8. Fejezet

Processzor (CPU) és memória: tervezés, implementáció, modern megoldások

The Architecture of Computer Hardware and Systems Software: An Information Technology Approach

3rd Edition, Irv Englander

Wilson Wong, Bentley College

Linda Senne, Bentley College

CPU architektúrák

- CISC – Complex Instruction Set Computer
- RISC – Reduced Instruction Set Computer
- CISC és RISC összehasonlítása
- VLIW – Very Long Instruction Word
- EPIC – Explicitly Parallel Instruction Computer

CISC architektúra

- Példák
  - Intel x86, IBM Z-széria, idősebb CPU architektúrák
- Tulajdonságai
  - Néhány általános célú regiszter
  - Több különböző címzési mód
  - Nagyszámú speciális, összetett utasítások
  - Változó méretű utasítások

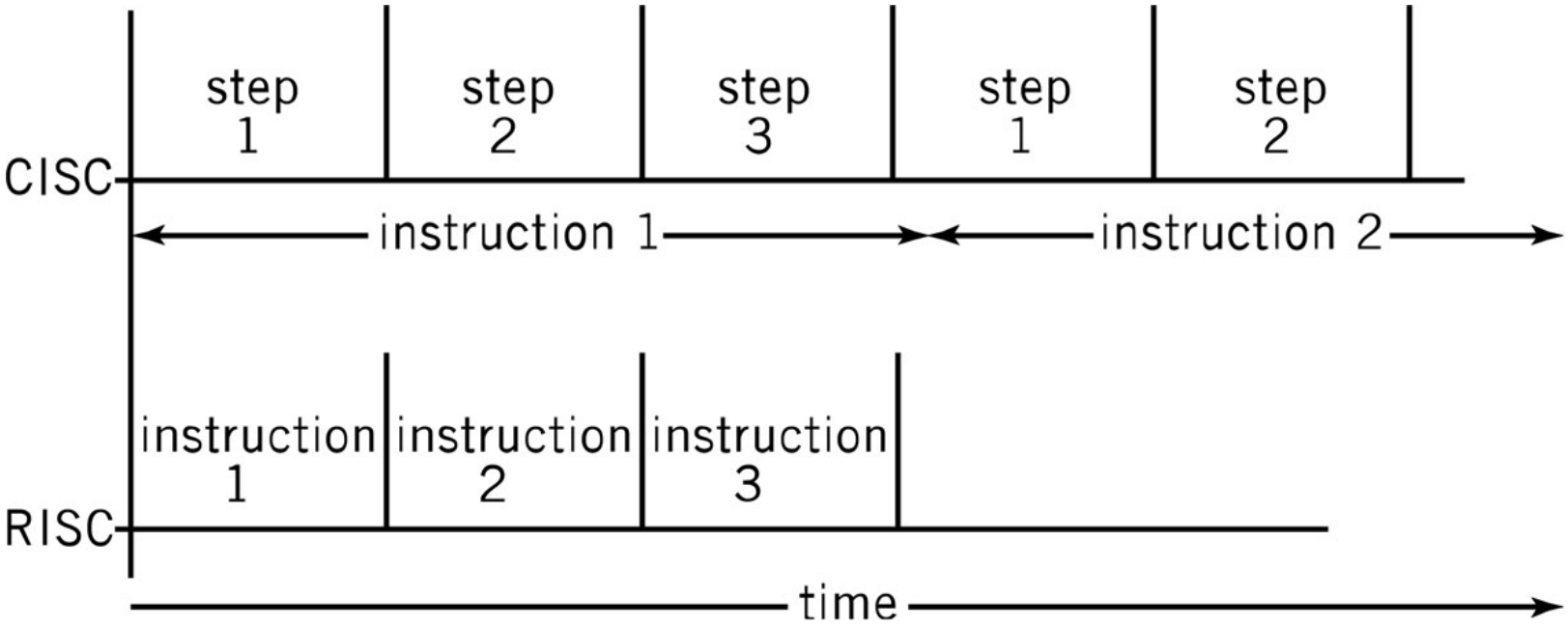
CISC architektúra korlátai

- Az összetett utasításokat ritkán használják a programozók és fordítók
- Memóriát használó utasítások (load és store) végrehajtása lassú pedig ezek teszik ki utasítások jelentős részét
- Eljárás, ill. függvény hívások problémát okoznak
  - Hívási paraméterek átadása
  - Regiszterek tartalmát el kell menteni és vissza kell állítani a hívások után

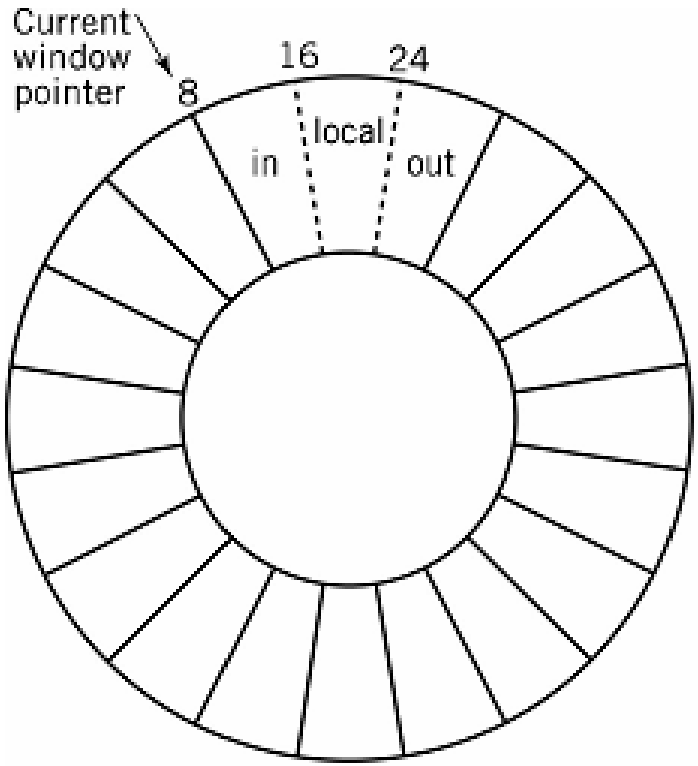
A RISC tulajdonságai

- Példák
  - Power PC, Sun Sparc, Motorola 68000
- Korlátozott és egyszerű utasításkészlet
- Fix hosszúságú, meghatározott formátumú utasítás szavak
  - Pipeline feldolgozás lehetősége, párhuzamos fetch és execution (azonos feldolgozási idő)
- Korlátozott számú címzési mód
  - Egyszerűbbé teszi a megvalósító hardvert (CPU)
- Regiszter-orientált utasításkészlet
  - Csökkenti a memória hozzáférések számát
- Nagyszámú regiszter
  - Csökkenti a memória hozzáférések számát
  - Hatékony a szubrutin (alprogram) hívások végrehajtása

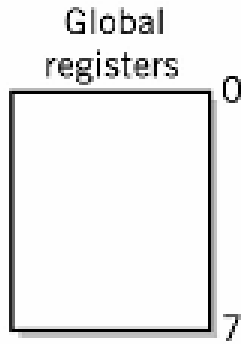
CISC vs. RISC feldolgozás



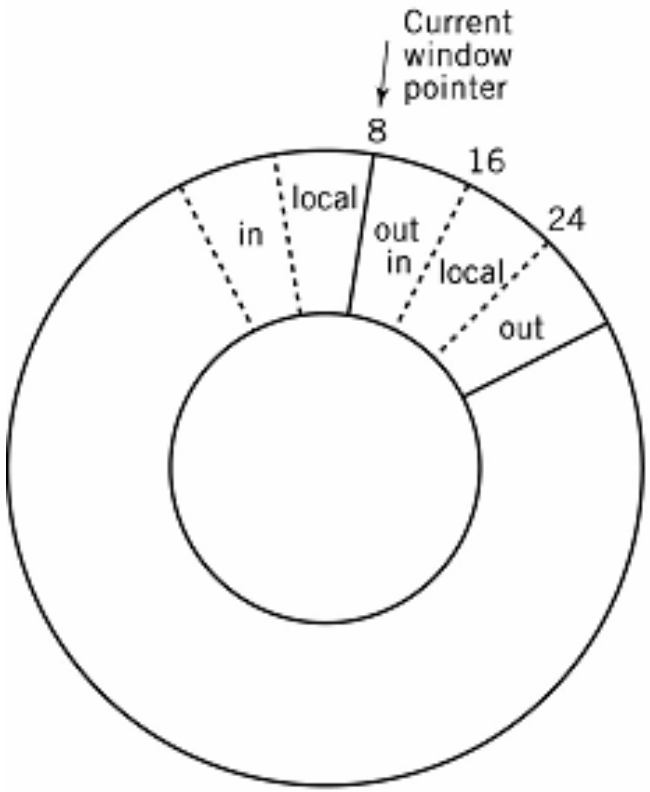
Körkörös regiszter puffer

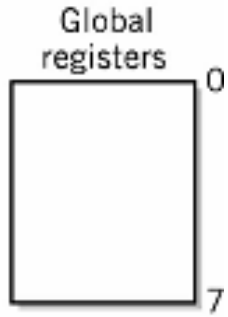


a. Current window



Körkörös regiszter puffer - Szubrutin hívás után





b. After window shift

CISC és RISC teljesítményének összehasonlítása

- RISC → Egyszerűbb utasítások
  → több utasítás
  → több memória hozzáférés
- RISC → nagyobb a bus forgalom és nagyobb a cache „hibák” valószínűsége
- Több regiszter növelhetné a CISC teljesítményét de nincs számukra hely
- Modern CISC és RISC architektúrák egyre hasonlóbbak lesznek

