ÓBUDAI EGYETEM

SZÁMÍTÓGÉP ARCHITEKTÚRÁK ALAPJAI

15-16. előadás

Durczy Levente

2025/2026/1

ÓBUDAI EGYETEM

Második generációs szuperskalárok:

Az első generációs szuperskalár CPU-k kibocsátási szűk keresztmetszete miatt az átbocsátóképesség növeléséhez át kellett tervezni az architektúrát. ➡ 2. gen. szuperskalár CPU-k!

Kibocsátási rátájuk körülbelül 4 utasítás/ciklus RISC és 3 utasítás/ciklus CISC architektúrák esetén, tehát az első generációhoz képest jelentős teljesítménynövekedés történt.

Ilyen processzorok voltak például:

- Intel Pentium Pro (1995)
- AMD K6
- PowerPC 604

Egy CPU-t akkor tekinthetünk második generációs szuperskalárnak, ha jellemző rá:

- Dinamikus utasítás ütemezés, valamint
- Regiszter átnevezés (e kettővel kiküszöbölték a közvetlen kibocsátási szűk keresztmetszetet!)
- Elágazások előrejelzése dinamikus előrejelzéssel ugrástörténet figyelembevételével (ez körülbelül 90-95%-os pontosságot biztosított)
- Kifinomult és kibővített gyorsítótár alrendszer
- Sorrenden kívüli kiküldés (OoO: Out of Order)
- RISC mag (x86 arch.)

Dinamikus utasítás ütemezés:

Más néven várakoztatás vagy pufferelt utasításkibocsátás és kiküldés.

Lényege, hogy az utasítások kibocsátása pufferelt, sorrendi módon történik, a kiküldés viszont sorrenden kívüli (Stréber modell)!

Ez jelentősen megnöveli a mikroarchitektúra elejének átbocsátóképességét, valamint kiküszöböli a kibocsátási szűk keresztmetszetet.

Kibocsátás: Dekódolóból a várakoztató állomásba

Kiküldés: Várakoztató állomásból a Végrehajtóegység (E) felé

Regiszter átnevezés (transzparens):

A dinamikus utasítás ütemezés ugyan növeli az átbocsátóképességet, viszont a függő utasításokat nem küszöböli ki, azok továbbra is lassítják a végrehajtást!

Megoldás: regiszter átnevezés, ➡ kiküszöböli az ál adatfüggőségeket.

A WAR és WAW függőségeket még a kibocsátás előtt megszünteti (még mielőtt a várakoztató állomásba kerülnek).

A gyakorlatban ennek eléréséhez a CPU minden regiszterhez allokál egy átnevezési (piszkozat) regisztert. Ilyenkor az átnevezési logika követi az aktuális regiszter allokációkat, átnevezik a forrás regisztereket is, így az architektúrális regisztereket elég csak visszaíráskor figyelembe venni!

Az allokációt a CPU az utasításokhoz, és nem az architektúrális regiszterekhez köti!

Ez a megoldás dinamikus elágazásbecslés számára is előnyös, mivel, ha kiderül, hogy a program mégsem azon az ágon folytatódik, amihez az adott utasítást végrehajtottuk, az eredmény még csak a piszkozat regiszterben van jelen, ahonnan könnyen törölhető!

Dinamikus elágazásbecslés:

Az elágazások történetét történet bitek formájában írja le. A dinamikus becslés lehet 1, 2 vagy 3 bites:

1 bites: a történet bit értéke lehet 0 vagy 1 attól függően, hogy előző elágazásnál ugrás vagy soros folytatás történt. A következő elágazásnál a történet bit alapján döntötte el a CPU, hogy ugrás vagy soros végrehajtás következik.

2 bites: a történet biteket 2 bittel ábrázoljuk.

- 11 – határozott elágazás: általában kezdeti állapot, ekkor mindenképpen ugrik
- 10 – gyenge elágazás
- 01 – gyenge soros folytatás
- 00 – határozott soros folytatás

Mivel legtöbbször az elágazásnál ugrás történik, ezért a történet bitek értéke kezdetben 11. Amennyiben a következő elágazásnál mégis hibás volt az ugrás, a történeti bitek értéke 10-ra változott, tehát ezt követi alkalommal megint meg fogja próbálni az ugrást. Ha harmadszorra is hibás volt az ugrás, átkerül 01 állapotba.

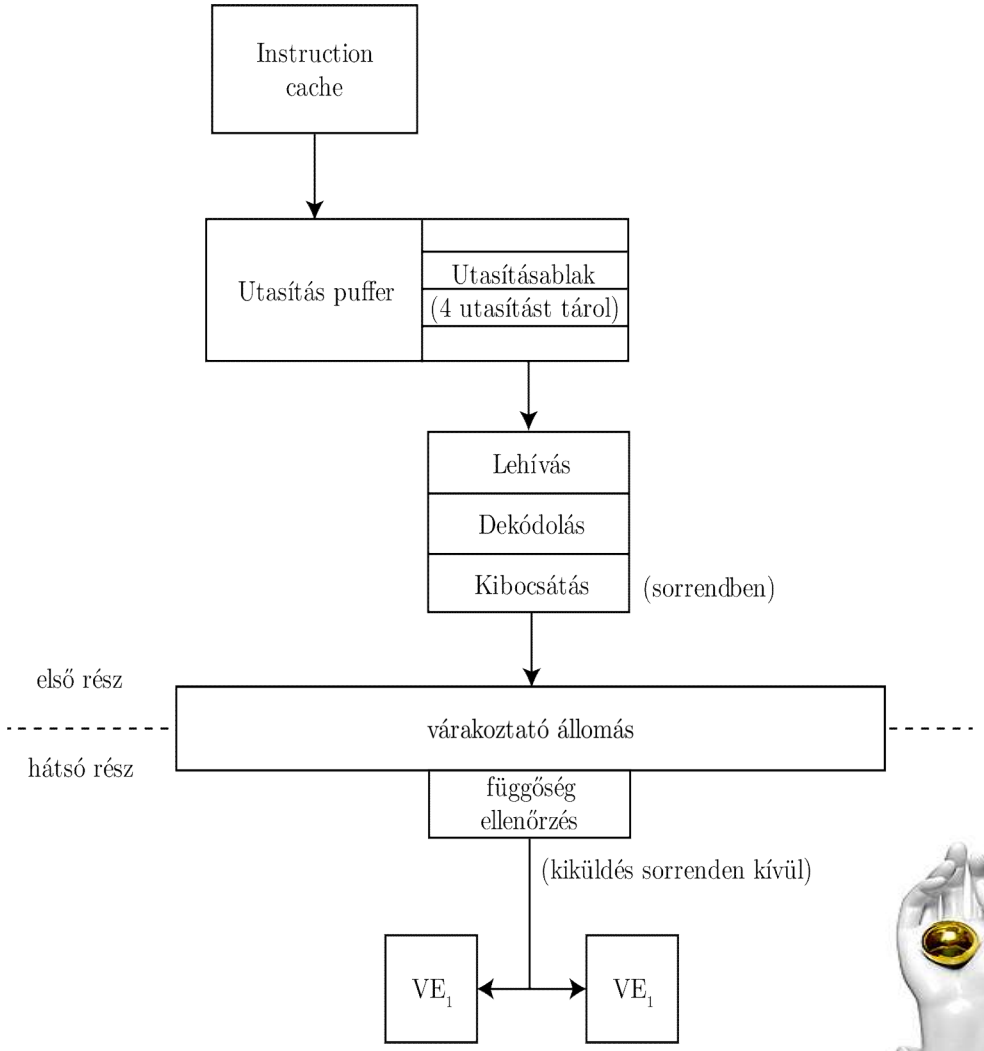
ÓBUDAI EGYETEM

Sorrenden kívüli kiküldés (Out of Order):

A második generációs szuperskalároknál kibocsátáskor nincs függőségvizsgálat, ezért a lehívás, a dekódolás és a kibocsátás nominális rátával működhet, ami ebben az időben általában 3-4 utasítás/ciklus volt. Az utasítások kibocsátáskor a kibocsátási pufferbe kerülnek, ez a várakoztató állomás. A kibocsátási puffer lehet közös is, de előfordulhat, hogy különböző típusú utasításoknak (FX, FP stb.) külön pufferek vannak fenntartva.

