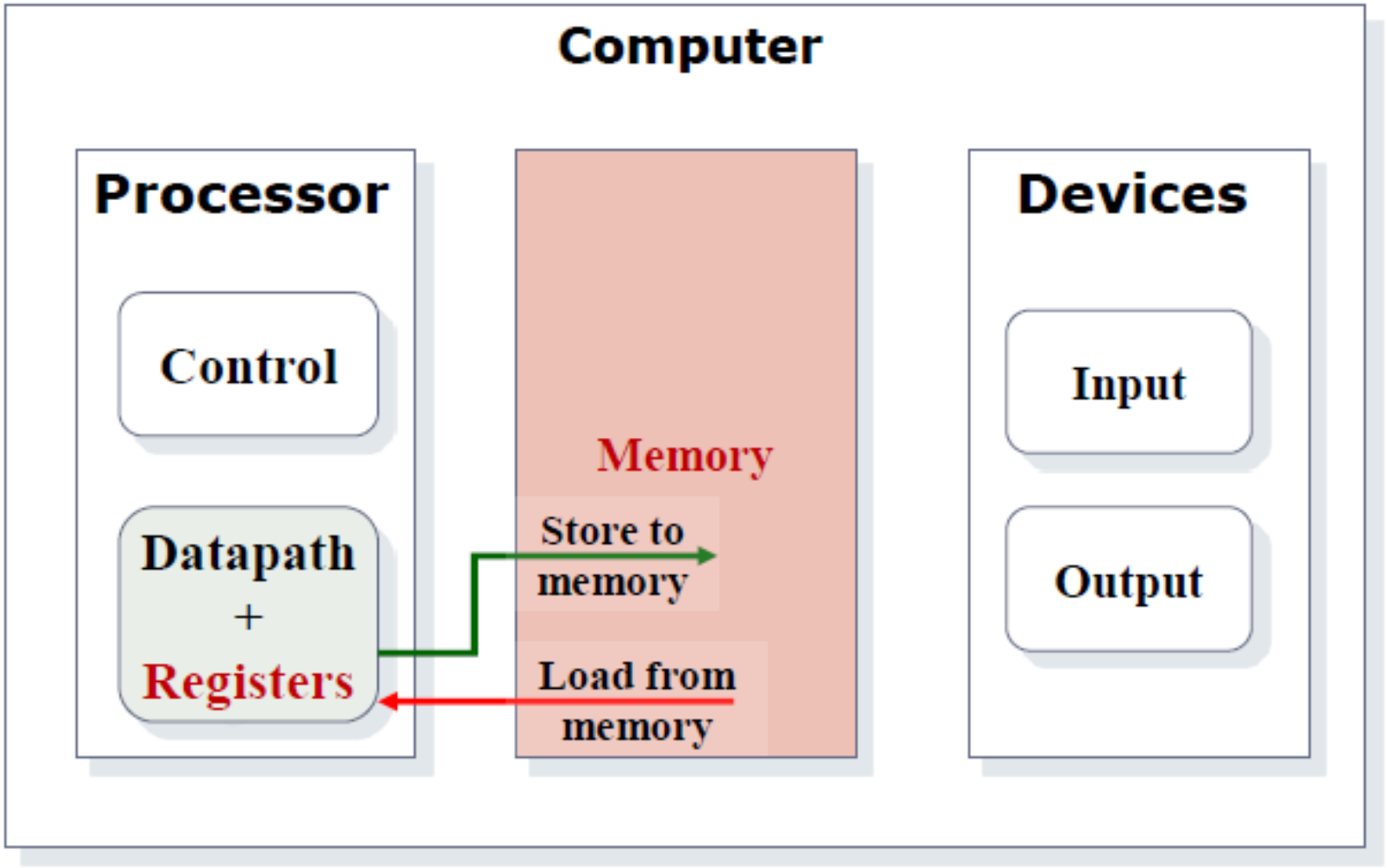
ARKITEKTURE KOMPJUTERI

LEKSIONI IV

Memoria, tipet e memorieve, Pipelining

Lektor: Alban Deda

Memoria


Computer
Processor
Control
Datapath
+
Registers
Memory
Store to
memory
Load from
memory
Devices
Input
Output

Memoria

- Nevojitet memorje me kapacitet të madh, kosto të ulët dhe që aksesohet shpejt!
- Si realizohet?
  - *Shfrytëzohet fakti që jo gjithë të dhënat dhe instruksionet aksesohen vetëm një herë dhe me të njëjtin probabilitet.*

Memoria – Parimi i Lokacionit

- **Lokalitet në kohë:** nëse një e dhënë është aksesuar, atëhërë ka një probabilitet të madh që kjo e dhënë të aksesohet sërisht.
- **Lokalitet në hapësirë:** nëse një e dhënë është aksesuar, atëherë ka një probabilitet të madh që edhe të dhënat që ndodhen afër saj të aksesohen së shpejti.

Memoria

Imagjinoni sikur të duhet të vendosni çdo libër në vend përpara se të merrni një libër tjetër.

ZGJIDHJA

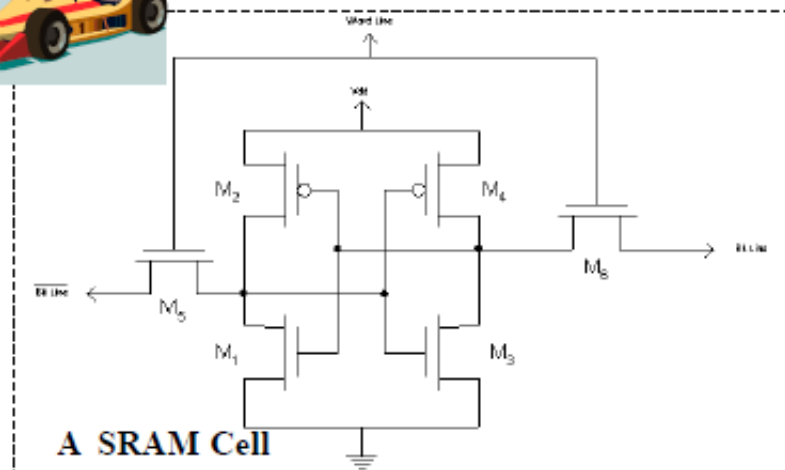
Po sikur të mbajmë në tavolinë disa libra që mendojmë se do na duhen sërisht?