Modern megoldások az utasítás végrehajtás meggyorsítására

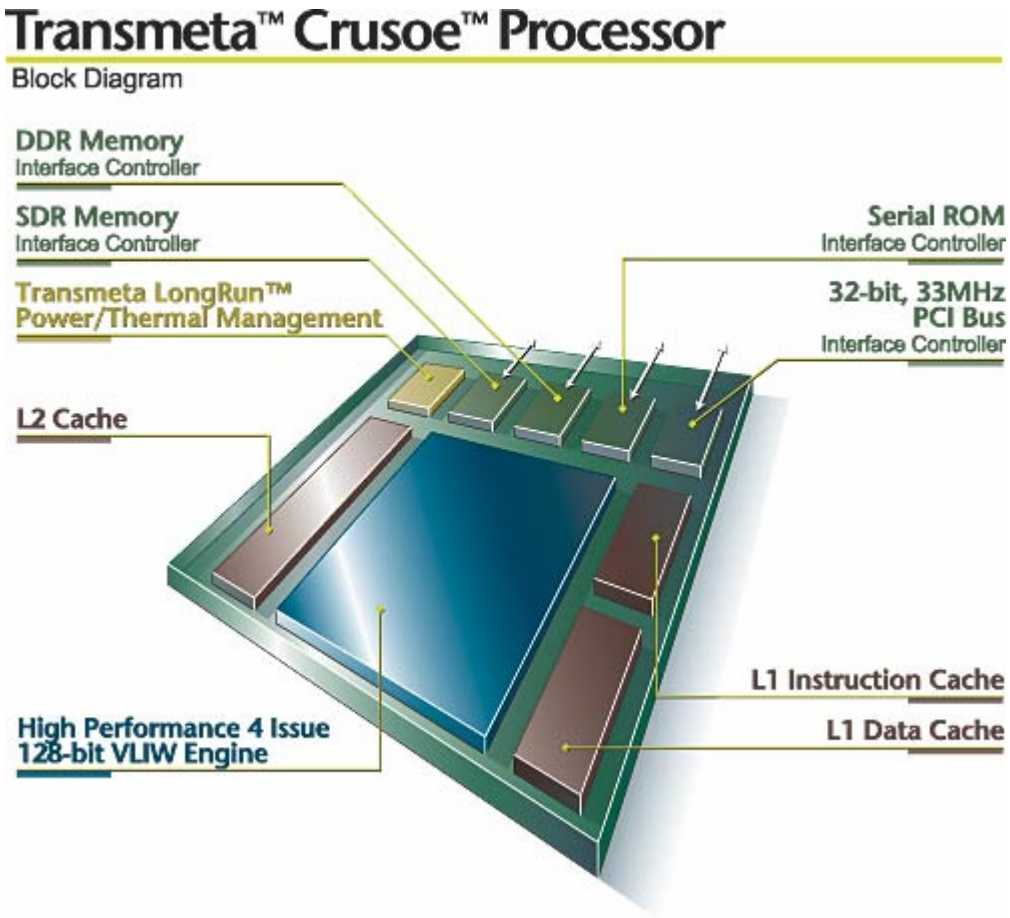
- VLIW – Very Long Instruction Word
- EPIC – Explicitly Parallel Instruction Computer
- A gyorsítás módja:
  - utasítások párhuzamos végrehajtása
- Probléma:
  - utasítások végrehajtás függ egymástól:
    - adat függés
    - vezérlés függés

VLIW architektúra

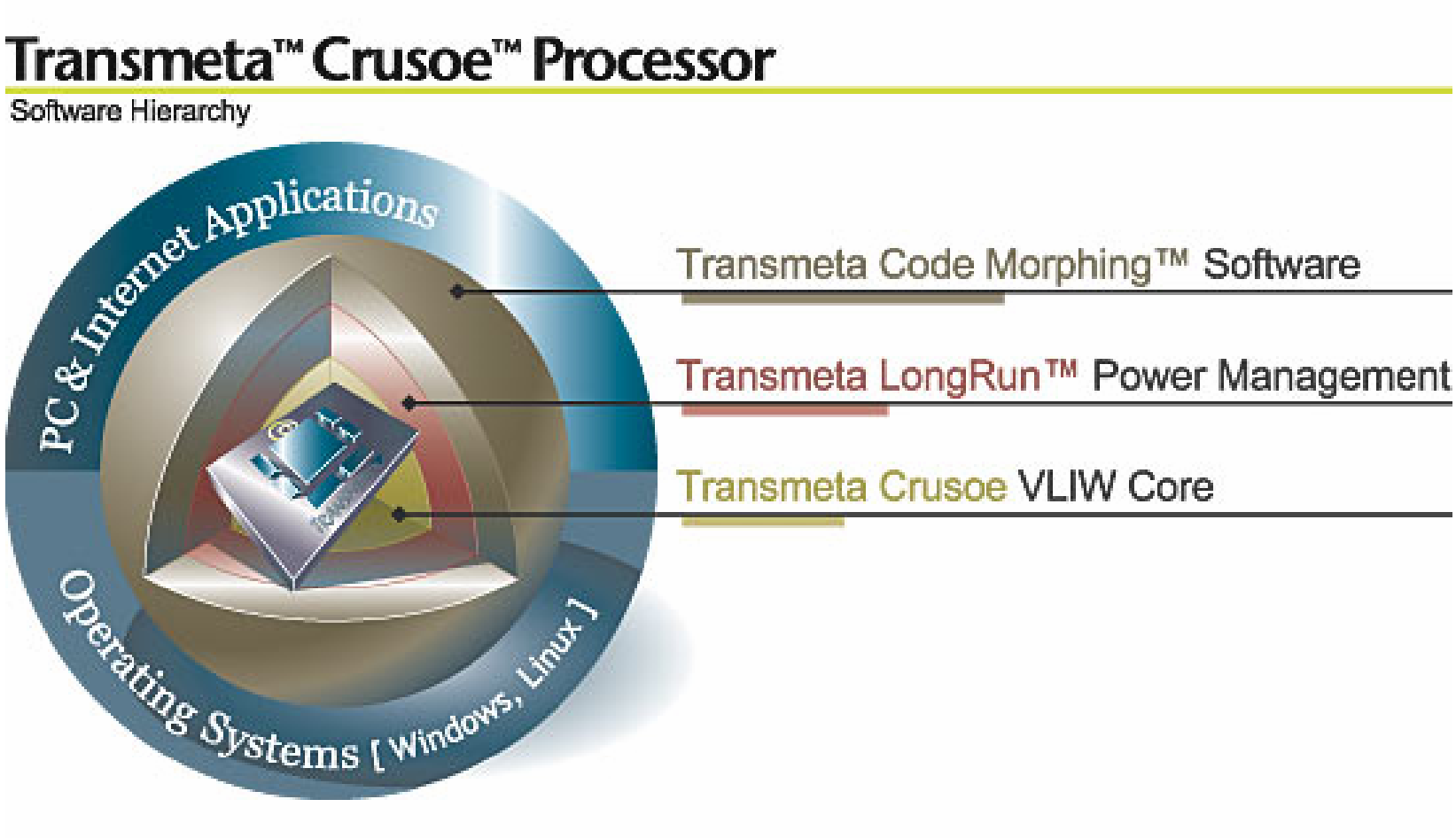
Very Long Instruction Word
(Nagyon Hosszú Utasítás Szó)

- Transmeta Crusoe CPU
- 128-bit-es utasítás csomag = molekula
  - 4 db 32-bit-es atom (atom = utasítás)
  - 4 utasítás párhuzamos elvégzése
- 64 általános használatú regiszter
- Kód átalakító réteg (code morphing layer/code morphing software)
  - A más CPU-k assembly utasításait tartalmazó kódot molekulákra fordítja
  - x86 (Intel Pentium) assembly utasítás-sorozatok futási időben történő „átfordítása” VLIW utasítás molekulákká
  - a végrehajtott utasítások NEM a Crusoe CPU utasításai

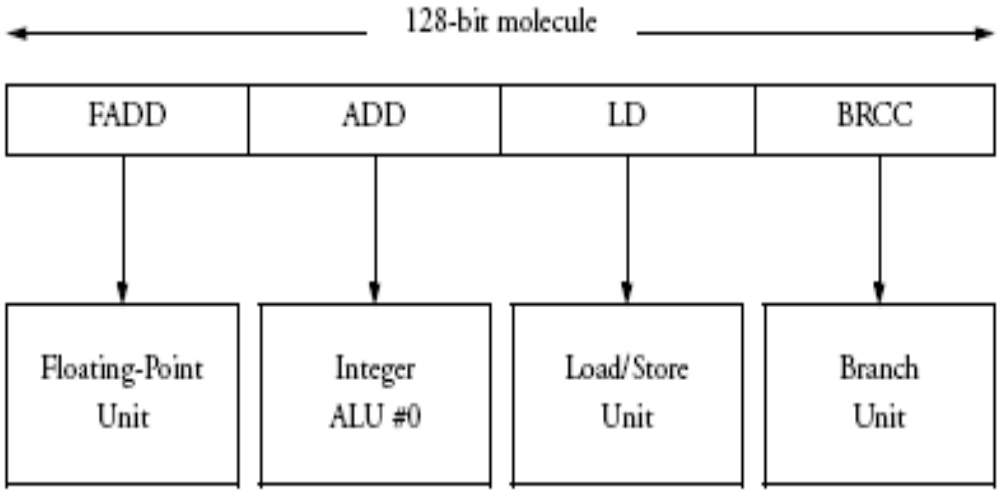
Transmeta Crusoe processzor felépítése

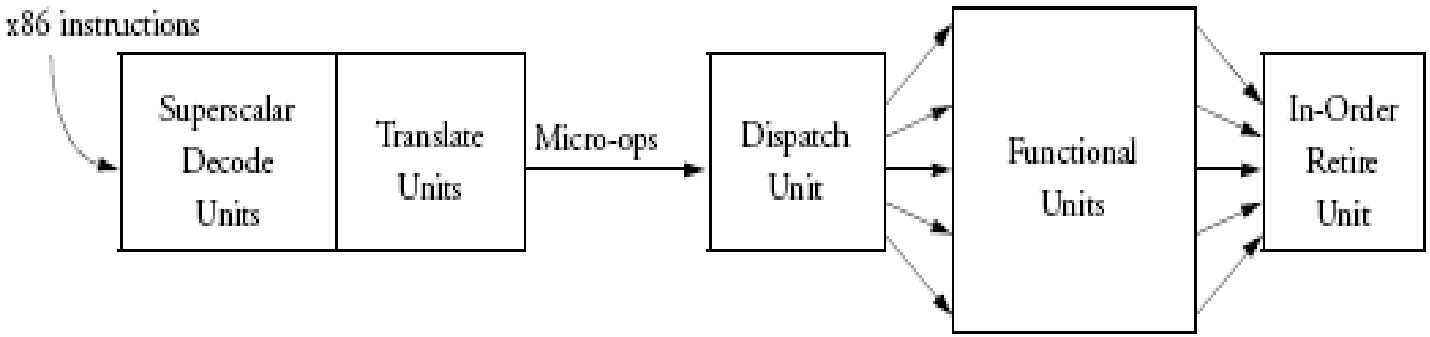


VLIW architektúra



VLIW utasítás végrehajtás





EPIC architektúra

- Intel Itanium CPU (IA-64)
- Új utasítás készlet az EPIC architektúrájú CPU-khoz
- 128-bit utasítás csomag
  - 3 db 41-bit-es utasítás (=123 bit)
  - maradék 5 bit határozza meg az utasítások típusát a csomagban
- 128 db 64-bit-es általános használatú regiszter
- 128 db 82-bit-es lebegőpontos regiszter
- Alkalmassá tették a hagyományos Intel X86-os utasítások végrehajtására is
- Ajánlások programozók és a fordítóprogramok számára közeli utasítások függőségének kiküszöbölésére
  - biztosítja az utasítások párhuzamos végrehajthatóságát