Emiatt a vezérlő óraciklusonként akár több tucat utasítás közül is választhat, hogy mit küldjön ki végrehajtásra. A döntés állapotbitek alapján történik, a vezérlő ezek segítségével dönti el az utasításokról, hogy azok függők vagy függetlenek. A CPU a független utasításokat küldi ki végrehajtásra.

Ebben az esetben a várakoztató állomás mentén beszélhetünk első és hátsó részről.



ÓBUDAI EGYETEM

Várakoztató állomás mérete különböző CPU-k esetén:

	Model	Family	Year	No. of ROB entries
Intel	Core 2		2006	96
	Penryn		2007	96
	Lynnfield		2009	128
	Sandy Bridge	2nd gen.	2011	168
	Haswell	4th gen.	2013	192
	Skylake	6th gen.	2015	224
	Sunny Cove/	10th gen. Ice Lake	2019	352
	Willow Cove	11th gen. Tiger Lake	2020	352
	Golden Cove	P core of 12th gen. Alder Lake	2021	512
AMD	Zen		2017	192
	Zen 2		2019	224
	Zen 3		2020	256
	Zen 4		2022?	320
Apple	Firestorm	Big core of A14/M1	2020	630

ÓBUDAI EGYETEM

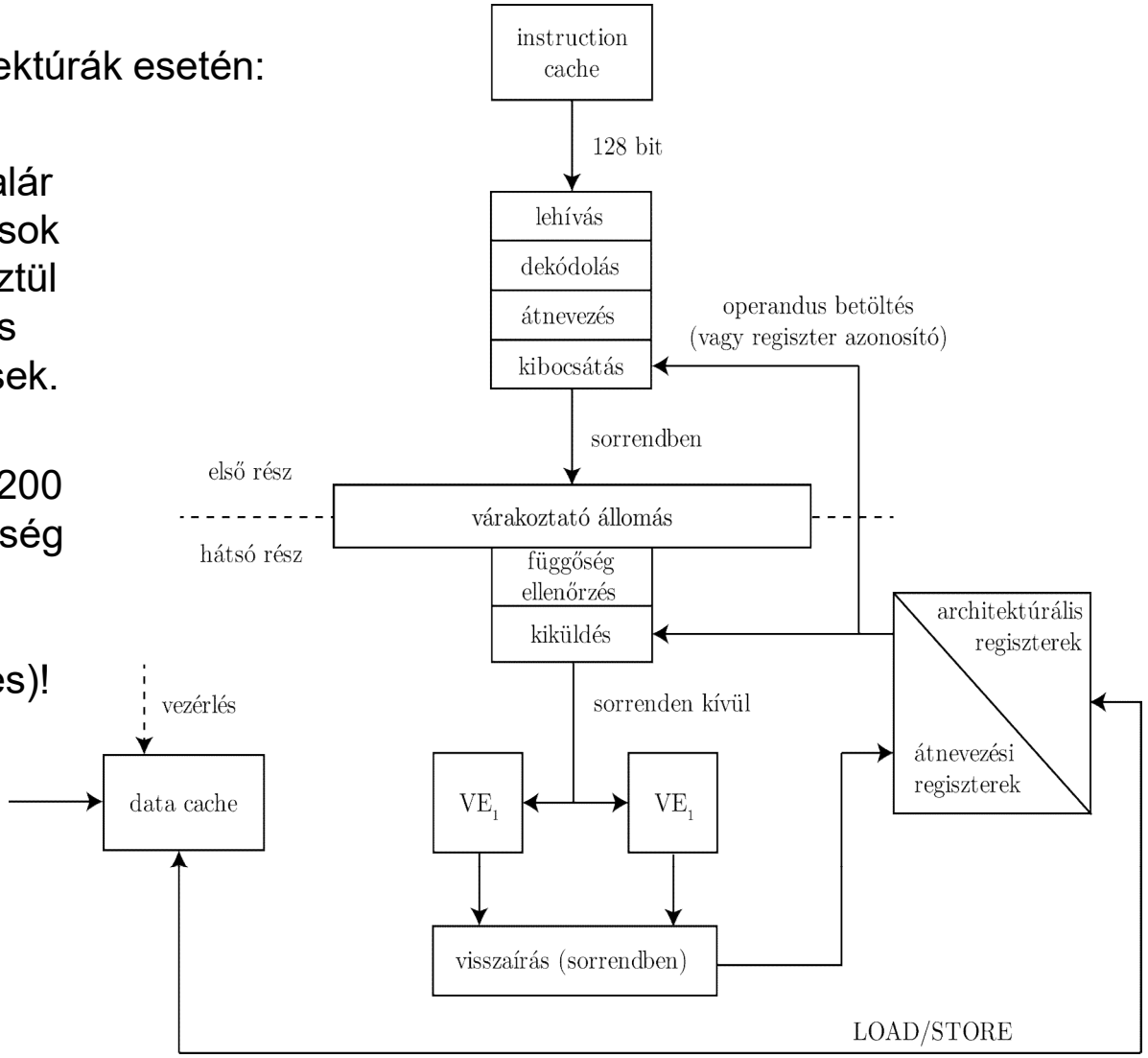
Végrehajtási modell RISC architektúrák esetén:

A második generációs szuperskalár RISC architektúráknál az utasítások lehívása 128 bites buszon keresztül történik. A lehívás és a dekódolás általában 4 utasítás/ciklus szélesek.

A kibocsátás egy várakoztató állomásban történik (ma akár ~ 200 bejegyzéses), itt történik a függőség ellenőrzés, majd az utasítások sorrenden kívül továbbítódnak a végrehajtóegységek felé (kiküldés)!

Az operandusok betöltése kétféleképpen történhet:

- kibocsátáskor vagy
- kiküldéskor.



ÓBUDAI EGYETEM

Kibocsátáshoz kötött operandusbetöltés:

Ha egy valós adatfüggőség miatt kibocsátáskor még nem áll rendelkezésre az operandus, akkor a helyére annak az átnevezési regiszternek az azonosítója kerül, ahol majd a függőség feloldásához szükséges eredmény elő fog állni. Az operandusok és a regiszter azonosítók megkülönböztetésére a CPU állapot biteket iktat a műveleti kódba az alábbi módon (RISC!):

MK	OP_1		OP_2	$Á_1$	OP_3	$Á_2$

Az állapot bitek értéke lehet 0, azaz az operandus nem áll készen, vagy 1, ha az operandus készen áll. A CPU csak az állapotbiteket vizsgálja!
0 értékű állapot bit esetén az utasítás a várakoztató állomásban van addig, amíg az operandus rendelkezésre nem áll.
Ekkor a kiküldés előtt az átnevezési regiszterből betöltődik az operandus, az állapot bit értéke pedig 1-re változik, az utasítás pedig végrehajtásra kerül.

