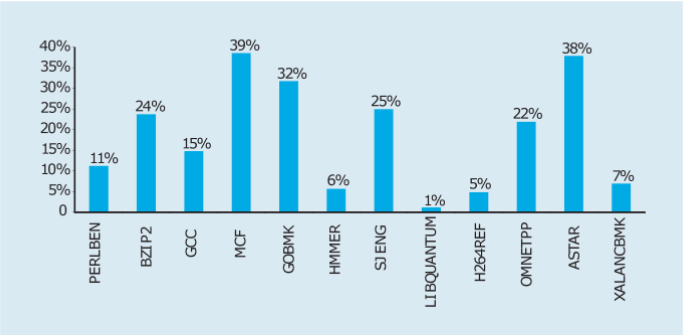
- hardware personalizat pentru un domeniu de aplicații
  - studiu de caz: Graphics Processing Unit (GPU)-uri [LTEK20]
    - compuse din numeroase nuclee optimizate pentru calcule grafice
    - ofera mai multe "benzi paralele" pentru calcul
    - procent din CPU-uri ocupat de GPU: 15-25% în 2010,crește pînă la 40% în 2017
    - În top 500, 10% din supercalculatoarele anului 2012 au avut încorporate acceleratoare (GPU-uri în special), iar pentru anul 2017, procentul a crescut la 38%

Considerații privind simplificarea procesoarelor:

- ILP a fost metoda de creștere a performanței CPU-urilor între 1986 și 2002
  - CPU-urile au exploatat posibilitatea suprapunerii execuției instrucțiunilor
- execuție speculativă
  - CPU-urile fac o predicție și continuă execuția pe o "cale" de cod din mai multe posibile, a cărei efecte le va îndepărta în cazul în care predicția a fost greșită
- execuția speculativă este atât "sursa performanței ILP cât și a ineficienței sale" [HePa19]
  - o cale de execuție corect prezisă poate economisi energie
  - o cale de execuție greșit prezisă trebuie eliminată urmată imediat de reluarea căii corecte, ambele cu consum suplimentar de energie

Pipeline + branchprediction
→ În caz de exec la predicție → Rollback
→ calcul speculativ

Considerații privind simplificarea procesoarelor:



10

Instrucțiuni "irosite":

- procent de instrucțiuni irosite din totalul celor executate
- benchmark-ul SPEC pentru întregi, CPU: Intel Core i7
- în medie, 19% din instructiuni sunt irosite

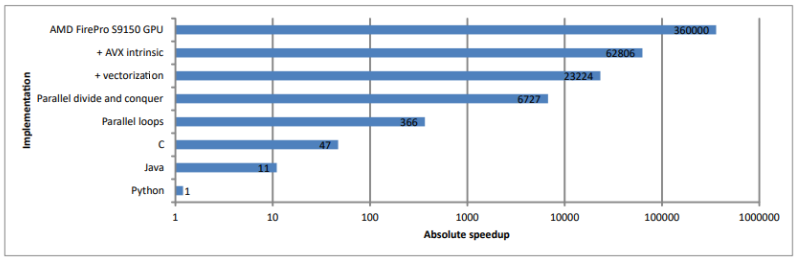
Considerații privind specializarea de domeniu:

- Hennessy și Patterson prezic o tranziție de la arhitecturile "general-purpose" către cele de tip "domain-specific" [LTEK20]
  - pot executa doar câteva sarcini dar extrem de eficient [Hepa18]
  - domeniul trebuie sa suporte paralelizarea
- specializarea domeniului are un efect invers: utilizarea arhitecturii pentru aplicații/domenii noi
  - studiu de caz: utilizarea GPU-urilor:
    - initial, GPU-urile sunt folosite pentru accelerarea calculelor grafice
    - sunt, apoi, adoptate pentru sarcini non-grafice, ex: algebra liniară [LTEK20]
    - GPU-urile sunt instrumentale în revoluția "deep-learning" permițând antrenarea eficientă a modelelor de dimensiuni mari, care ar fi ridicat probleme de performanță pentru CPU-uri [LTEK20]