VLIW vs. EPIC

- VLIW
  - bármilyen utasítás sorozat végrehajtása
  - „optimalizálás” (függőségek vizsgálata) a processzor (ill. a processzor környezet) feladata
- EPIC
  - csak „optimalizált” utasítás sorozat végrehajtása
  - „optimalizálás” (függőségek vizsgálata) a programozó (ill. a program fejlesztő környezet) feladata

CPU fejlett szolgáltatásai

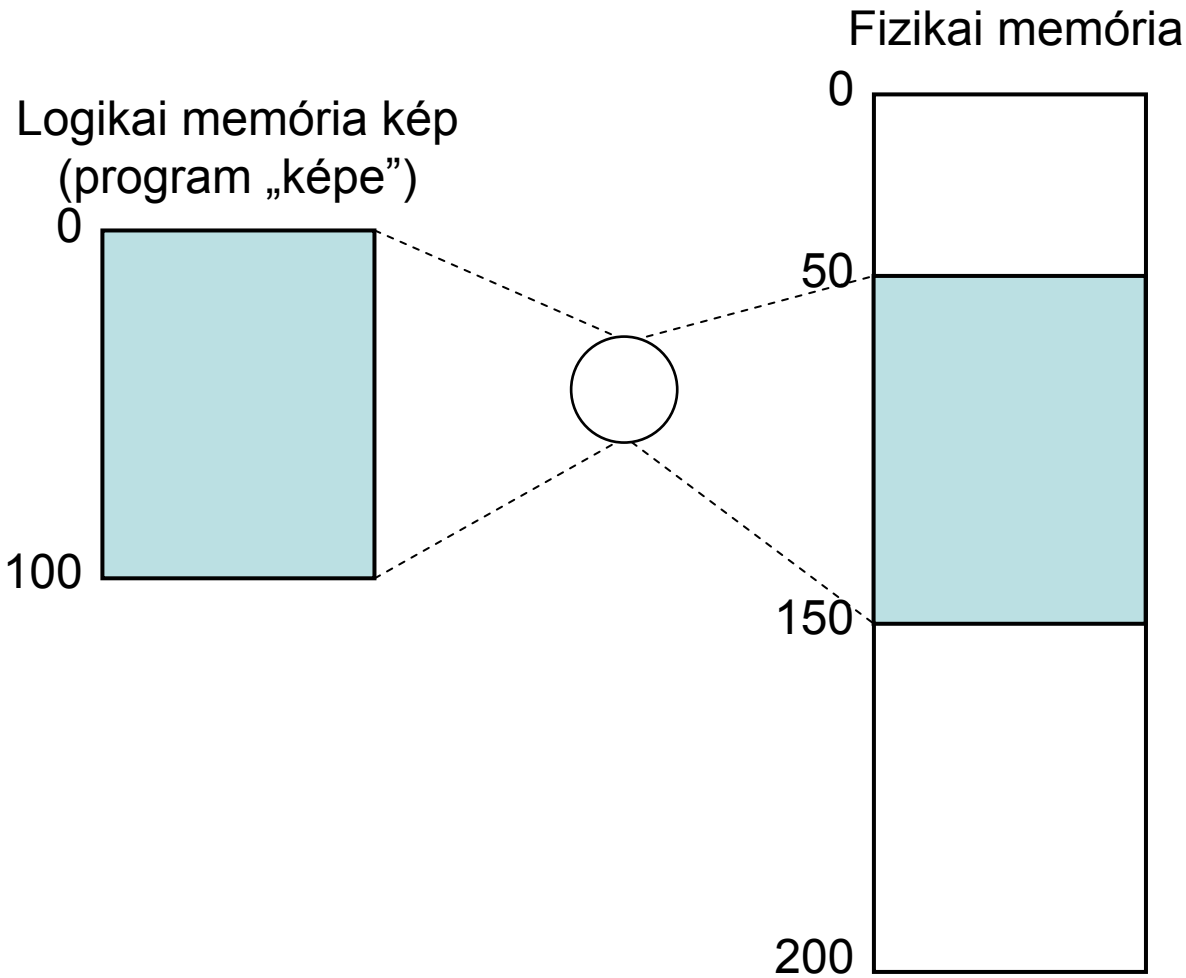
Memóriacím-transzformáció: Paging (lapozás)

- Futási időben történő memóriacím-transzformáció (címfordítás)
- Operációs rendszer feladata
- Hardver támogatja a megvalósítását
- A futó programtól független

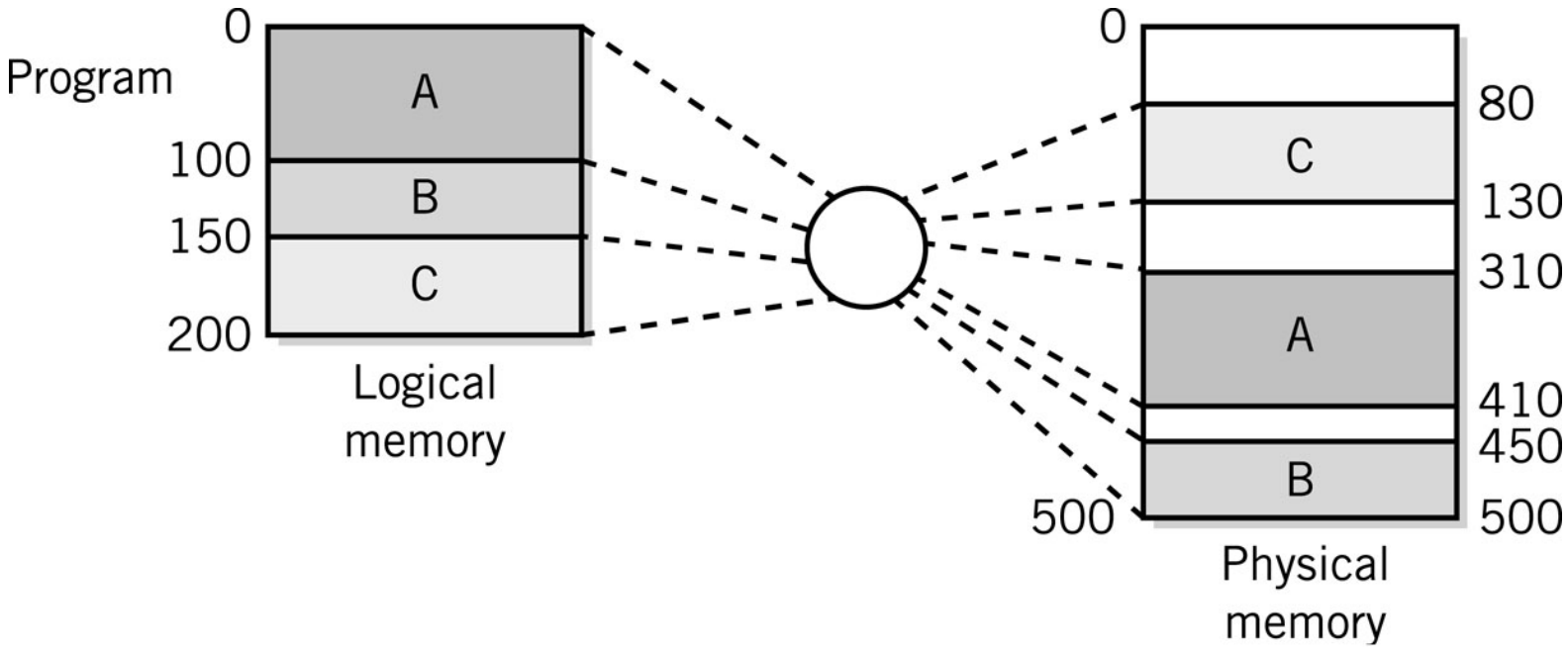
Logikai és fizikai címek összehasonlítása

- A logikai címek az adatok, utasítások, ill. ugróutasítás-címek **relatív** (egymáshoz viszonyított) helyét adják meg, amely **független** azok valós elhelyezkedésétől (fizikai címtől)
- A logikai címeket **átfordítjuk (konvertáljuk)** fizikai címekké
- A fizikai címeknek nem kell feltétlenül sorrendhelyesnek lenni

Címfordítás logikája

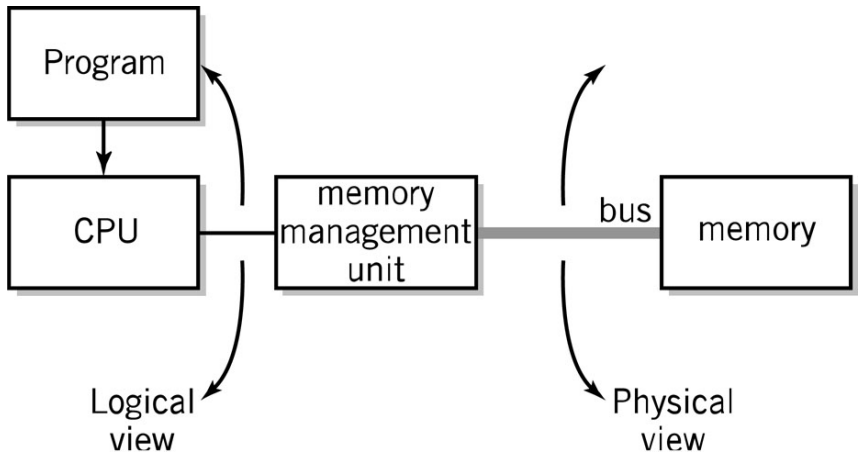


Logikai és fizikai címek összerendelése (mapping)