Tölcsér elv:
A második generációs szuperskalároknál a már említett kibocsátási ráta körülbelül 3-4 utasítás/ciklus, a kiküldési ráta pedig általában 5-8 utasítás/ciklus. Ezt nevezzük tölcsérszerű elvnek, azaz a processzor hátsó része felé haladva nő az átbocsátási képesség (szélesedik). Ez a felépítés segít a szűk keresztmetszet kiküszöbölésében.

Kiküldéshez kötött operanduslehívás:

A későbbi generációknál rájöttek a mérnökök, hogy felesleges már a kibocsátáskor betölteni az operandusokat.

Ezért úgy módosították az architektúrát, hogy a kibocsátásnál ugyan megmaradt a gépi kód struktúrája, de az állapot bitek minden esetben 0-ra voltak állítva, az operandusok helyére pedig a regiszter azonosító került.

A függőség ellenőrzés és az operandusok betöltése így mindig a kiküldés előtt történik, ami egyszerűsíti és gyorsítja a működést!

Végrehajtás CISC architektúra esetén:

A CISC utasítások hossza változó (Intel Pentium Pro: 1-17 byte), futószalagos végrehajtás nem optimális!

Kezelés: **CISC utasítások konvertálása egységesen 128 bites, RISC-szerű utasításokká!**

Az átalakítás a dekódolás fázisban, a RISC-CISC visszaalakítás pedig a visszaírás során történt.
Ez a felépítés (kívül CISC architektúra, belül RISC mag) máig jellemző az Intel processzorokra!

Átalakítás után átlagosan 1 CISC utasítás ~ 1.2 – 1.5 RICS utasítás!

Vagyis egy CISC CPU 3 utasítás / ciklus rátája a RISC magban már 4 utasítás / ciklus!

Összefoglalás:

Függőségek kezelése:

- erőforrás függőség: több végrehajtó egység alkalmazása
- ál adatfüggőségek: regiszter átnevezés teljes mértékben megoldja
- valós adatfüggőség: még mindig blokkol. Részben kezelve a sorrenden kívüli kiküldéssel, várakoztató állomással és spekulatív elágazáskezeléssel
- vezérlés függőség: spekulatív elágazáskezeléssel részlegesen kezelhető

Végrehajtás:

Az utasítás akkor kerül kiküldésre a várakoztató állomásból, ha a bemenő operandusai rendelkezésre állnak. Ez megfelel az adatvezérlet (stréber) végrehajtási modellnek. (adatfolyam számítási modell)

Esettanulmány: Intel Pentium Pro (1995)

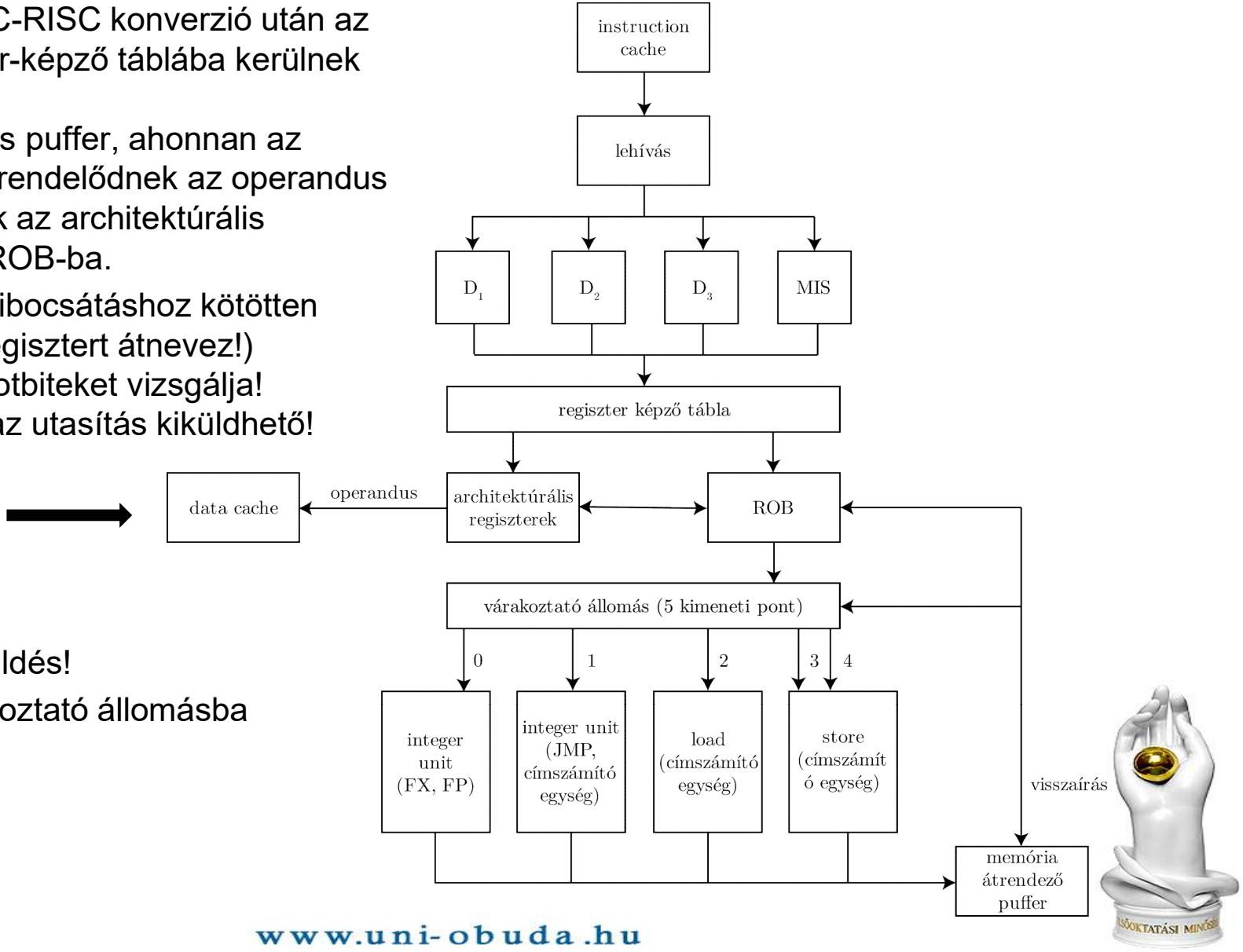
Tulajdonságai:

- Kezdeti frekvencia: 133 MHz
- FX futószalag 14 fokozatú (első generációs elődje 5 fokozatú volt ⟶ nagy ugrás)
  - Előnye: a rövidebb fokozatok miatt növelhető a frekvencia
  - Hátránya: több függőség
- RISC-szerű utasításokat használ. Ez is segített a rövidebb fokozatok kialakításában!
- 20 bejegyzéses várakoztató állomás (FX, FP stb. utasításokat vegyesen tárolta)
- A szigorúan soros konzisztenciát a ROB (reorder buffer) biztosítja
- Regiszterátnevezés
- A lehívás 128 bites. Itt fontos az utasítás határra illesztés (CISC miatt)!
- A dekódolás során a CISC utasításokat RISC-re konvertálja, óraciklusonként ~3 CISC utasítást hajt végre (így kompatibilis lesz, de gyors RISC utasításokat hajt végre!)
- A dekóder négy egységből áll
  - D1 – legfeljebb 4 RISC utasítássá átalakítható CISC utasítások dekódolását végzi
  - D2 és D3 – egy RISC utasítássá átalakítható CISC utasítások dekódolását végzi (egyszerű dekódolók – összeadás, kivonás)
  - MIS – 4-nél több RISC utasítássá átalakítható CISC utasítások dekódolását végzi
- A ROB több független utasítás esetén mindig a korábbit küldi ki végrehajtásra
- A kívülről CISC architektúrának köszönhetően megmaradt a kompatibilitás, ugyanakkor a RISC mag miatt jelentőség javult a teljesítmény!