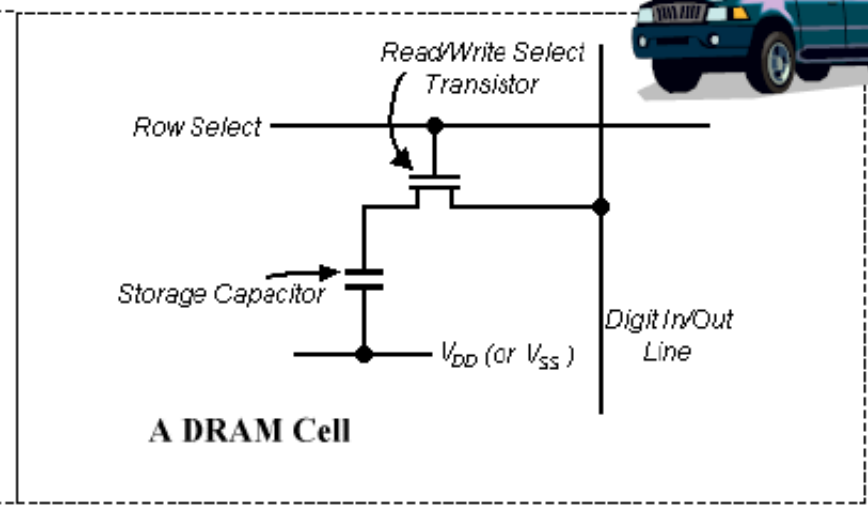
Hierarkia e Memories

- Eshtë një strukturë që përdor disa nivele mmorjeje; sa më shumë rritet distanca nga procesori aq më shumë rritet madhësia e memorjes dhe koha e aksesimit.
- Shfrytëzon **parimin e lokalitetit**
- Ka tre lloje teknologjish për ndërtimin e memorjeve.

	Kapaciteti	Koha mesatare e aksesimit	Kosto/GB (në 2008)
SRAM	100 KB	0.5-5 ns	$2000-$5000
DRAM	100 MB	50-70 ns	$20 - $70
Hard Disk	100 GB	5-20 ms	$0.20 - $2