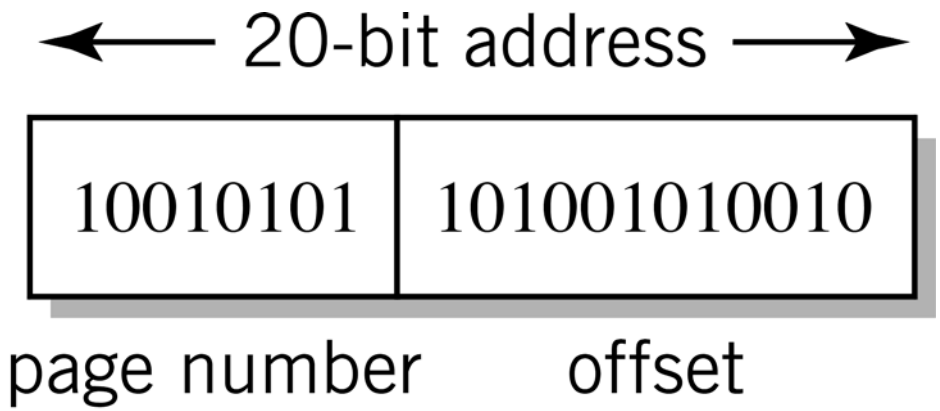


Logikai és fizika memóriakép

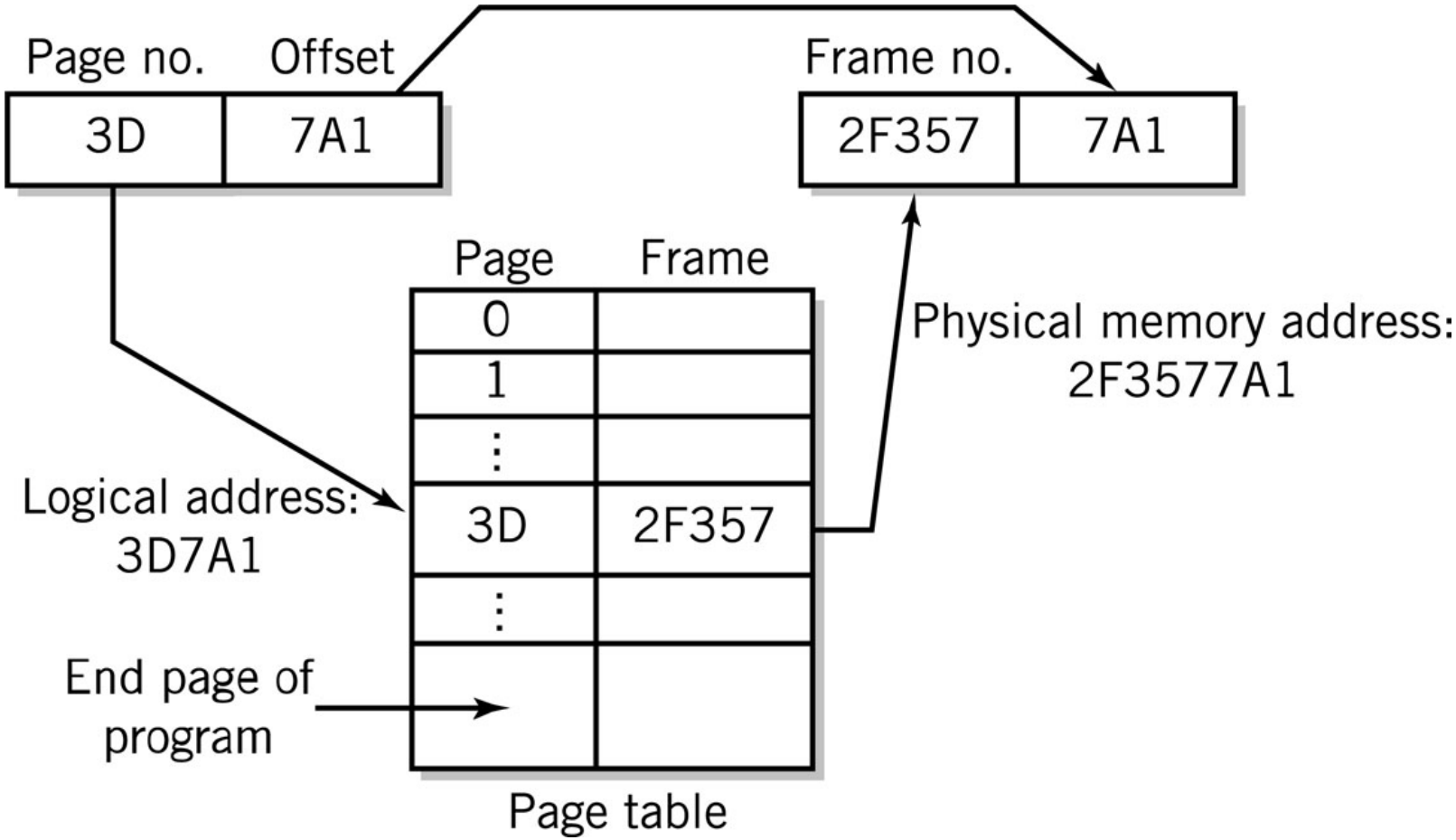
- Memória címfordítás HW támogatása
  - MMU (Memory Management Unit)
  - általában része a CPU-nak
- Logikai címek
  - programban tárolt címből a CPU az aktuális címzési logika szerint állítja elő
  - MMU (+operációs rendszer) átfordítja fizikai címmé
- Fizikai címek
  - CPU külső buszán jelennek meg



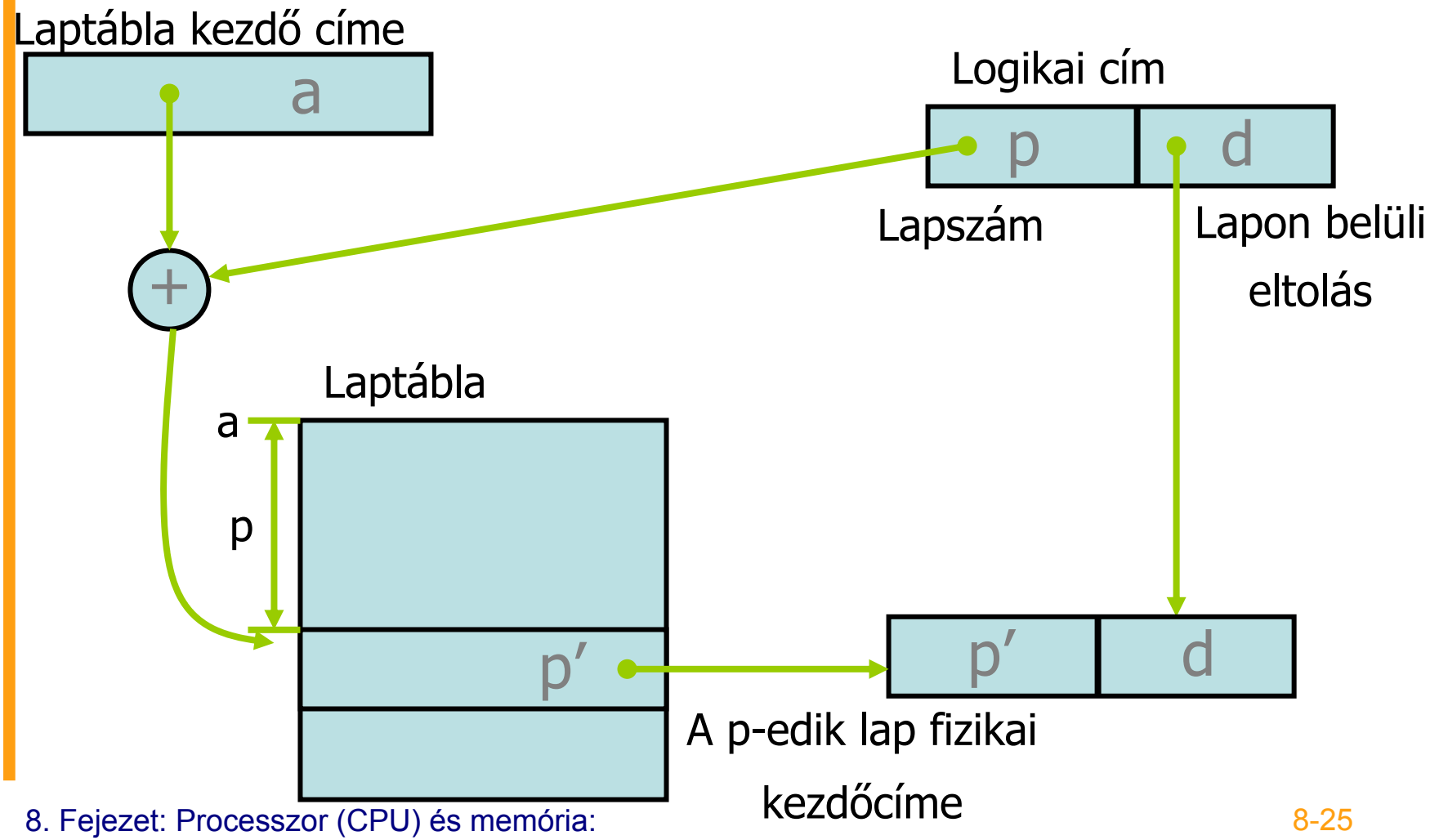
Lapcím felépítése



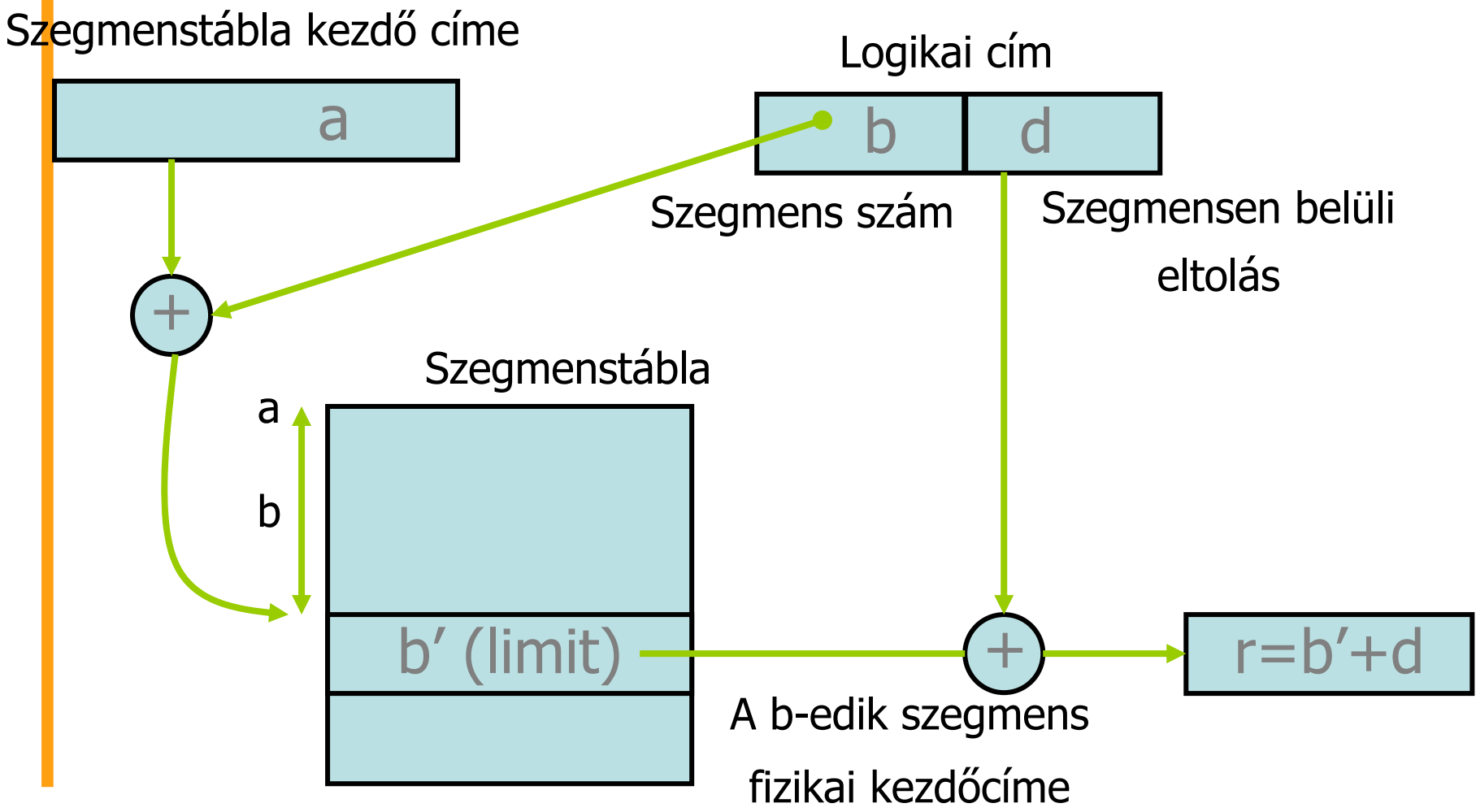
Címfordítás menete



Címfordítás lapozás esetén



Címfordítás szegmentálás esetén



Címtranszformáció adta további lehetőségek

- Memória elérés:
  - címtranszformáció
- MMU (+ OR):
  - címfordító táblázat
- Virtuális memóriakezelés (VM):
  - memória kiváltására használjunk háttértárat
  - címtranszformációt egészítsük ki a VM támogató lépésekkel
  - Bonyolult működés, számos többlet feladat: operációs rendszer!