ÓBUDAI EGYETEM

- Dekódolás és a CISC-RISC konverzió után az utasítások a regiszter-képző táblába kerülnek (sorrendben!).
  Ez egy 6 bejegyzéses puffer, ahonnan az utasításokhoz hozzárendelődnek az operandus címek és betöltődnek az architektúrális regiszterekbe és a ROB-ba.
- Operandus lehívás kibocsátáshoz kötötten
  (a CPU minden célregisztert átnevez!)
  a CPU csak az állapotbiteket vizsgálja!
  (ha mindkettő 1-es, az utasítás kiküldhető!

Végrehajtási modell:



- Sorrenden kívüli kiküldés!
- Eredmények a várakoztató állomásba kerülnek vissza

Végrehajtás:

A 0. porton 6 db végrehajtó egység van:

- Összetett FX VE
- Egész osztó (FX DIV)
- Eltoló/léptető egység (SHF)
- Egyszerű FP VE (+/-)
- Összetett FP VE (MUL)
- Összetett FP VE (DIV)

Az 1-es portra két VE csatlakozik:

- Összetett FX VE
- Címszámító egység (JMP)

2-es port: - Címszámító egység (LOAD, adatbehívási címek)

3-as és 4-es port: - egy-egy Címszámító egység (STORE, tárolási címekhez))

A reorder buffer (ROB) működése:

- tartalmazhatja az átnevezési regisztereket (nem minden esetben)
- vezérli a várakoztató állomást
- vezérli a memória átrendező puffert
- kiíráskor (visszaírás) elvégzi a rekonverziót is

A ROB-ot folyamatosan frissíteni kell a függőségek mielőbbi feloldása érdekében (tehát fontos, hogy az átnevezési regiszterazonosítók helyére minél előbb bekerüljenek az operandusok!).

Amikor egy eredmény előáll, a processzor asszociatív keresést végez regiszterazonosítók alapján (forrásoperandus mezők!), hogy a várakozó utasítások között van-e olyan, aminek operandusa a most előállt eredményt igényli.

Ha talál ilyet, az eredményt betölti a regiszterazonosító helyére és az állapot bitet 1-re állítja. Ha mindkét forrás operandus állapot bitje 1 lesz, a ROB kiküldi az utasítást a végrehajtó egységek felé.

A működés egy kör alakú puffer segítségével szemléltethető.

A várakoztató állomás a bemeneti és a végmutató között található puffer regiszterekből áll.

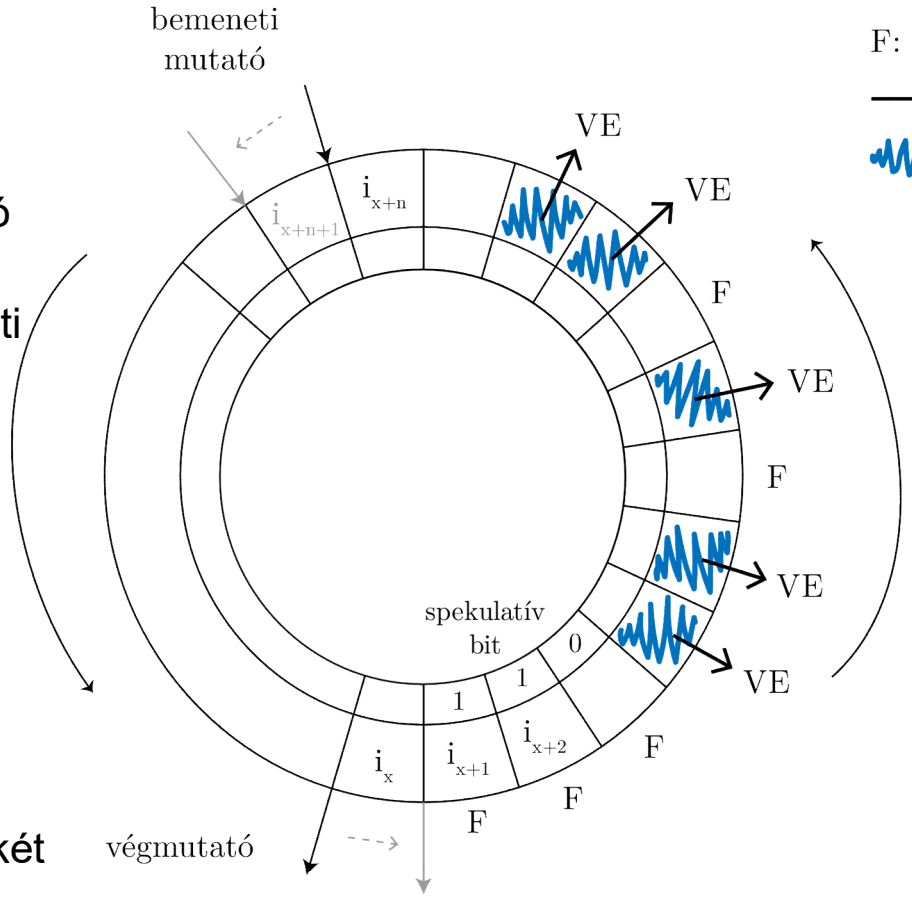
ÓBUDAI EGYETEM

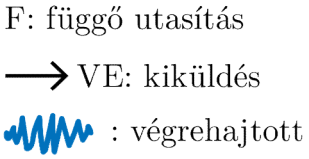
Az utasítások a bementi mutatónál töltődnek be. A ROB elvégzi a regiszterátnevezéseket és beírja a forrás operandusokat, majd később az eredményt is az átnevezési regiszterekbe.

A bemeneti és végmutató között bármely független utasítás kiküldhető a végrehajtó egységek felé.

Ha egy utasítás kiírásra került, a bemeneti és végmutató is egyet elmozdul az ábrán az óramutató járásával ellentétesen.

A ROB minden utasításhoz hozzárendel egy spekulatív bitet. Azoknál az utasításoknál, amiknél még nem biztos a végrehajtás szükségessége (olyan elágazás részei, ahol a feltétel még nem került kiértékelésre), a spekulatív bit értékét 1-re állítja!





Amíg a spekulatív állapot fennáll, az eredmény nem írható ki, így biztosítva szekvenciális konzisztenciát és az architekturális regiszterek hibás értékkel történő feülírását!