Memoriet SRAM vs DRAM





SRAM

6 transistorë / qelizë memorjeje

→ **Densitet i ulët**

Aksesim i shpejtë: vonesa 0.5 – 5 ns

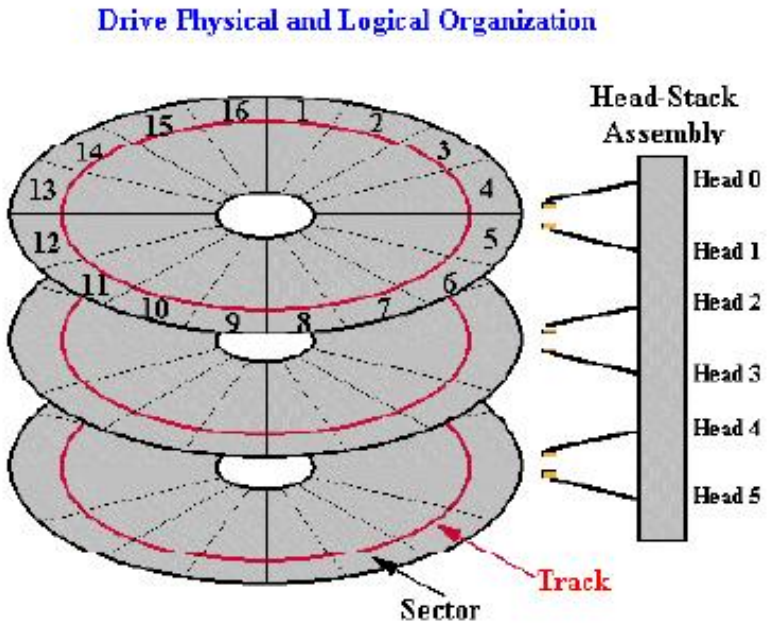
DRAM

1 transistor / qelizë memorjeje

→ **Densitet i lartë**

Aksesim i ngadaltë: vonesa 50-70ns

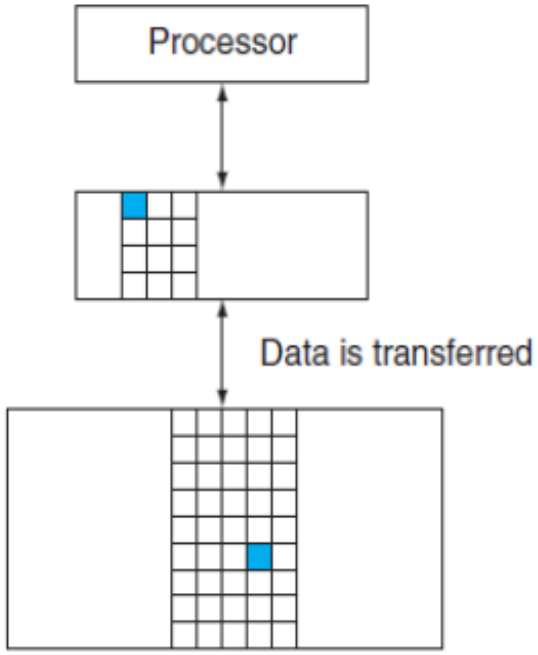
Memoria – Magnetic Disk



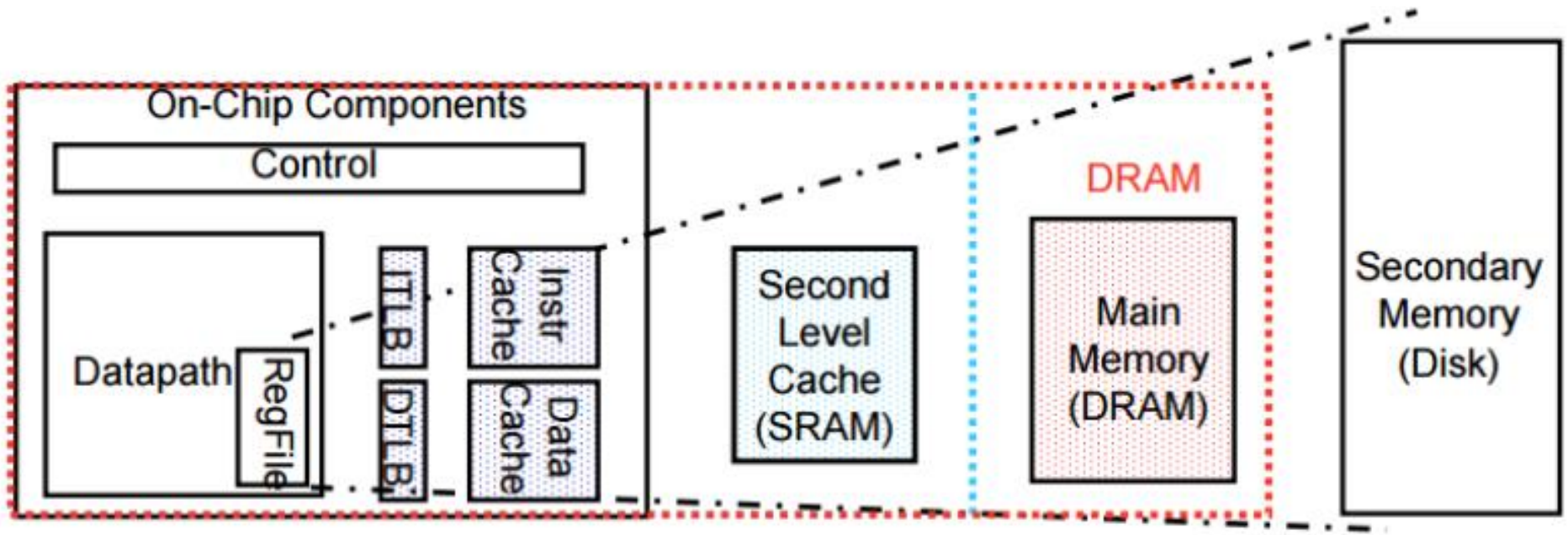
Vonesa mesatare: 5 - 20 ms

Memoria

- Kopjimi i të dhënave bëhet midis dy niveleve që janë afër.
- Sasia më e vogël e informacionit që mund të zhvendoset quhet **bllok** (madhësia e tyre është fuqi e 2-it).
- Nëse të dhënat e kërkuara nga proçesori gjenden në nivelin e sipërm atëherë kemi hit; në të kundërt kemi miss.



Memoriet



Speed (%cycles):	½'s	1's	10's	100's	1,000's
Size (bytes):	100's	K's	10K's	M's	G's to T's
Cost:	highest				lowest

CACHE – Parimi

- Ruaj të dhënat e përdorura së fundmi në një memorje **të vogël por të shpejtë.**
- Shko tek memoja e madhe dhe e ngadaltë:
  - Vetëm kur nuk gjen dot të dhënat/instruksionet në memorjen më të shpejtë
- Si funksionon?

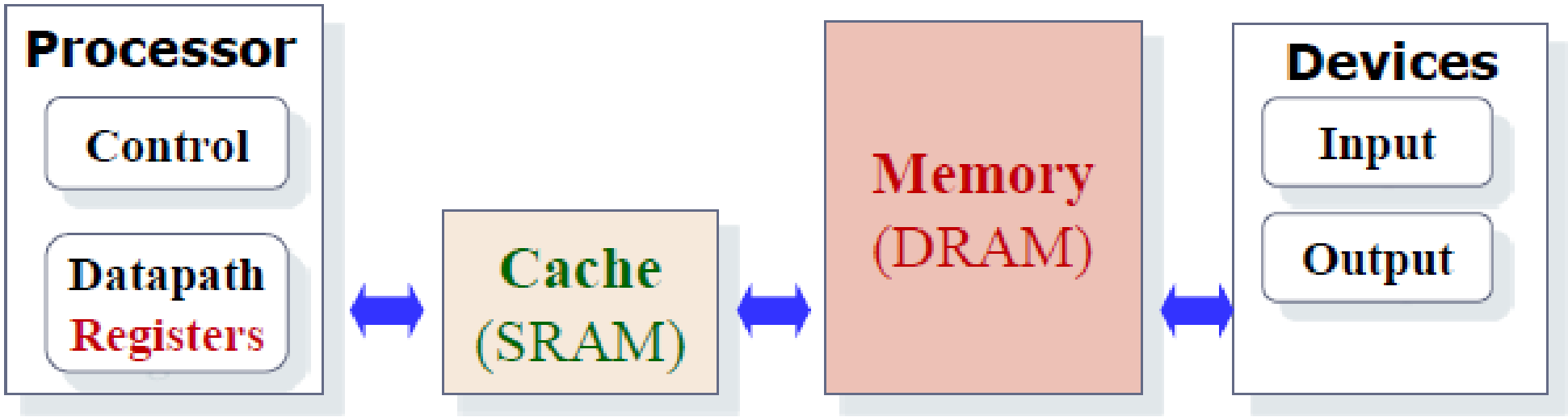
Parimi i Lokalitetit

Programi akseson vetëm një pjesë të vogël të hapësirës së adresimit të memorjes brenda një intervali të shkurtër kohe.

CACHE

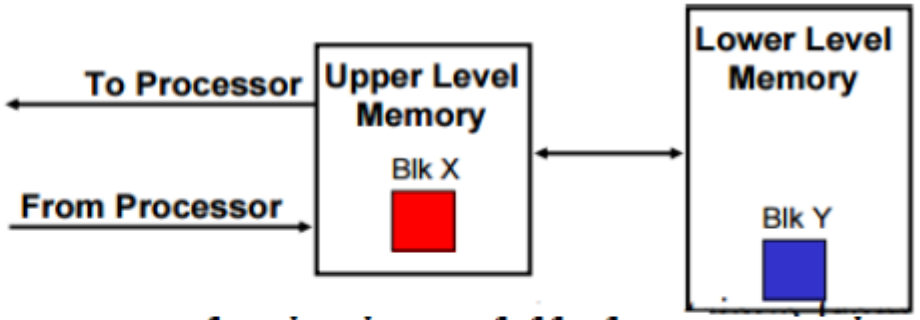
- Memorja midis proçesorit dhe memorjes kryesore.
- Mban të dhëna që janë aksesuar edhe që mendohet se do të aksesohen së shpejti (parimi i lokalitetit)
- Zakonisht ka disa nivele cache-je
  - Niveli i parë L1 ndodhet ne proçesor
  - Niveli i dyte L2 ndodhet midis proçesorit dhe memorjes kryesore
- Cache-ja ndërtohet nga teknologjia SRAM

CACHE


Processor
Control
Datapath
Registers
Cache
(SRAM)
Memory
(DRAM)
Devices
Input
Output

CACHE – Hit vs Miss

- Hit Rate: përqindja e aksesimeve të memorjes që gjenden në nivelin e sipërm.
- Hit Time: koha për të aksesuar nivelin e sipërm të hierarkisë, duke përfshirë edhe kohën që duhet për të përcaktuar nëse blloku ndodhet apo jo në atë nivel.
- Miss Rate: 1-(Hit Rate)



- Miss Penality: koha për zëvendësimin e bllokut në një nivel hierarkie më të lartë edhe koha për kalimin e bllokut te proçesori.

Pipelining

Permiresimi i Shpejtesise

- Synimi kryesor i zhvilluesve te teknologjive te ndryshme eshte : PERMIRESIMI I KOHES se ekzekutimit
- Permiresimi behet ne disa nivele (sic quhet random, cilat jane parametrat e PC)
- PROCESORI eshte nje nder parametrat kryesore qe duhet permiresuar.
- Procesori proceson INSTRUKSIONET (ku ndodhen instruksionet ?)