Fejlett módszerek a memória-elérés gyorsítására

Wide Memory Path Access
(Széles sávon történő memória elérés)

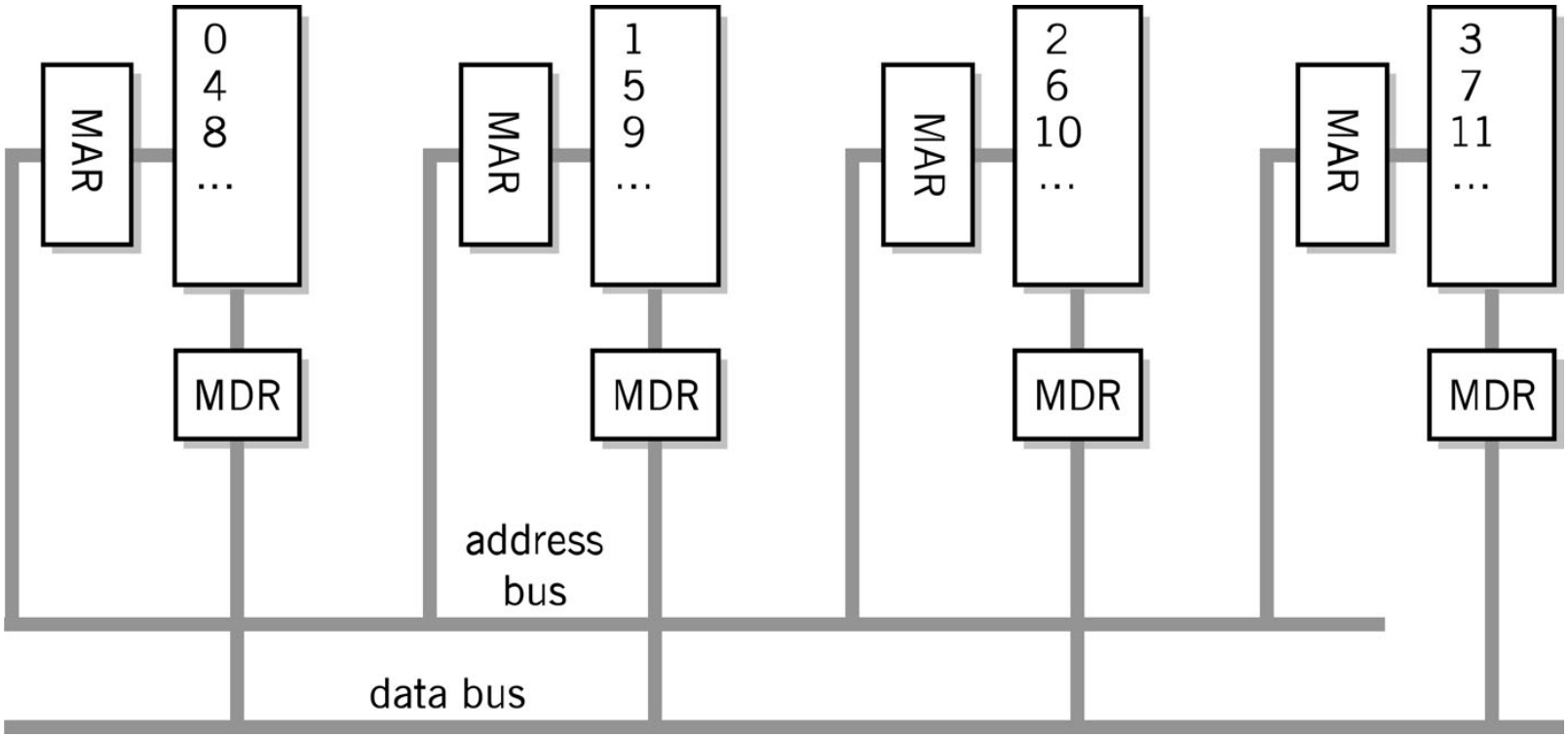
Memory Interleaving
(Memória felosztás)

Cache memória

A memória elérés gyorsítása

- A memória lassú a CPU sebességéhez képest!
  - 2 Ghz-es CPU = 1 ciklus a másodperc ½ milliárdod része alatt ($0{,}5 \times 10^{-9}$ másodperc)
  - 70ns DRAM = 1 elérés a másodperc 70 milliomod része alatt (70×10^{-9} másodperc)
- Eljárások a memória elérések gyorsítására
  - **Wide Memory Path Access (~Széles sávú memória elérés)**
    - Több byte-ot olvas ki a memóriából egy olvasási ciklusban
    - Széles (2,4,8) byte-os adat busz és MDR
  - **Memory Interleaving (~Memória felosztás)**
    - Memóriát részekre osztjuk, mindegyiknek lesz saját cím- és adatregisztere (MAR és MDR)
  - **Cache memória**

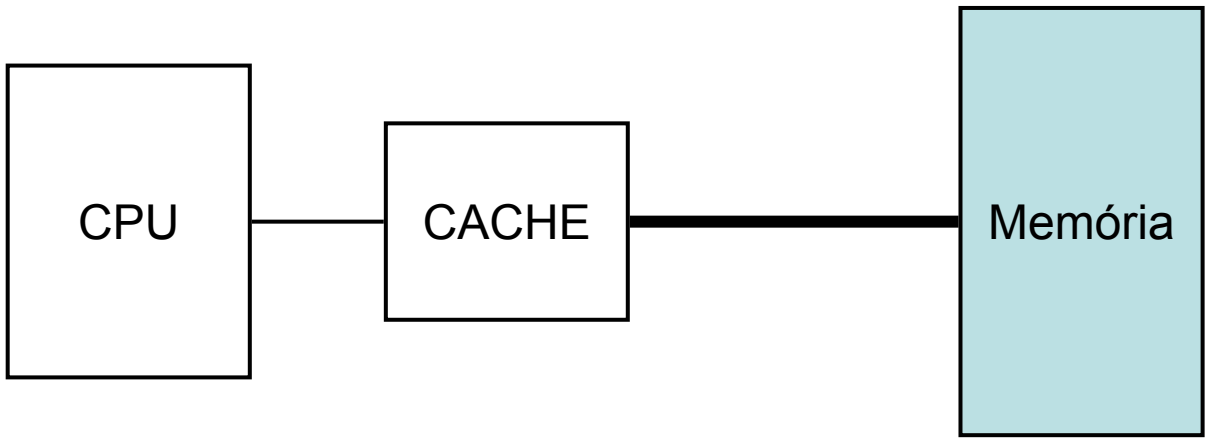
Memory Interleaving (Memória felosztás)



Cache

Cache

- Gyorsítótár, gyors elérésű memória (SRAM) a memória (ill. más lassú tárolók) elérésének meggyorsítására



Miért éri meg a cache használata?

- A leggyorsabb merevlemezek elérési ideje is körülbelül 10 milliszekundum ($10x10^{-3}$ másodperc) körül van
- 2 GHz-es CPU-nak 10 milliszekundumot kellene várnia egy merevlemezről beolvasott adat elérésekor ($10x10^{-3}/0.5x10^{-9}=2x10^{-7}$)
- **20 millió órajelciklust veszítünk el!**