Feltétel kiértékelése után:

- Helyes irány esetén a spekulatív bit értékét 0-ra állítja és az eredmény kiírható
- Hibás becslés esetén az utasítás törlésre kerül a ROB-ból, az átnevezési regiszterek felszabadításra kerülnek, és a helyes irányba történő utasítás lehívásra kerül

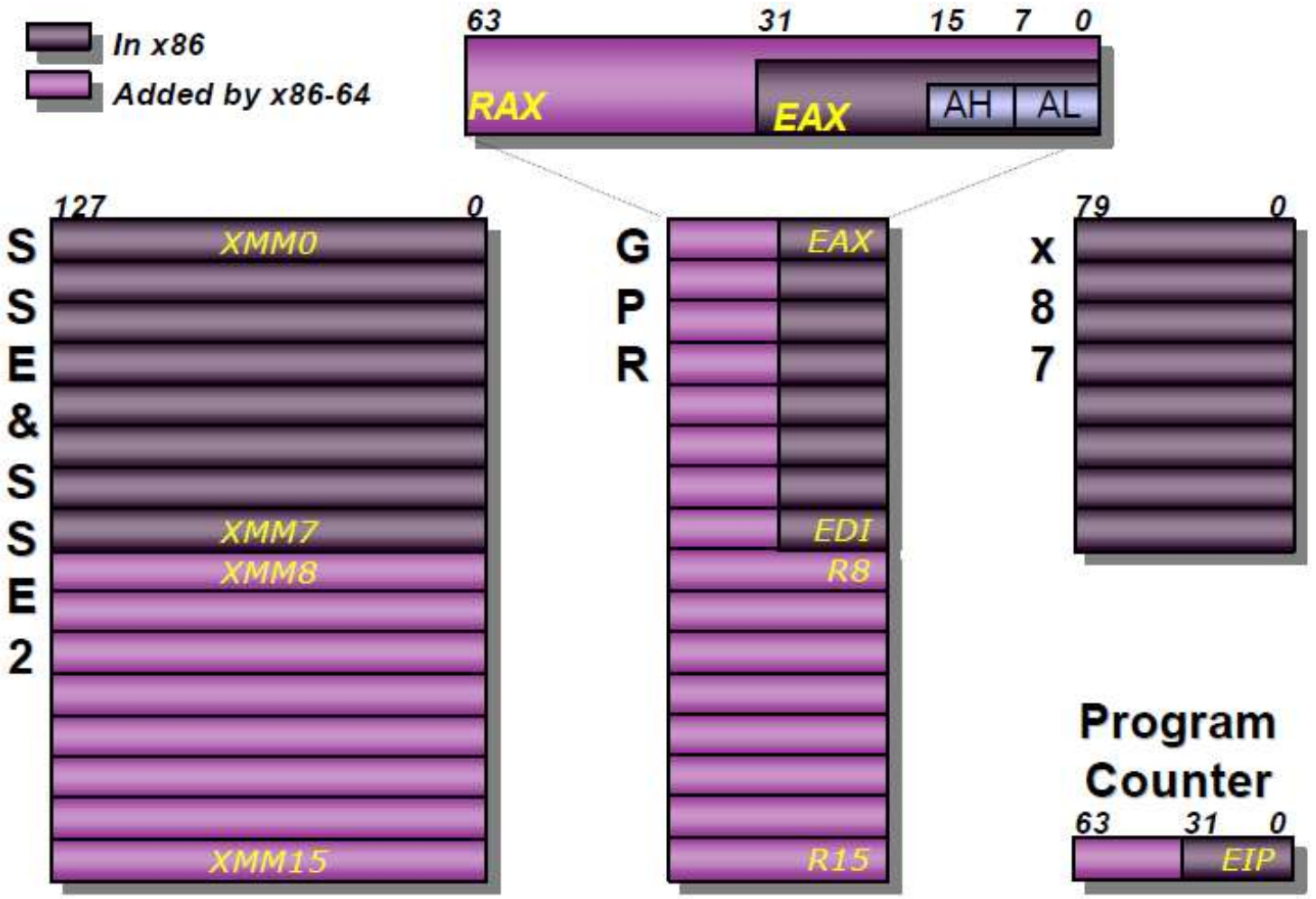
<u>Kiírási szabályok:</u>

Kiíráskor az eredmények az architekturális regiszterekbe (majd a memóriába / cache-be) kerülnek és az átnevezési regisztereket felszabadítja!

a) Egy utasítás csak akkor írható ki a ROB-ból, ha már minden korábbi utasítás kiírásra került
b) Spekulatív állapotban lévő utasítás nem írható ki a ROB-ból!
c) Egy CISC utasítás akár több RISC utasítássá is konvertálódhat a végrehajtáskor, de a rendszer kívülről CISC architektúrájú.
   Ezért az egy CISC utasításhoz tartozó RISC utasítások eredményei csak akkor írhatók ki, ha már mindegyiknek előállt az eredménye. Így az eredmények egyszerre kerülnek kiírásra és megtörténhet a rekonverzió.

X86-64 új regiszterek (2000-es évek eleje):

Új utasítások is szükségesek a 64 bites regiszterekhez!



Harmadik generációs szuperskalár architektúrák:

A második generációs szuperskalárok az eddig említett módszerekkel elérték az architektúra teljesítménybeli korlátait, így más irányba kellett fejleszteni (Nem maradt több kihasználható párhuzamosság ILP szinten!).
Így jelent meg 1997 környékén az utasításon belüli párhuzamosság, amivel elérkeztek a harmadik generációs szuperskalár CPU-k. Egyesek szerint nem nevezhető harmadik generációnak, hanem csak a másodiknak a kibővítése multimédiás/3D képességekkel.

Utasításon belüli párhuzamosság típusai:

- *Duál műveleti utasítások* (például multiply-add: y = ax+b.): Egy utasítással több művelet is elvégezhető. Elsősorban a numerikus feldolgozást segíti. Általános alkalmazásokban ritka.

- *SIMD utasítások*: egyetlen utasításon belül több operanduson ugyanazt a műveletet végezzük el. Például: Intel MMX PADDW utasítása, ami 4 darab 16 bites fixpontos összeadást végez el egy műveletben