Instruksionet

- Instruksionet ndodhen ne ISA (makina virtuale ndermjet Hardware dhe Software).

- Kemi thene – per instruksione te cilat kerkohen me shpesh, Procesoret i ruajne keto instruksione ne regjistrat e tyre, pa i kaluar ne Memoriet e tjera.

- Sa me i madh numri i instruksioneve per te kryer nje veprim, aq me e madhe koha e realizimit te atij veprimi, pasi – per cdo instruksion pasardhes, duhet te presim qe te kryhet instruksioni paraardhes.

Cilat eshte zgjidhja qe te shkurtojme kete kohe, te ekzekutimit te instruksioneve, njeri pas tjetrit ?

Pipelining

Pipelining quhet :

Teknike implementimi, ku disa instruksione MBIVENDOSEN gjate ekzekutimit

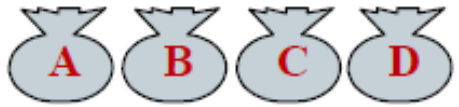
Pipelining analogji

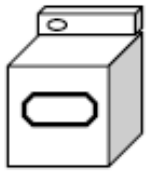
Pastrimi i rrobave

- Katër hapa:
  1. Larja në lavatriçe
  2. Tharja në tharëse
  3. Palosja/Hekurosja
  4. Sistemimi nëpër vendet e caktuara

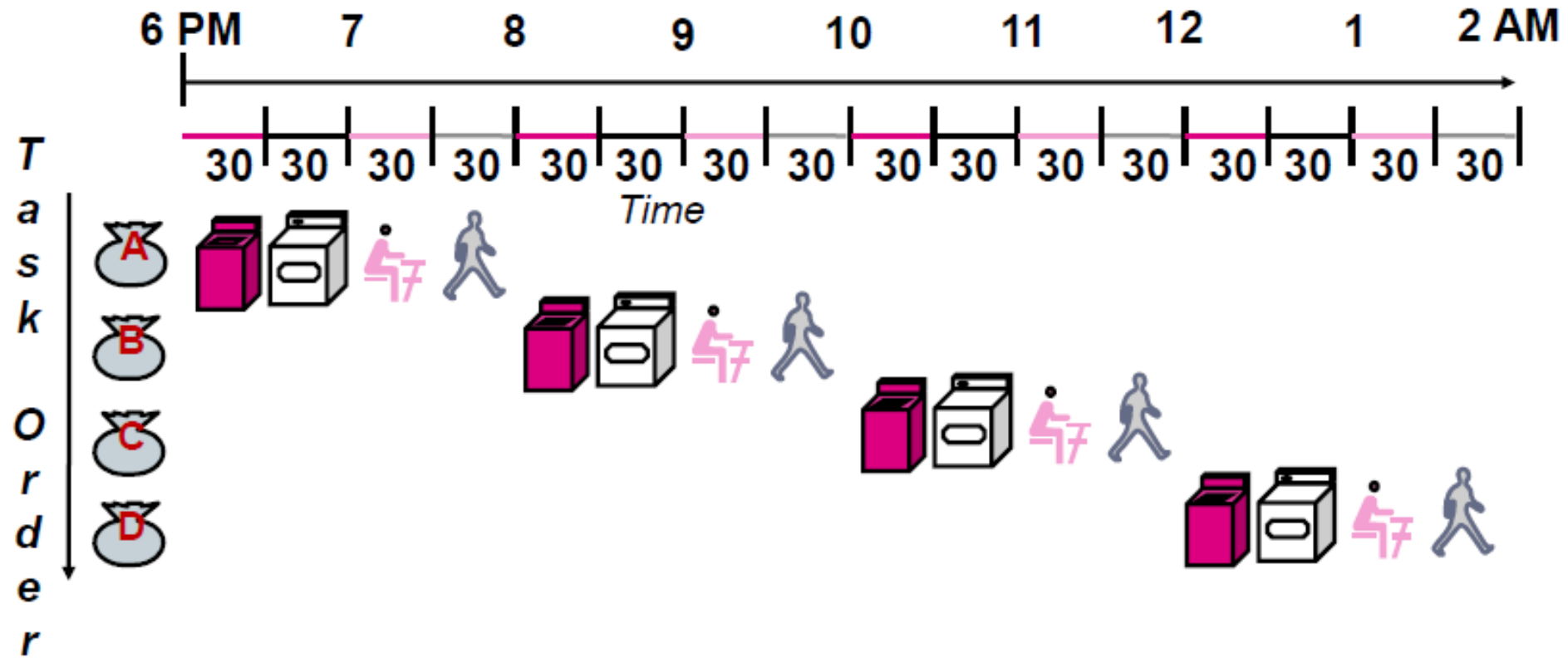
Pipelining analogji

- **4 ngarkesa me rroba të papastra**
- **Larja** 30 minuta
- **Tharja** 30 minuta
- “**Palosja**” 30 minuta
- “**Sistemimi**” 30 minuta