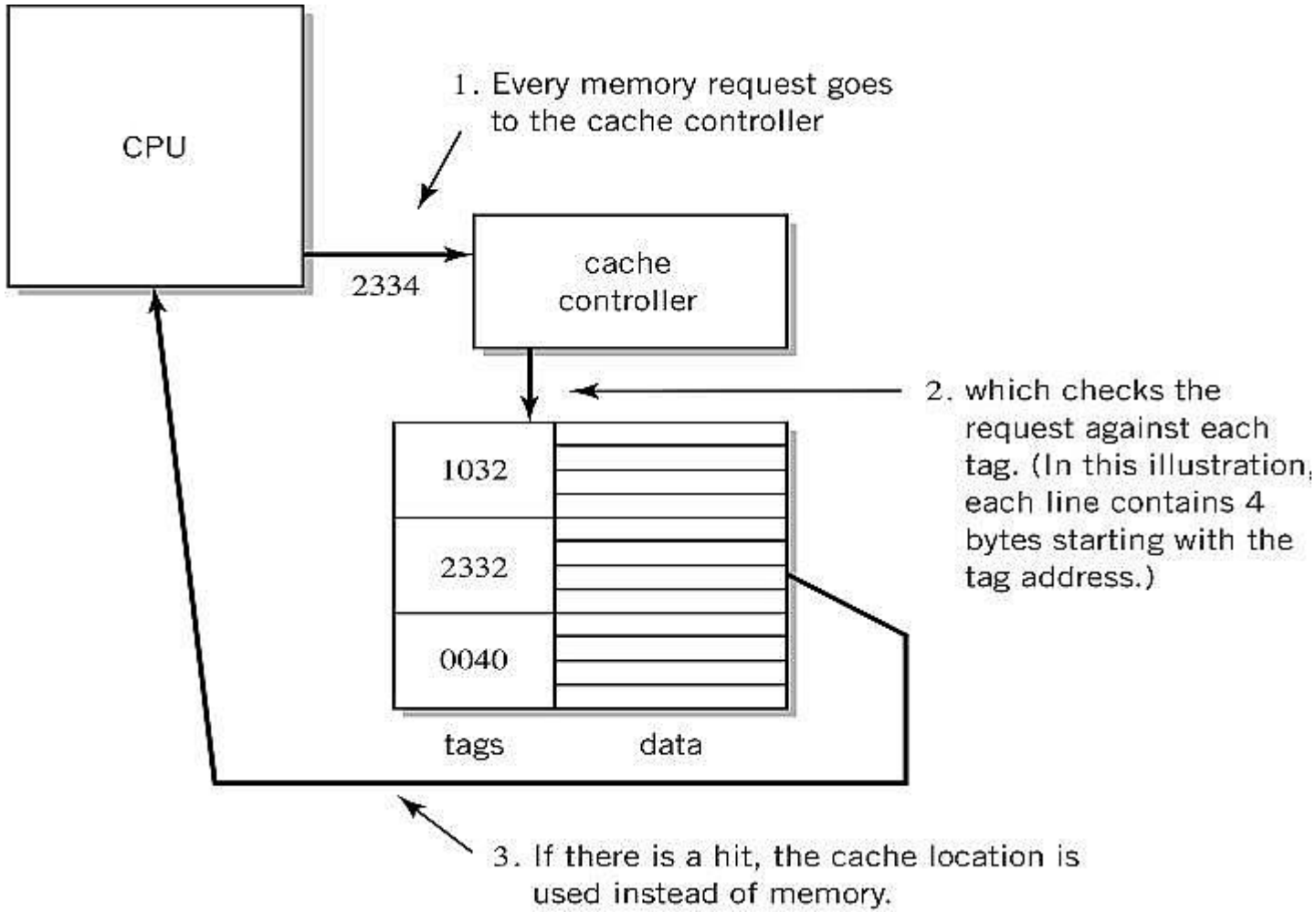
Cache memória felépítése

- Blokkokba szervezett felépítés
- Blokk mérete: 8 vagy 16 byte
  - a memória (tároló) egy részének másolata
  - Pl. 64 KB cache → 8192 db 8 byte-os blokk
- Minden blokknak van egy címkéje:
  - egy memória (tároló) cím!
- Cache controller (vezérlő)
  - Hardver elem, ami képes a címkék kezelésére
- Cache line (vonal)
  - a memória (tároló) és cache memória közötti adatcserét lebonyolító elem

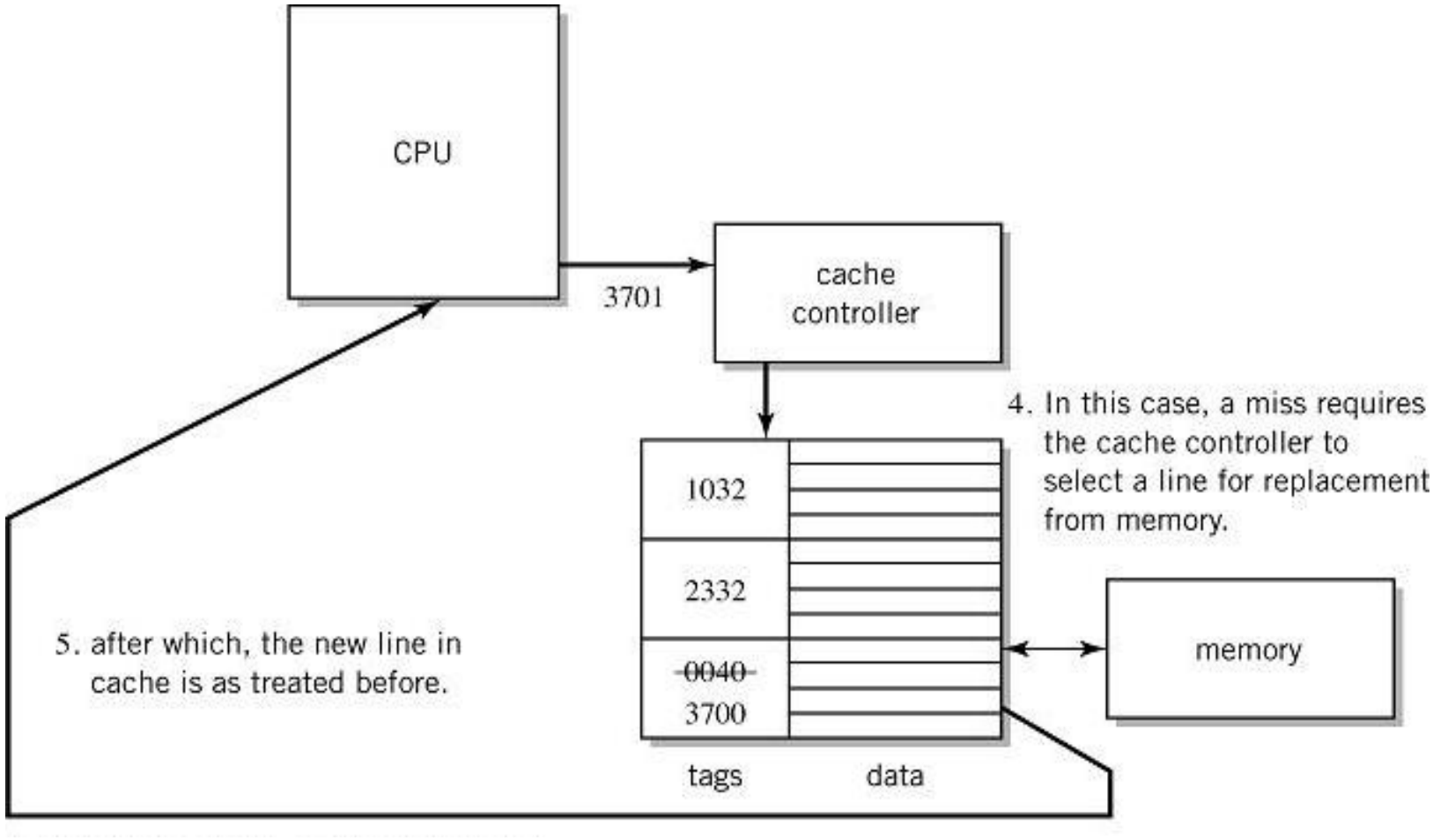
Cache működése

- Memória olvasása
  - Memória elérésekor a cache controller ellenőrzi, hogy a kért tartalom a cache-ben van-e
- Találati arány (hit ratio)
  - a cache találatok aránya az összes memória kérésből
- Memória írása
  - Feladat a cache és a memória összehangolása
- Alternatív megoldások
  - ***Write through (~Átírás)***
    - azonnali memória frissítés
    - lassabb, de biztonságos
  - ***Write back (~Visszaírás)***
    - memória frissítés csak felülíráskor
    - gyorsabb, de kockázatos

Cache használata lépésről-lépésre



Cache használata lépésről-lépésre

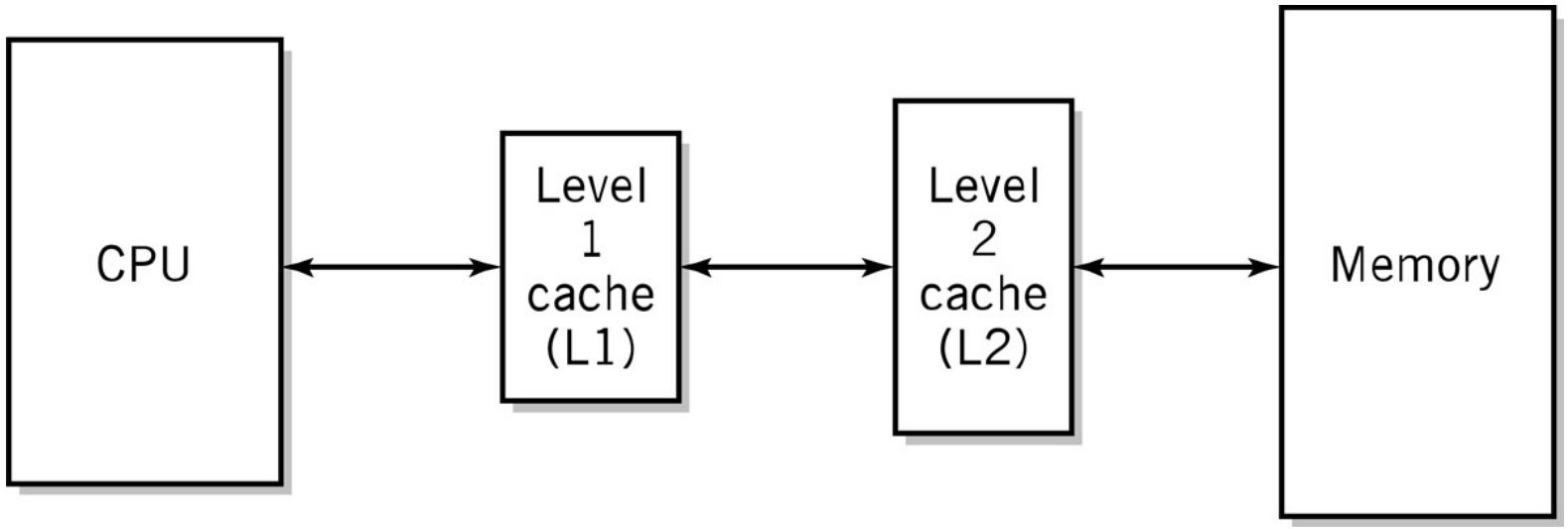


Teljesítménybeli előnyök

- Általános a 90% körüli találati arány
- 50%-ot is meghaladó műveleti sebesség növekedés
- A programok végrehajtásának „lokális természete” miatt működik hatékonyan a cache
  - ***Rövid időintervallumokat nézve a CPU által végrehajtott memória hivatkozások általában a memória kis régiójára vonatkoznak!!!***
  - Jól megírt program jellemzően rövid ciklusokat, eljárásokat vagy funkciókat tartalmaz
  - A feldolgozott adatok gyakran tömbökben vannak tárolva egymás után
  - Egy adott funkció megvalósításához szükséges változókat együtt tároljuk

Kétszintű cache-ek

- Miért kell a két cache méretének különbözőnek lenni?