- *VLIW architektúrák*

ÓBUDAI EGYETEM

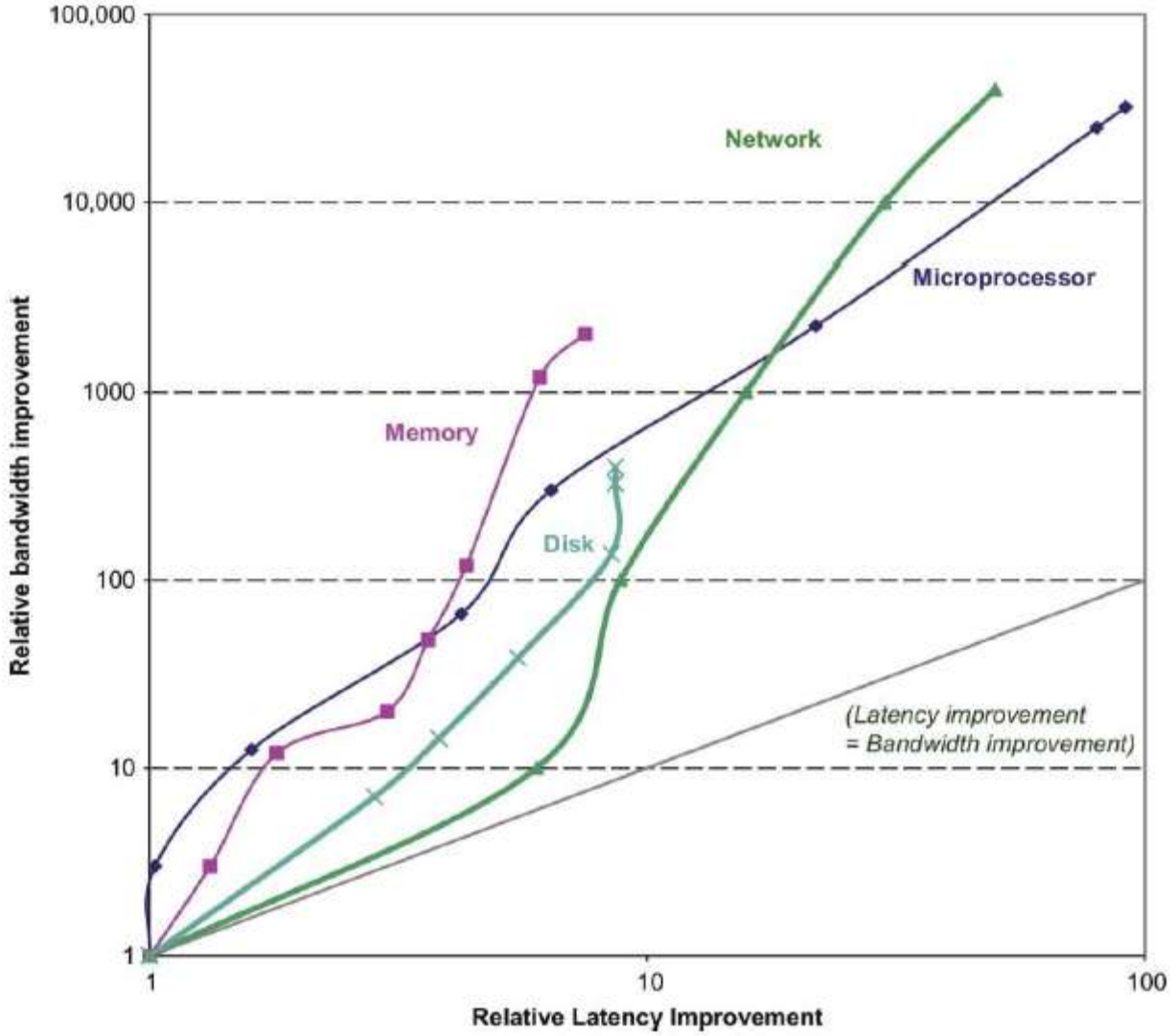
Teljesítmény mérföldkövek sávszélesség és késleltetés szempontjából:

(sávszélesség növekedés több ezerszeres, késleltetés csökkenés ~20-40-szeres)

Milestone	1	2	3	4	5	6
Microprocessor	16-bit address/bus, microcoded	32-bit address/bus, microcoded	5-stage pipeline, on-chip I & D caches, FPU	2-way superscalar, 64-bit bus	Out-of-Order, 3-way superscalar	Superpipelined, on-chip L2 cache
Product	Intel 80286	Intel 80386	Intel 80486	Intel Pentium	Intel Pentium Pro	Intel Pentium 4
Year	1982	1985	1989	1993	1997	2001
Die size (mm2)	47	43	81	90	308	217
Transistors	134,000	275,000	1,200,000	3,100,000	5,500,000	42,000,000
Pins	68	132	168	273	387	423
Latency (clocks)	6	5	5	5	10	22
Bus width (bits)	16 bits	32 bits	32 bits	64 bits	64 bits	64 bits
Clock rate (MHz)	12.5	16	25	66	200	1500
Bandwidth (MIPS)	2	6	25	132	600	4500
Latency (nsec)	320	313	200	76	50	15
Memory Module	DRAM	Page Mode DRAM	Fast Page Mode DRAM	Fast Page Mode DRAM	Synchronous DRAM	Double Data Rate SDRAM
Module width	16 bits	16 bits	32 bits	64 bits	64 bits	64 bits
Year	1980	1983	1986	1993	1997	2000
Mbits/DRAM chip	0.06	0.25	1	16	64	256
Die size (mm2)	35	45	70	130	170	204
Pins/DRAM chip	16	16	18	20	54	66
Bandwidth (MB/s)	13	40	160	267	640	1,600
Latency (nsec)	225	170	125	75	62	52

ÓBUDAI EGYETEM

Sávszélesség és késleltetés a számítógép alrendszerei esetén (1980-2004):

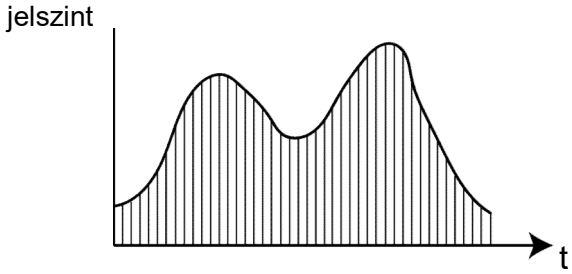


A SIMD architektúra jellemzői:

- A SIMD utasítások egy kibővített (multimédiás) utasításkészletet alkottak ➡ a logikai architektúra kiterjesztését vonja maga után
- Egy utasításhoz 8 bemenő operandusra szükség ➡ sokszorosára nő a memóriaigény
- Memória sávszélesség növekedés ➡ emiatt az L2 cache lapkára kerül, vagyis beépül a processzorba (ez korábban például Pentium II-nél külön foglalatba került, ez volt a slot 1)
- A gyorsabb megjelenítés érdekében a buszrendszer is bővült:
  - Megjelent az AGP
  - Majd később a PCI Express
- 3D alkalmazásokat és játékokat tett lehetővé
- Először az Intel Pentium Pro processzorban (1997) jelent meg az MMX (Multimedia Extension), ami egyelőre csak fixpontos multimédiás utasításokat támogatott
- Az MMX technológia kibővítésre került a Pentium III processzorban (1999) SSE (Streaming SIMD Extensions) néven, ami már lebegőpontos utasításokat is tartalmaz
- Általános célú alkalmazásoknál nem nőtt a teljesítmény, viszont multimédiás alkalmazásokat (kép, videó, 3D játék) jelentősen meggyorsította, mivel ezeknél a nagy mennyiségű, függőség nélküli adatokat csak feldolgozni szükséges.

Fixpontos SIMD utasítások feldolgozása:

A fixpontos SIMD utasítások elsősorban a hang és a pixeles megjelenítést használó multimédiás alkalmazásokat gyorsítják. Egy utasításban 2-8 operandus lehetséges.



Hangfeldolgozás:

A hangot a feldolgozáshoz először digitalizálni kell. Ezt az analóg jelből történő meghatározott időközönkénti mintavételezéssel érhetjük el. Ebben a formában már tárolható és visszajátszható. A minőséget befolyásolja a mintavételezés sűrűsége (felbontás). Ennek növelése jobb minőséget, de nagyobb erőforrásigényt jelent.

Például 50 kHz-es mintavételezés 50 000 mintavétel/másodpercet jelent (kb. DVD minőség). Emberi beszéd hozzávetőlegesen 8 KHz-es mintavétellel értelmezhető.

A hangfeldolgozás másik jellemzője, hogy egy adott mintát hány biten tároljuk (minta nagysága). 8 bit esetén 256 féle hangmagasságot tudunk tárolni. A cél, hogy minél jobban hasonlítson a digitális jel az eredeti analóg függvényhez.

Képfeldolgozás

A képek esetén képpontokat tárolunk el, minden képponthoz hozzá kell rendelni a színét és a fényességét. A színek három értékből állnak össze: piros, zöld és kék. A színek előállításához használjuk a színfordítási táblázatot, amiben minden színhez egy számérték van hozzárendelve. Ez alapján áll elő a színskála kódolás.

Kezdetben 1 biten tárolták az értékeket (világos/sötét), aztán a színes kijelzők megjelenésével ez 8, később 16 (high color), végül 24 (true color) bitre bővült. Ezenfelül létezik a 32 bites színskálán is. 24 biten ábrázolják a színeket, a többi 8 bit pedig az alfa csatorna, amin az effekteket ábrázolják (például átlátszósági mutató).