Menyra Sekuenciale



- Mënyra sekuenciale kërkon 8 orë për 4 ngarkesa

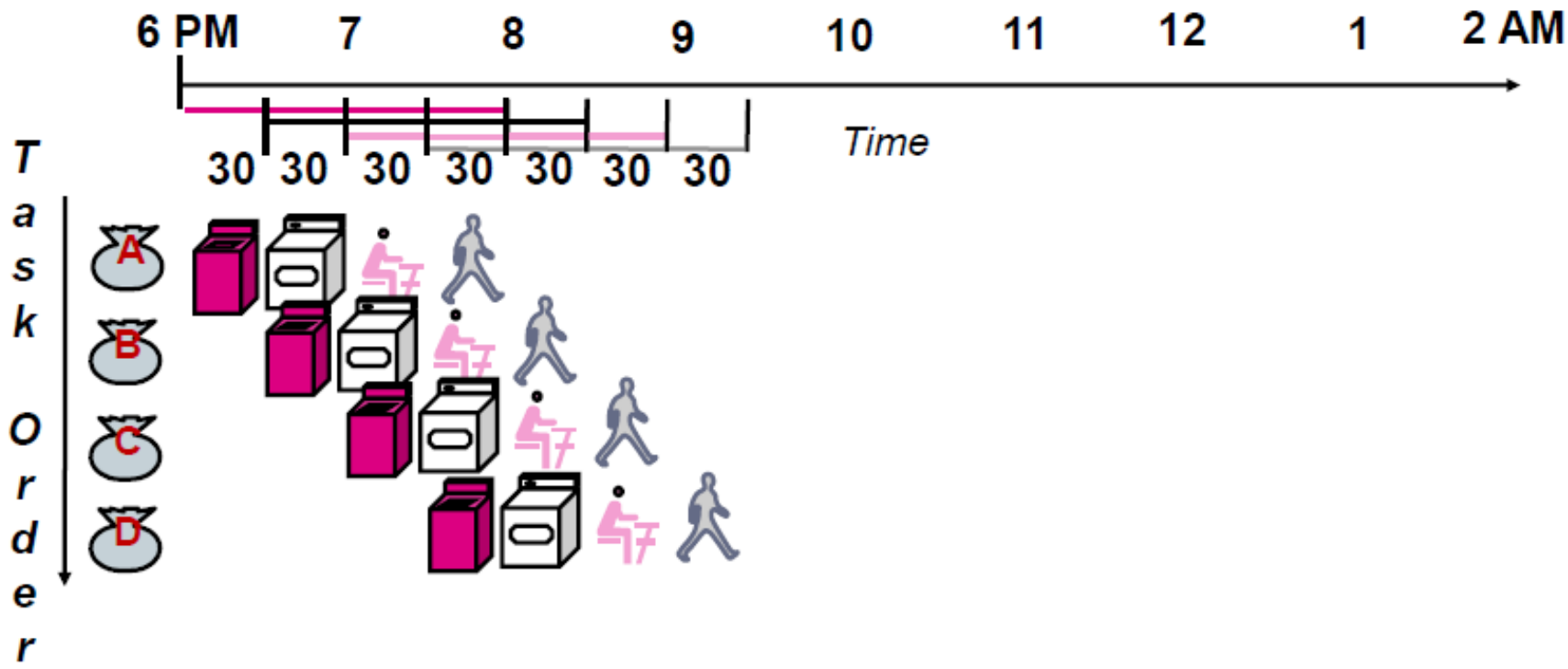
Permiresimi i kohes

- Nese koha e ekzekutimit te nje procesi, nuk mund te shkurtohet, atehere zgjidhja per permiresimin e kohes eshte MBIVENDOSJA

- Pra procesi i mbivendosjes presupozon qe : procesi i pare i ngarkeses se dyte, fillon ne momentin qe ka mbaruar procesi i pare i ngarkeses se pare.

- Pra larja e dyte, fillon kur ka mbaruar larja e pare
- Tharja e dyte, fillon kur ka mbaruar tharja e pare
- E keshtu me radhe

Permiresimi i kohes me Pipelining



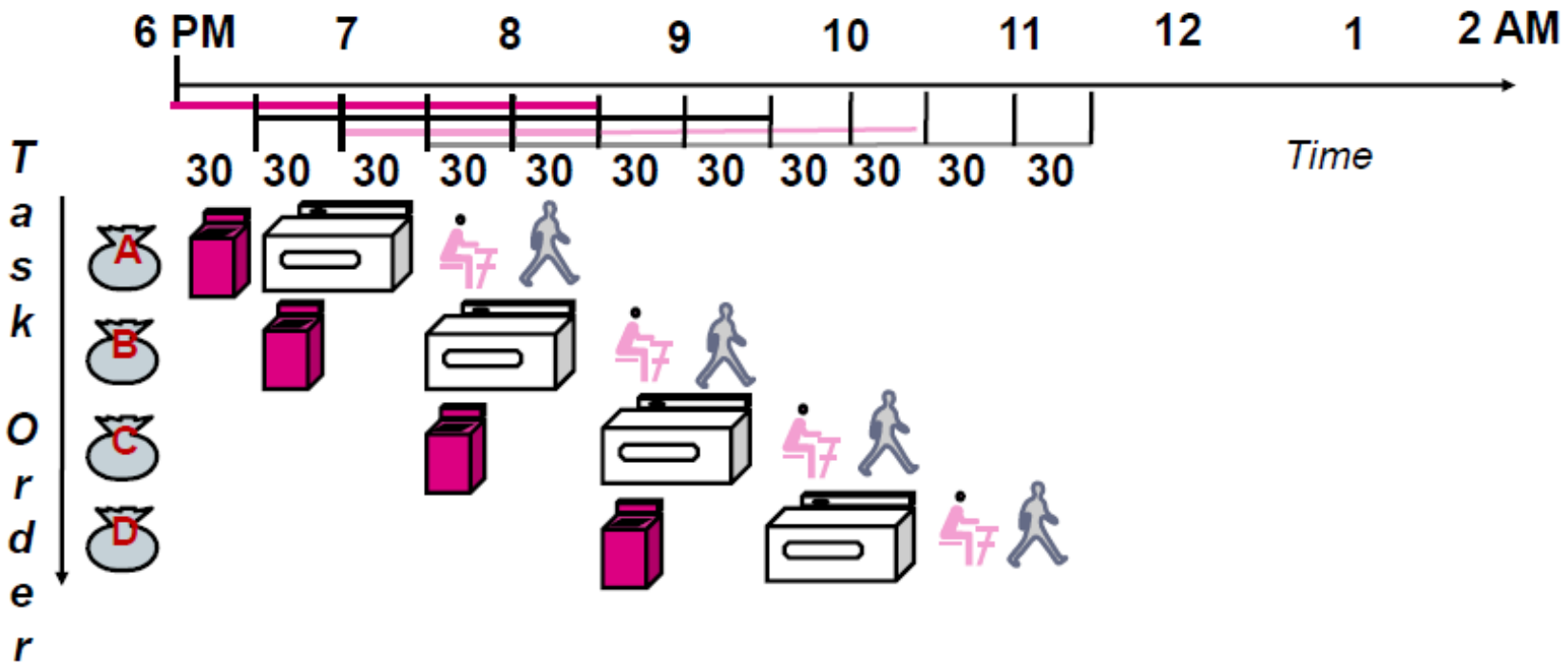
- Mënyra me pipelining kërkon 3.5 orë për 4 ngarkesa!
- 1 ngarkesë çdo 30 min
- **Speedup i mundshëm = 2 orë/30 min = 4 (nr i fazave)**

Pipelining – Pros vs Cons

Avantazhi/Limitimi

- (+) Mund të bëhet më shumë punë në më pak kohë
- (-) Procesi ngadalësohet nga faza më e gjatë