Cache és virtuális memória összehasonlítása

- Cache felgyorsítja a memória elérést
- A virtuális memória használata megnöveli a programok által „tapasztalt” tár nagyságát
  - Alkalmazásának lehetősége az adott gép konfigurációjától és a memória kapacitásától függ
  - A memóriát alacsony bit-enkénti költséggel tudja megnövelni

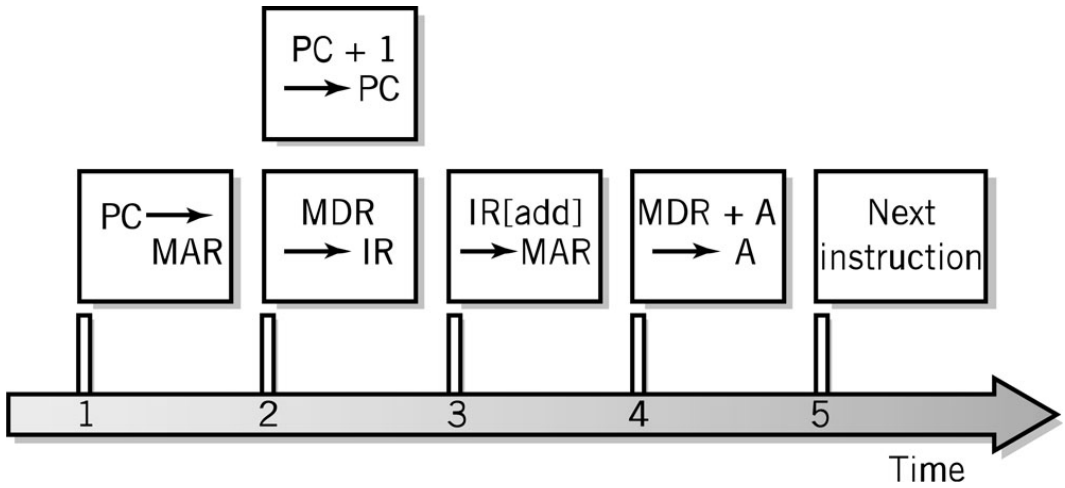
CPU műveletek végrehajtásának gyorsítása

Utasítás végrehajtás modern CPU architektúrákban

- Utasítások ütemezett végrehajtása
- Fetch/Execute egységek különválasztása
- Pipeline (futószalag jellegű) feldolgozás
- Feldolgozás skalár processzorokban
- Feldolgozás superskalár processzorokban

Időzítéssel kapcsolatos kérdések

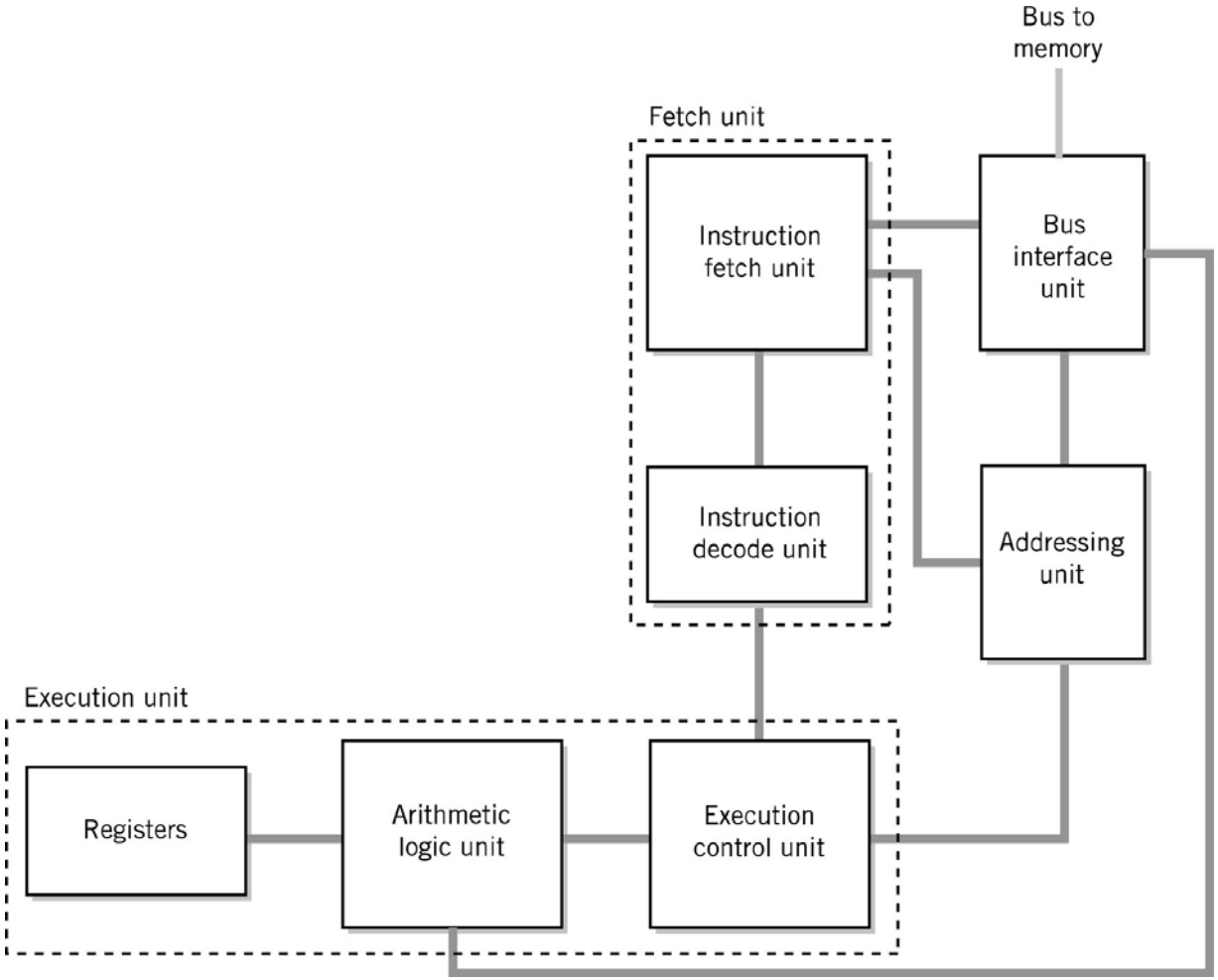
- Számítógép órajelét használjuk az utasítások lépéseinek ütemezésére
  - MHz – egymillió (10^6) lépés másodpercenként
  - GHz – egymilliárd (10^9) lépés másodpercenként
- Az utasítások végrehajtása általában több lépésből áll



Fetch-Execute egységek különválasztása

- Fetch fázist végrehajtó egység
  - Utasítást memóriából beolvasó egység
  - Utasítás dekódoló egység
    - Műveleti kód (opcode) meghatározása (utasítás része)
    - Utasítás típusának és az operandusok azonosítása
  - Ha több utasítás egymással párhuzamosan van fetch-elve, a dekódolásig, ill. a végrehajtásig egy pufferben tároljuk
  - IP – Instruction Pointer (utasítás mutató) regiszter
- Execute fázist végrehajtó egység
  - Az utasítás dekódoló egységtől kap utasításokat
  - A megfelelő végrehajtó egységek hajtják végre az utasításokban kódolt műveleteket

CPU lehetséges belső felépítése



Pipeline utasítás feldolgozás

- Az utasításokat közel azonos hosszú lépésekre bontjuk (pl. fetch és execute fázis)
- A lépéseket egymástól függetlenül működő egységek hajtják végre a CPU-n belül
- „Futószalag” technika, hogy átfedés jöjjön létre a lépések (pl. fetch-execute fázisok) végrehajtása között az egymást követő utasítások feldolgozásakor

Pipeline feldolgozás jellemzői

- Az egyes utasítások végrehajtási ideje nem változik
- Az utasítások párhuzamos végrehajtása miatt gyorsul a feldolgozás
- **Egy feldolgozó egység: egy adott pillanatban (órajelciklusban) csak egy utasítást lehet befejezni**
- Feldolgozás skalár processzorokban
  - A végrehajtott utasítások átlagos száma közel egyenlő az eltelt órajel-ciklusok számával

Pipeline feldolgozás - problémák

- Az utasítások különböző számú lépésből állhatnak
  - Nem tudjuk kihasználni a pipeline lehetőségeit
- Ha az utasításokat szekvenciálisan kell végrehajtani nem gyorsít
  - Adat függések esetén ez a helyzet
- Elágazás (branch) utasítások esetén a pipeline nem jó utasításokat hajthat végre

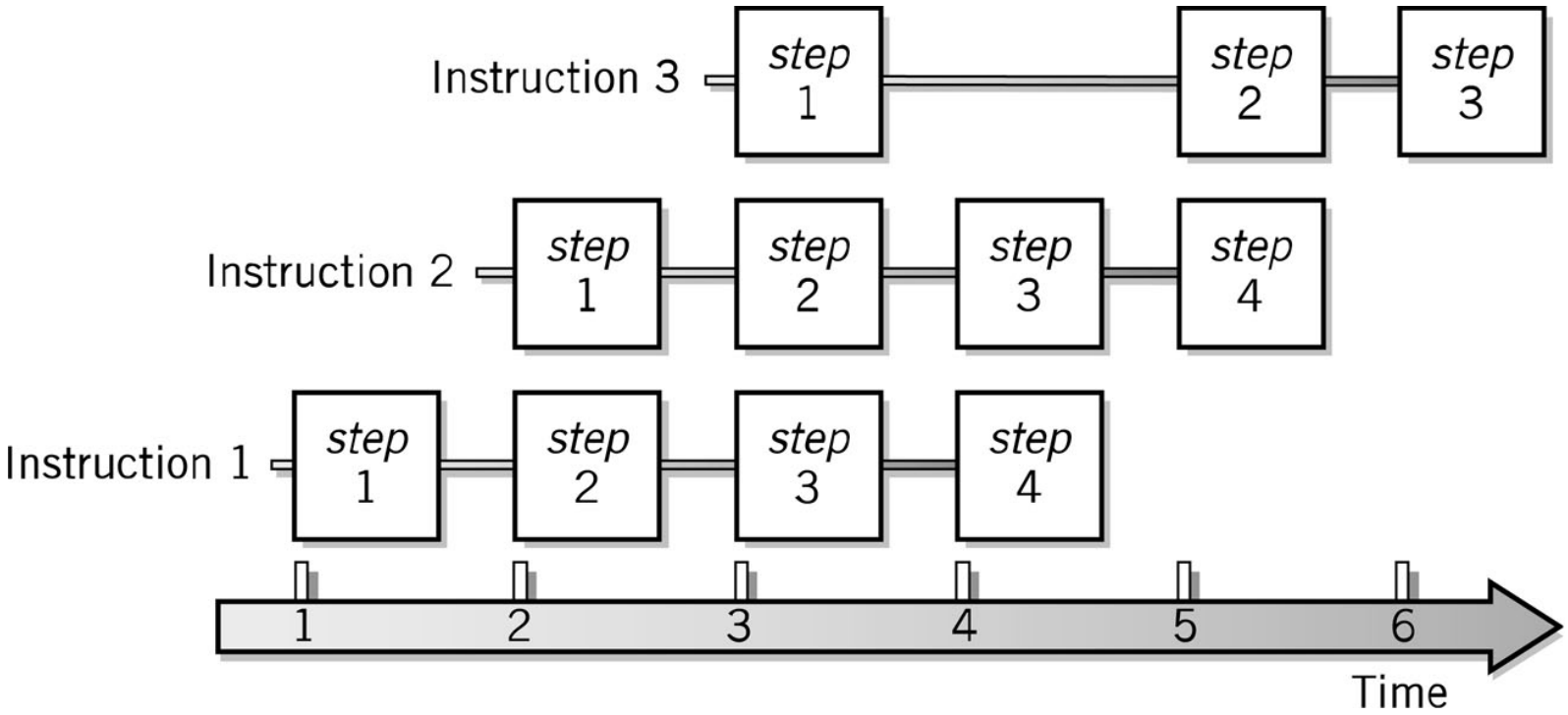
Elágazás utasítások kezelése

- Mindkét lehetőségre külön a pipeline sort kezel a CPU
- Jóslás alapú megoldások
- Megkötések az utasítások sorrendjére, úgy, hogy az elágazás utasítás utáni utasítás ne függjön a elágazás utasítás feltételétől