Probléma: A hang és képfeldolgozás során a problémát a nagy mennyiségű adat tárolása és feldolgozása okozza.

Megoldás

- Kezdetben célhardverrel: multimédiás kártyákkal
- Később a CPU multimédiás bővítésével

Logikai architektúra kibővítése:

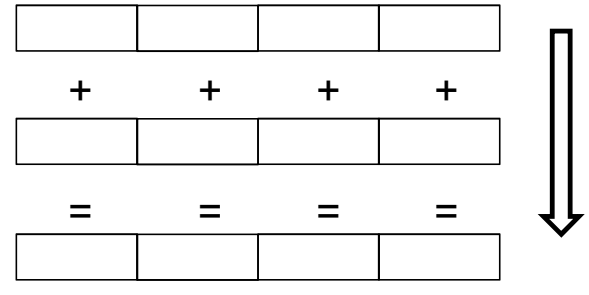
Bit-blokk átvitel és pakolt adattípusok!

A multimédiás megjelenítés tipikus művelete a bit-blokk átvitel. Lényege, hogy minden megnyitott ablakot egy bit-blokként kezelünk, így nem egyesével dolgozunk az operandusokkal. Ez azért szükséges, mivel például egy 1920x1080-as képernyő 24 bites színskála esetén körülbelül 6 MB adatot jelent. Ez 30 képkocka/másodperc esetén 180 MB/sec feldolgozási sebességre lenne szükség FX adatokat feltételezve. Ez a sebesség még ma is soknak számít, ezért a megoldására egyrészt tömörítést alkalmaztak, másrészt pedig architektúrális megoldást vezettek be. Az architektúrális megoldás egy új fixpontos adattípus bevezetése, a pakolt adattípus. A pakolt adattípus az Intel MMX processzorokban jelent meg először.

Típusai:

- Pakolt byte (8 bit hossz x 8 db = 64 bit)
- Pakolt félszó (16 bit hossz x 4 db = 64 bit)
- Pakolt szó (32 bit hossz x 2 db = 64 bit)

Példa:



A SIMD feldolgozási elvnek köszönhetően ezeket az adattípusokat egy utasítással is tudták módosítani az Intel MMX CPU-k.

Ezekhez új utasítások kellettek mindhárom pakolt adattípusra:

- 4 alapművelet
- Logikai műveletek

A 1990-es évek második felében nagyrészt az internet elterjedése miatt lettek szükségesek az új adattípusok.

Fizikai architektúra kibővítése:

A pakolt adattípusok feldolgozásához az architektúra fizikai módosítására is szükség volt:

- 64 bites regiszterek (új regiszterek helyett a lebegőpontos regiszterek használata, amik 80 bites voltak, így megfeleltek a 64 bit tárolására)
- 64 bites buszrendszer (először belső, majd külső buszok is)

A Pentium II-es processzorokban már két MMX végrehajtó egység került beépítésre, ami tényleges gyorsítást jelentett a függőségek hiánya miatt!

<u>Lebegőpontos SIMD utasítás feldolgozás:</u>

A lebegőpontos SIMD utasítások a vektoros és 3D képfeldolgozást gyorsítják, aminek használatával a feldolgozandó adat mennyisége csökkenthető. Általában 2-4 operandussal rendelkeznek utasításonként.

Vektoros képfeldolgozás:

Egyenesekkel vagy görbékkel határolt objektumok geometriai jellemzőkkel leírhatók.

Példa: egy egyenes ábrázolása Full HD kijelzőn keresztben 1920 képpontot jelent fixpontosan, ennyi adatot kell eltárolnunk. Vektorosan azonban elég a két végpontot eltárolni, ami lényegesen kevesebb adatot jelent. Kör esetében elégséges a középpont és a sugár tárolása. A módszer eredménye, hogy jóval kevesebb adatot kell tárolnunk, viszont a megjelenítéséhez sokkal többet kell számolni, de a számítási kapacitás rendelkezésre áll. Képek tárolása is lehetséges így, ha azokat geometriai objektumokra bontjuk fel, majd ezeknek az adatait tároljuk.

Előnye, hogy a képek könnyen mozgathatók, kicsinyíthetők és nagyíthatók. Lebegőpontos ábrázolás szükséges, mivel a geometriai műveletek során törtekkel is kell számolni!

Hátránya, hogy 2D ábrázolásnál az éles határok miatt kevésbé hasonlít a valóságra, ennek kiküszöbölésére textúrákat helyeznek a képekre, a fény-árnyék hatások segítségével élénkítik, vagy tompítják a színeket, összemossák az éles határokat!

3D ábrázolás:

3D ábrázolásnál értelmezni kell egy harmadik dimenziót a 2D-s képen. Az élethű megjelenítéshez biztosítani kell:

- a párhuzamosok a végtelenben összetartanak
- a közeli objektumok nagyobbnak, élesebbnek látszanak
- biztosítani atmoszférikus hatást: a távolabbi objektumok halványabbak és kékesebbek

A 3D-s filmeknél minimum 15-30 FPS-t kell előállítani. Ehhez minimum 500 000 objektum/másodperces megjelenítésre van szükség ➡ nagy mennyiségű lebegőpontos adat feldolgozására van szükség.
Ilyen lebegőpontos feldolgozásra először az Intel Pentium III-as processzora volt képes az SSE kiterjesztéssel (1999, Intel Katmai CPU).
Jellemzői:

- 500 MHz-en működött
- Számítási teljesítménye körülbelül 2 GFLOPS/sec (2 milliárd FP művelet/sec)

A mai processzorok jellemzően néhány TeraFLOPS/sec teljesítményt biztosítanak.

Logikai architektúra kibővítése:

Logikai oldalról nézve mindez egy új utasításkészletet jelentett: SSE. Körülbelül 70 darab új utasítást tartalmazott, amik a lebegőpontos pakolt adattípusokkal tudtak műveletet végezni.
Az adatok 128 bites pakolt típusok voltak:

- 4 x 32 bit – egyszeres pontosság
- 2 x 64 bit – kétszeres pontosság

Ezek megfelelnek az IEEE 754-es szabványnak, ami a lebegőpontos számokra vonatkozik.

Fizikai architektúra kibővítése:

Itt már szükség volt ténylegesen új regiszterek létrehozására, így 8 db új 128 bites regiszter került a processzorokba. Ezek mentéséről és betöltéséről az OS vagy az adott szoftver gondoskodik. Az első szoftveres támogatás a Windows 98-ban jelent meg.
Az új regiszterek bevezetésével 1985 óta (13 év után) először változtatta meg az Intel a processzorok regiszterkészletét. Következménye, hogy a megszakítás rendszert is át kellett tervezni és ez az OS gyártójával való együttműködést igényelte.