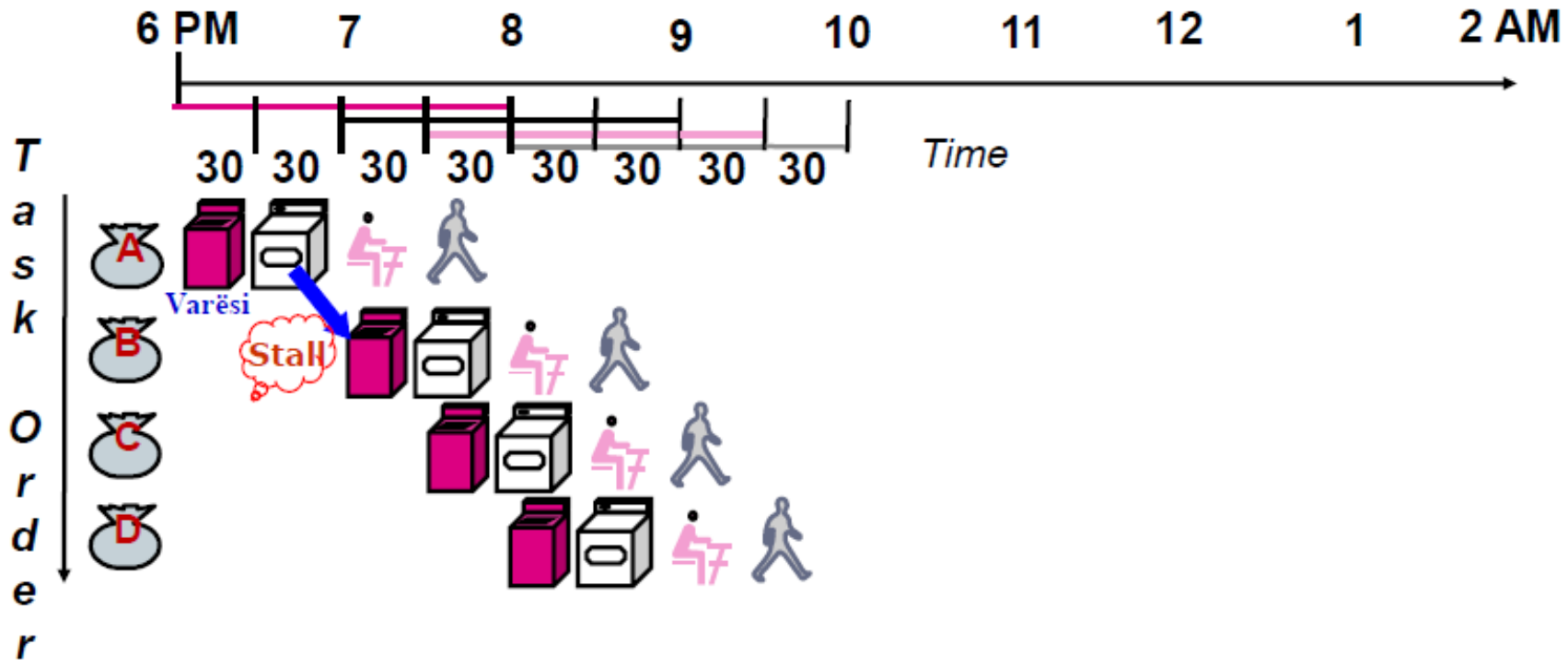
Pipelining – Cons



- **Mënyra me pipeline tani kërkon 5.5 orë!**
- Një ngarkesë në një orë (shpejtësia e tharëses)!
- **Proçesi ngadalësohet nga faza më e gjatë!**

Pipelining

Po nëse ka **varësi**?



- Supozojmë se herën e parë duhet të kontrollohet rezultati pas tharjes që të fillojë larja pasardhëse.
- **Mënyra me pipelining tani kërkon 4 orë për 4 ngarkesa!**

Pipelining tek Procesoret

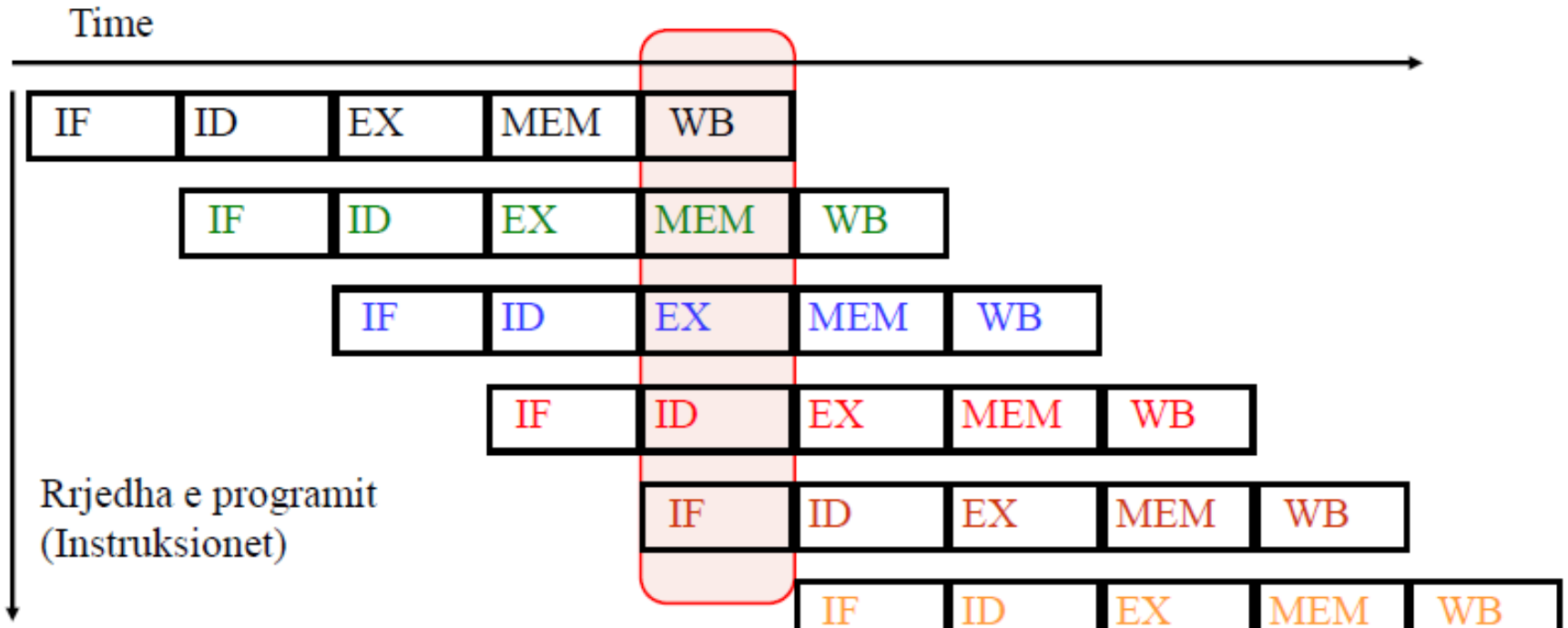
Pesë faza ekzekutimi

- **IF**: Fetch i instruksioneve nga memorja
- **ID**: Leximi i regjistrave gjatë kohës që bëhet dekodimi
- **EX**: Ekzekutimi i veprimit / llogaritja e adresës
- **MEM**: Aksesimi i një operand-i nga memorja
- **WB**: Shkrimi i rezultatit në regjistër

Ideja e pipelining:

- **Fazat të mbivendosen.**
- **Çdo fazë kërkon 1 cikël clock-u.**

Ekzekutim me Pipeline



Disa instruksione ndodhen në pipeline në të njëjtën kohë!