Adat függések kiküszöbölése

- Utasítások átrendezése
  - cél: egymás után következő utasítások eredményei ne függjenek egymástól
  - gyakran alkalmazott megoldás szuperskalár processzorokban a párhuzamos pipeline sorok feltöltésére

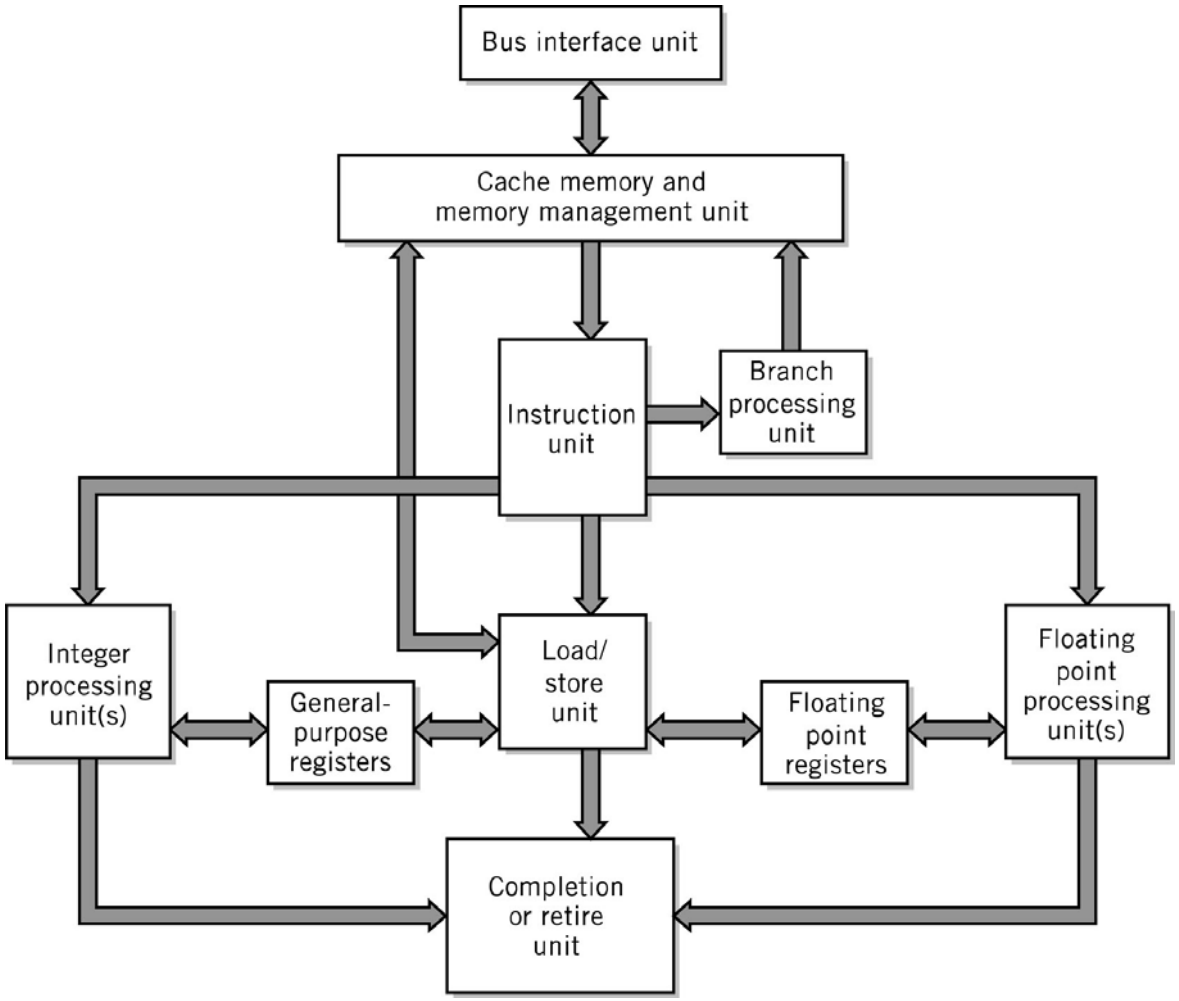
Pipeline példa



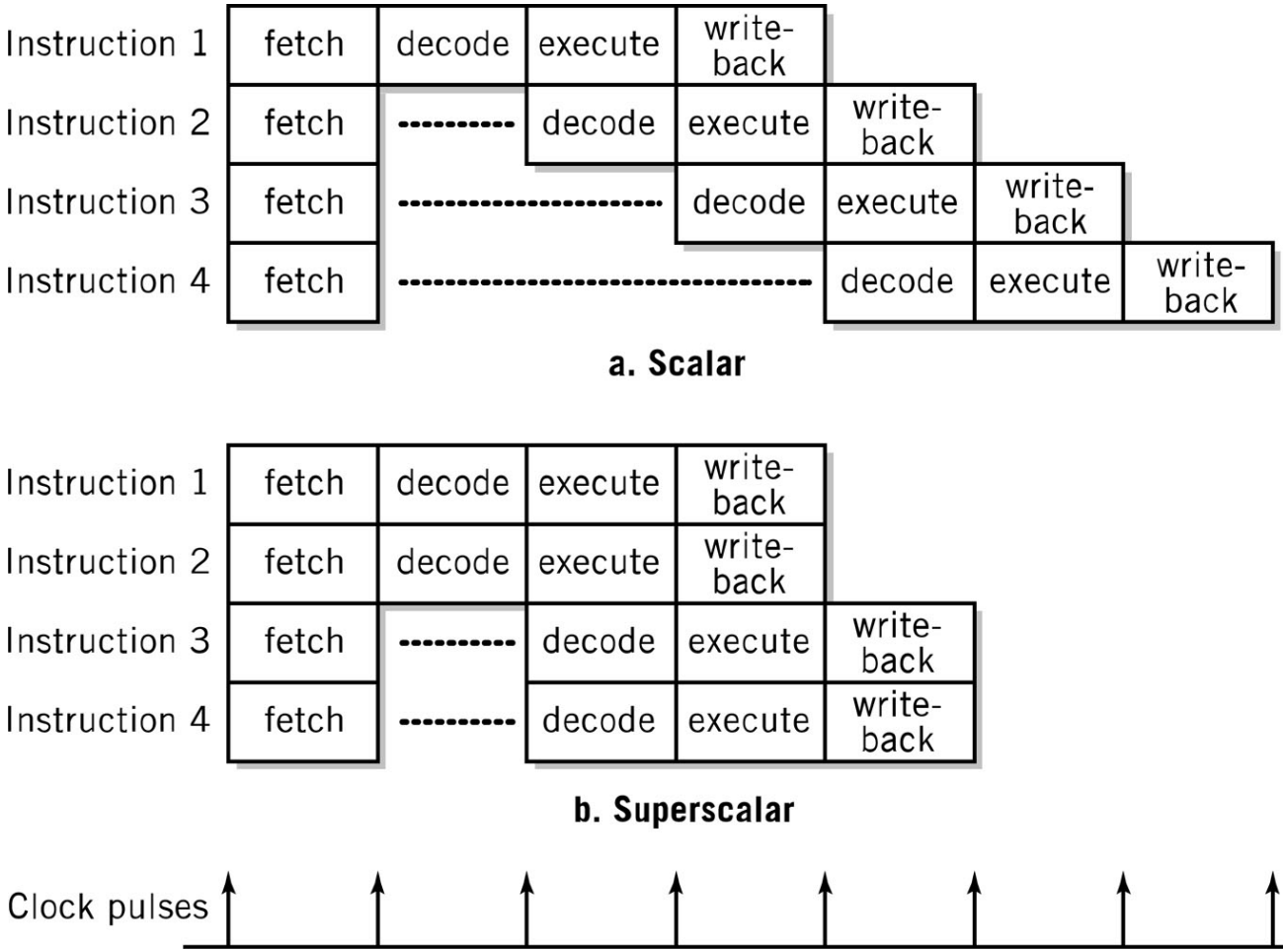
Feldolgozás szuperskalár processzorokban

- Átlagosan több, mint egy utasítást hajt végre egy órajel alatt
- A lehető legjobban elkülöníti a fetch és az execute utasításfázisokat
- Átmeneti tárolók (puffer regiszterek) a fetch és a dekódoló a fázisok
- Több piplene sor kezelése különböző utasításcsoportokhoz
  - Utasítás-csoportokbeli utasítások hossza azonos!!
  - Párhuzamos végrehajtó egységek
    - Load/Store utasítás végrehajtó egység
    - Branch (elágazás) kezelő egység
    - Integer aritmetikai egység
    - Lebegőpontos aritmetikai egység

Szuperskalár CPU blokkdiagramja



Skalár és Szuperskalár feldolgozás



Megoldások szuperskalár processzorokban

- Függőségek (hazárdok) kezelése fontos
- Soron kívüli (out-of-order) feldolgozás
  - „Elvégezhető” utasítások keresése a következő (akár 10-20) utasítás között
  - Jellemző adatfüggések „kikerülésére”
- Elágazás utasítások feldolgozása
  - Párhuzamos több pipeline sor kezelése vagy az elágazások jóslása
  - „Branch History Table” használata a jóslás javítására
- Regiszterek párhuzamos használatából adódó ütközések
  - Logikai regiszterek használata
  - Változtatható szerepű regiszterek

CPU implementációja

CPU műveletek hardveres megvalósítása

- Hardver implementáció
  - a műveleteket logikai kapuk hajtják végre
  - adatmozgató, logikai, bitforgató műveletek
- Előnyei
  - Nagy sebesség
  - RISC processzorok jellemzően ilyenek
    - utasítások egyszerűek, gyorsak

Mikroprogramozott CPU implementáció

- A CPU un. mikro-programokat hajt végre
  - A mikro-programok rövid programok, amelyeket a CPU-ba integrált ROM tárol
  - A használható utasítások egyszerűek: pl. regiszterek közötti adatcsere, logikai egység vezérlése.
  - A CPU utasításokat ilyen mikro-program utasításokból lehet összeállítani
- Előnyei
  - Rugalmasabban kezelhető utasításkészlet
  - Könnyebb az összetett utasítások végrehajtása
  - Tudjuk emulálni más CPU-k utasításkészletét
- Hátránya
  - Az utasítások végrehajtása általában több órajelet igényel

Copyright 2003 John Wiley & Sons