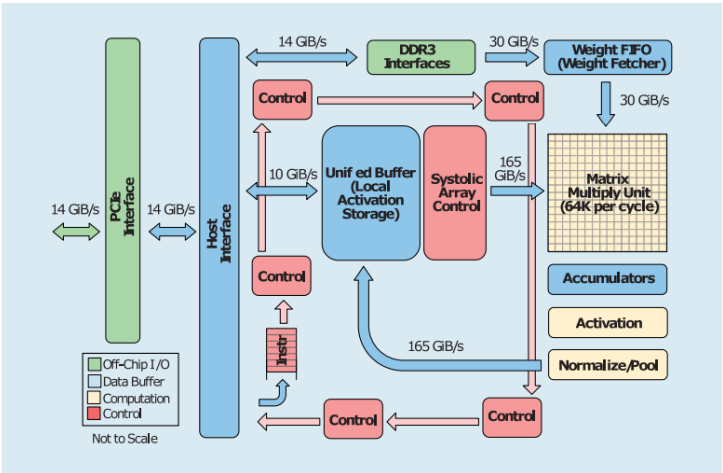
Exemplu de specializare de domeniu: înmulțirea a 2 matrici de dimensiuni 4096x4096

```
1 for i in xrange(4096):
2     for j in xrange(4096):
3         for k in xrange(4096):
4             C[i][j] += A[i][k] * B[k][j]
```

Tehnici de accelerare:



Studiu de caz: Google Tensor Processing Unit (TPU)



Tehnici de accelerare
→ costisitoare
eficiență ↘
vulnerabilități ↗
· hardware streamlining
[LTEK20]
core size ↘ core count ↗

→ simplificare GPU
→ paralelizare

1/5 instrucțiuni executate inutil

GPU → pt. aplicație de alg. liniară + antrenare ML/AI

Arhitectura Google TPU:

- descrisă într-un articol prezentat la conferința International Symposium on Computer Architecture, ediția din 2017
  - unul din cele mai anticipate articole ale conferinței
  - momentan, arhitectura a ajuns la a șasea iterație (Trillium)

Origini [JYPP17]:

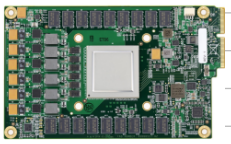
- în 2006, Google analizează puterea computațională necesară pentru rularea unor aplicații specifice, cum sunt Deep Neural Network (DNN)
  - acestea puteau fi executate folosind HW-ul existent (cu cost zero)
  - de ce să fie accelerate?
- în 2013, utilizatorii Google folosesc căutare vocala cel puțin 3 minute, zilnic
  - utilizeaza modele DNN pentru recunoaștere vocală
  - concluzie: un astfel de comportament al utilizatorilor necesită o capacitate dublă de calcul (dublarea datacenter-urilor)
    - costisitor daca sunt utilizate CPU-uri

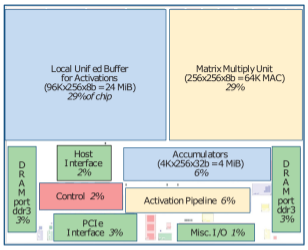
TPU:

- design Application Specific Integrated Circuit (ASIC) specializat
  - 256 x 256, 8-bit, Matrix Multiply Unit
  - 28MiB memorie internă
  - 92 Tera Operations per Second (TOpS)
  - accelerare între 15 și 30 ori comparativ cu CPU/GPU
  - eficiență energetică (TOpS/Watt) între 30 și 80 ori mai bună comparativ cu CPU/GPU

Accelerarea operațiilor DNN:

- quantization: transformarea numerelor de virgulă flotantă la întregi de lățime redusă (8-bit)
- înmulțirea 8-bit int vs 16-bit float
  - arie de 6 ori mai mică
  - energie cunsumată de 6 ori redusă
- adunare 8-bit int vs 16-bit float
  - arie de 38 ori mai mică
  - energie consumată de 13 ori redusă





14

Facilități arhitecturale:

- Matrix Multiply Unit
  - *elementul esențial al unității*
  - rețea de 65'536, unități MAC pe 8-bit
  - arhitetură sistolică pentru accelerarea înmulțirii matricilor
- **Adunarea și înmulțirea întregilor**: elementele esențiale ale accelerarii calculelor de către TPU