Konkluzion

- Me pipeline disa detyra kryhen në të njëjtën kohë pasi përdorin burime të ndryshme.
- Pipeline rrit performancën pasi rrit **throughput-in** e instruksioneve
- Pipeline **nuk ndryshon** kohën e ekzekutimit të një instruksioni (latency)
- Throughput-i është i rëndësishëm pasi kompjuterat ekzekutojnë biliona instruksione!

Rreziqet e Pipelining

Rreziqet e pipeline

- **Rrezik (Hazard):** instruksioni pasardhës nuk ekzekutohet dot në ciklin pasardhës
- Ekzistojnë tre lloje rreziqesh
  - Structural Hazards
  - Data Hazards
  - Control Hazards

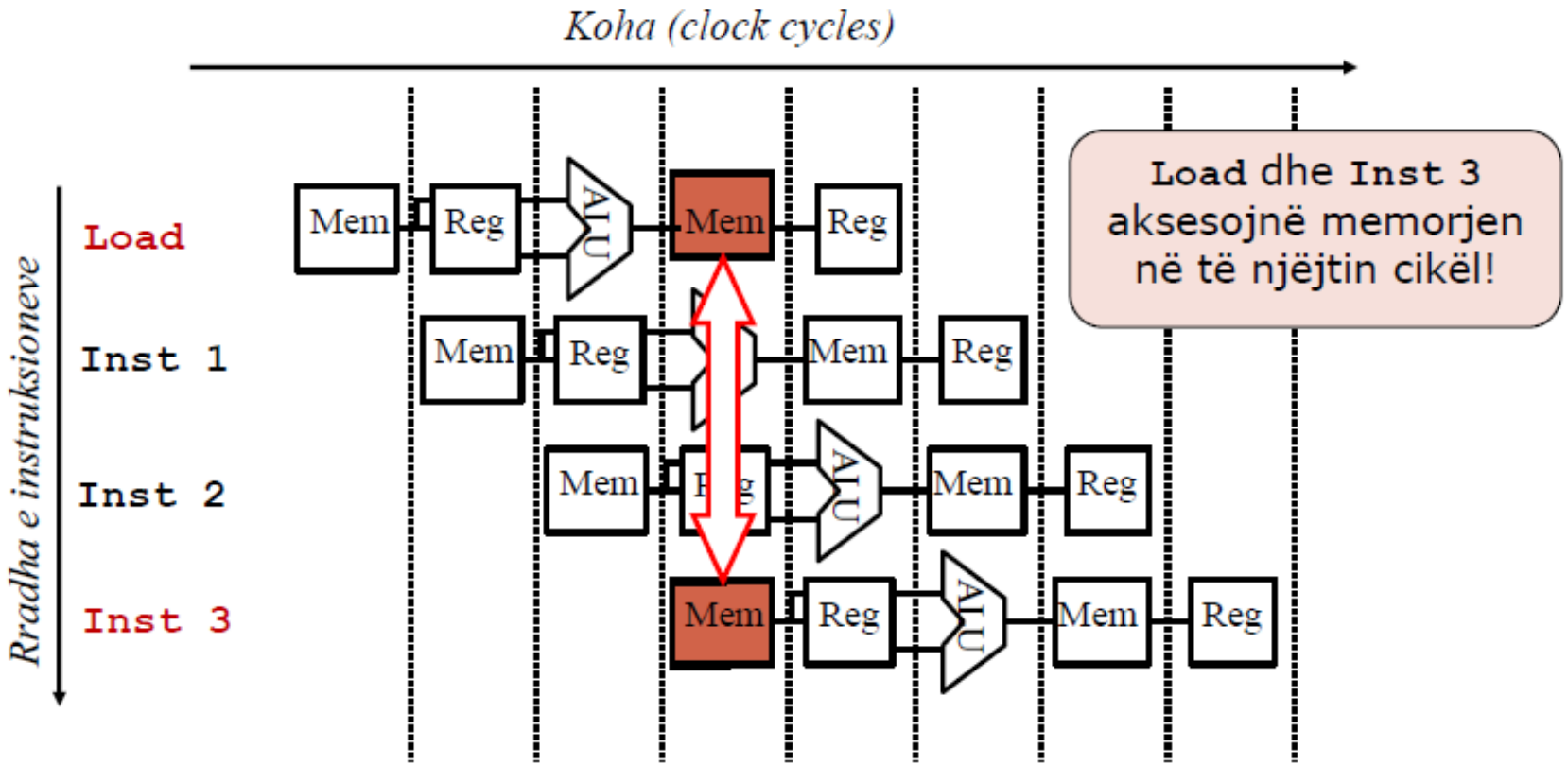
} Varësi midis instruksioneve

Structural Hazards

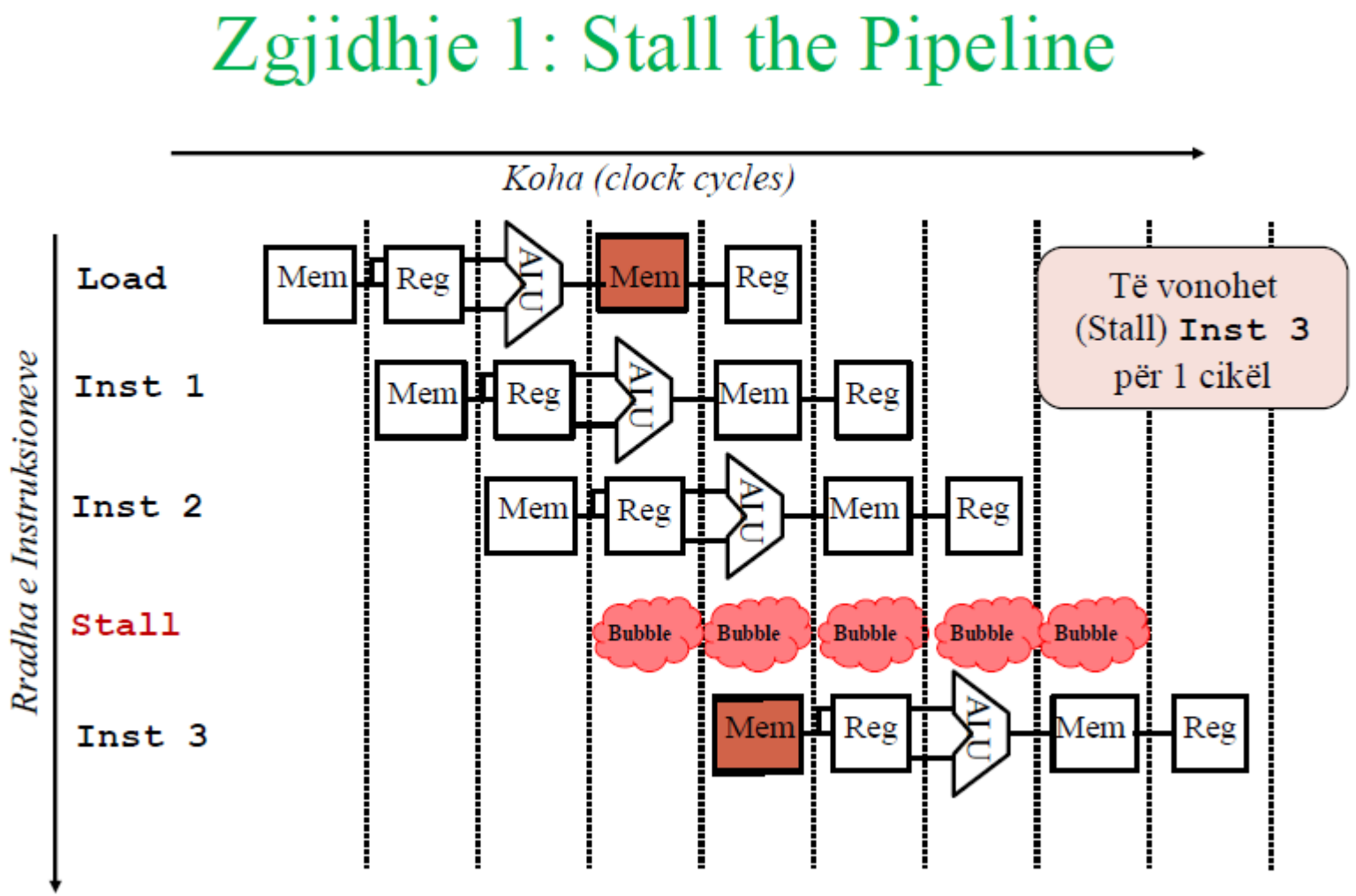
- Hardware nuk mund të ekzekutojë kombinimin e instruksioneve në të njëjtin cikël clock-u.
- Analogji: Nëse do të kishim një makinë që bën larje/tharje *(atëherë larja dhe tharja nuk mund të bëheshin në paralel)*

Structural Hazards

- Supozojmë se ka vetëm një **modul memorje:**



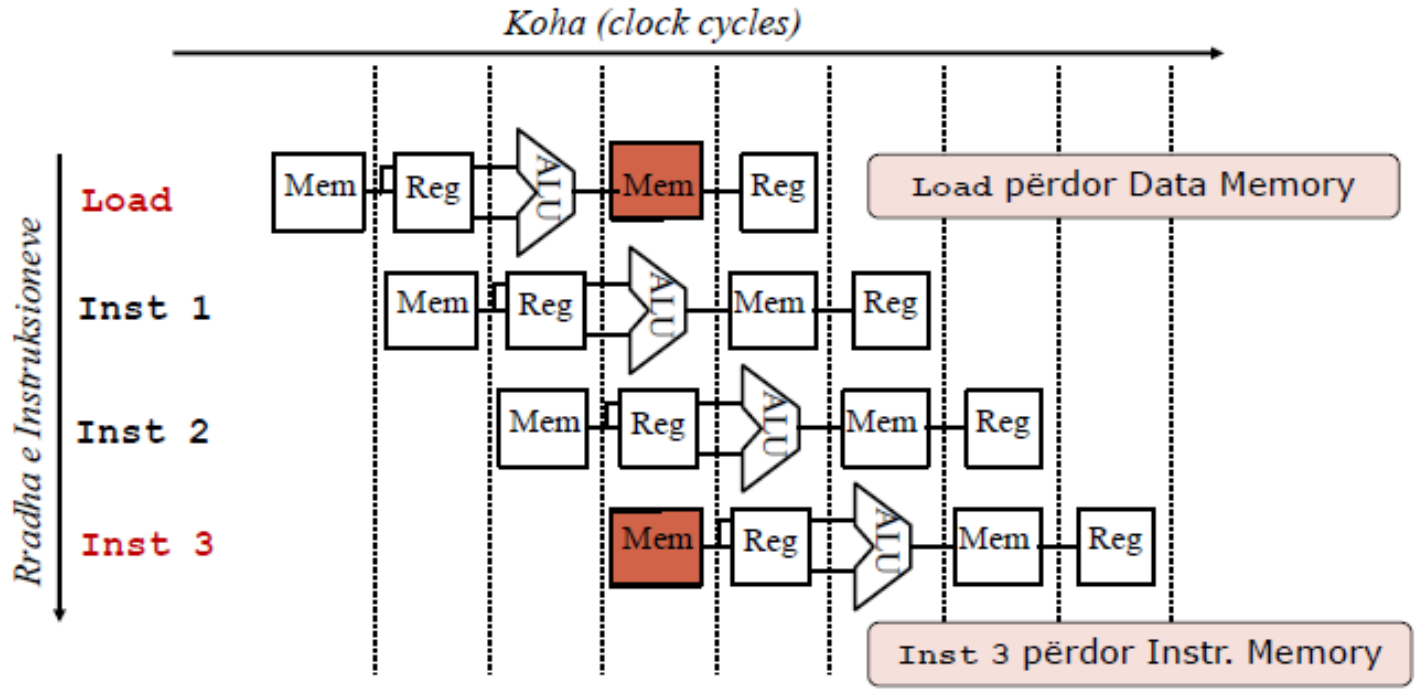
Structural Hazards – Zgjidhjet



Structural Hazards – Zgjidhjet

Zgjidhje 2: Të ndahet memorja

- Të ndahet memorja në:
  - **Data** dhe **Instruction** memory



Data Hazards

- Kur instruksione të ndryshme aksesojnë (read/write) të njëjtin regjistër.
  - Përmbajtja e regjistrit është shkaku i varësive
  - Njihet si **data dependency**
- Pipeline qëndron në pritje sepse një fazë duhet të presë për një tjetër që të përfundojë.

Data Hazards – Zgjidhja

- Me ane te **forwarding** os **bypassing**

- Kemi rastin kur dy veprime te njepasnjeshme, te cilat po ekzekutojne pipelining :

- Mbledhja dhe Zbritjen duhet te aksesojne operandat nga i njejti regjister

  – Sapo ALU të llogarisë shumën, kjo shumë mund të merret si input për zbritjen (para se të shkruhet në regjistër): **forwarding ose bypassing**