VLIW architektúrák:

(very long instruction word)

Fejlesztés kezdete: 70-es évek

Működési elve:

- időbeli és térbeli párhuzamosság (hasonlóan a szuperskalár architektúrákhoz)
- a függőségek kezelését a compiler végzi (ellentétben a szuperskalárral)
- teljesen új utasításformátummal rendelkezik ➡ nem kompatibilis egyetlen korábbi architektúrával sem
- 1 darab, több (egymástól független) utasítást tartalmazó utasításszó
- a hardver teljesen párhuzamosított kódot kap végrehajtásra (statikus kezelés)
- minden utasítás egy-egy végrehajtó egységet közvetlenül vezérel
- az utasításszó hossza függ a végrehajtó egységek számáról (akár 1024 bit)

Előfeltételek:

- függőségek kezelése
- utasítások hatékony ütemezése

A szuperskalár architektúráknál a CPU-ra hárul a párhuzamosítható utasítások felismerése és a párhuzamosan végrehajtható kódrészletek kinyerése! (dinamikus megoldás, extra hardverek segítségével)
Ezért egy modern szuperskalár CPU-ban a tranzisztorok jelentős része nem az utasítások végrehajtását, hanem a párhuzamosságok felismerését, kezelését, és az utasítások ütemezését szolgálja!

VLIW architektúrák esetében ezt a fordítóprogram végzi (compiler), maga a hardver teljesen párhuzamosított kódot kap végrehajtásra! (statikus kezelés)
Következmény, hogy a VLIW architektúrák egyszerűbb felépítésűek lehetnek, kevesebb tranzisztor szükséges hozzájuk!

Előnyei a szuperskalár architektúrákhoz képest:

- Ugyanolyan fokú párhuzamosság mellett jóval egyszerűbb felépítés ➡ kevesebb tranzisztor!
- Azonos komplexitás mellett jóval nagyobb feldolgozási szélesség valósítható meg!

Hátrányai:

- Statikus ütemezés (a compiler felelőssége minden egyes függőség típus kezelése)
  ➡ a compilernek nagyon pontosan kell ismernie a fizikai architektúrát. Például a végrehajtó egységek számát és típusát, ismétlési és késleltetési idejüket, gyorsítótárak betöltési késleltetését, …. Ez bonyolult és erősen architektúra függő.
- Elvárás, hogy a compiler agresszív párhuzamos optimalizálást hajtson végre annak érdekében, hogy óraciklusonként elegendő párhuzamosan végrehajtható utasítást találjon!
- Minden új konfigurációhoz új compilert kell írni. (Itanium esetén részben kezelték)
- Teljesen új utasításkészletet (ISA-t) használ: „hosszú utasítás szó” ➡ nem kompatibilis a korábbi programokkal, rendszerekkel.

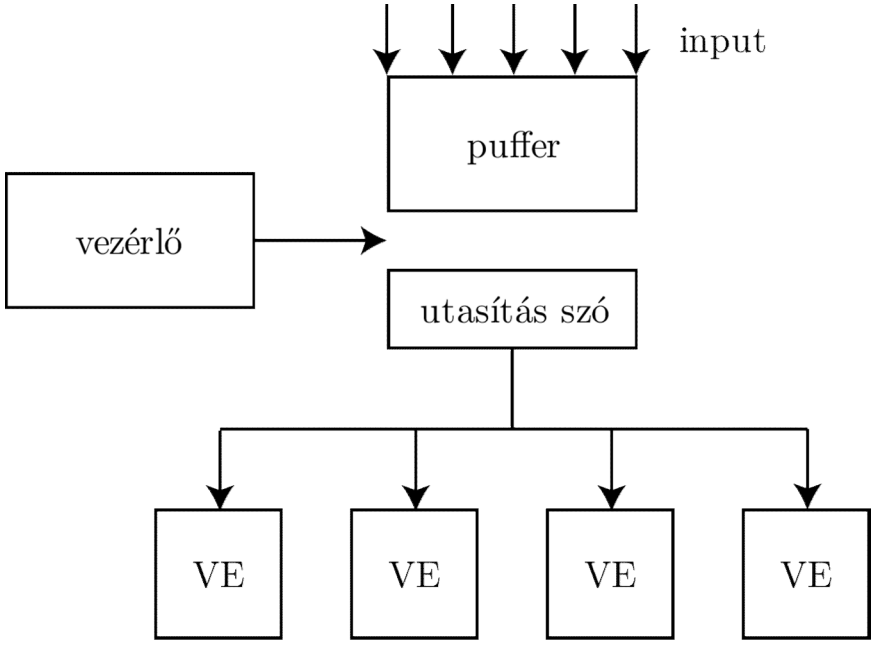
Logikai ábrázolás:

A vezérlő logika állítja elő a pufferen keresztül bejövő adatokból az utasítás szavakat, hogy azok kiküldhetőek legyenek a végrehajtó egységek részére párhuzamosan és időben egymástól függetlenül. Minden ciklusban szükséges ugyanannyi utasítást találni.

Típusai

Két típusú VLIW architektúrát különböztetünk meg:

- korai (széles) VLIW architektúra
- késői (keskeny) VLIW architektúra



Első generációs (széles) VLIW-ek:

A széles VLIW architektúrákat az 1970-es években fejlesztették. Nagyjából 10-20 végrehajtó egységgel rendelkeztek. Az utasítás szavak 256-1024 bit hosszúak voltak. A compilerek ezt még nem tudták kihasználni a kezdetleges függőség kezelés miatt, vagyis a CPU-t nem tudták ellátni elegendő független utasítással. ➡ Nem volt hatékony!

Második generációs (keskeny) VLIW-ek:

Fejleszétse a 90-es évek elején kezdődött (Intel, Hewlett-Packard: Itanium project)
2-8 végrehajtó egységet tartalmaztak és lényegesen nagyobb teljesítményt biztosítottak a fejlettebb compilerek segítségével, mint a korabeli futószalag processzorok.
A digitális jelfeldolgozás megjelenése szintén segített a keskeny VLIW architektúrák elterjedésében ➡ sok független utasítás!
A sok nehézség miatt csak a 2000-es évek elején jelentek meg az első Itanium processzorok, 4-8 utasítást tartalmazó utasítás szóval (~10 év fejlesztés ➡ nagyon hosszú idő!)
Az Intel ezeket EPIC (Explicitly Parallel Instruction Computing) processzoroknak nevezte

Jellemzői:

- Fordítási időben történő ütemezés: a Compiler több információval rendelkezik a szoftverről, mint a CPU ➡ a CPU-ról lekerült a bonyolult ütemezési mechanizmus ➡ nagy mennyiségű tranzisztor szabadul fel ➡ több VE és több regiszter fér el a lapkán!
- A plusz tranzisztorokat extra regiszterek kialakítására használták fel ➡ regiszterekben gazdag architektúra
- Tervezési elv, hogy a lehető legtöbb utasítás fusson párhuzamosan, akkor is, ha egy részük teljesen feleslegesen hajtódik végre ➡ növeli a fogyasztást!
- A fordítási időben történő ütemezés miatt lehetőség van az ún. spekulatív adatbetöltésre, azaz az utasítások olyan átrendezésére, mint pl.: a LOAD és a USE (adat megjelenése) közé más, független utasítások kerüljenek! (memória késleltetés miatt!) ➡ optimalizáló Compiler!

Tervezési elv előnyei:

- Vezérlés függőség (például feltételes elágazás) esetén mindkét ág végrehajtása kerül
  ⟶ csökken az elágazási késleltetés
- Spekulatív adatbetöltés: optimalizáló compiler funkció, ami annyit jelent, hogy a compiler úgy állítja össze a gépi kódot, hogy amíg például egy adat betöltésére várakozik, addig más utasításokat hajt közben végre.
- Alkalmas nagy megbízhatóságú szerverekben történő alkalmazásra.

Tervezési elv hátrányai:

- Energiapazarló, ezért mobil eszközökben nehezen alkalmazható a magasabb fogyasztás miatt.
  ⟶ elsősorban szerverekben terjedt el, valamint digitális jelfeldolgozást végző eszközökben

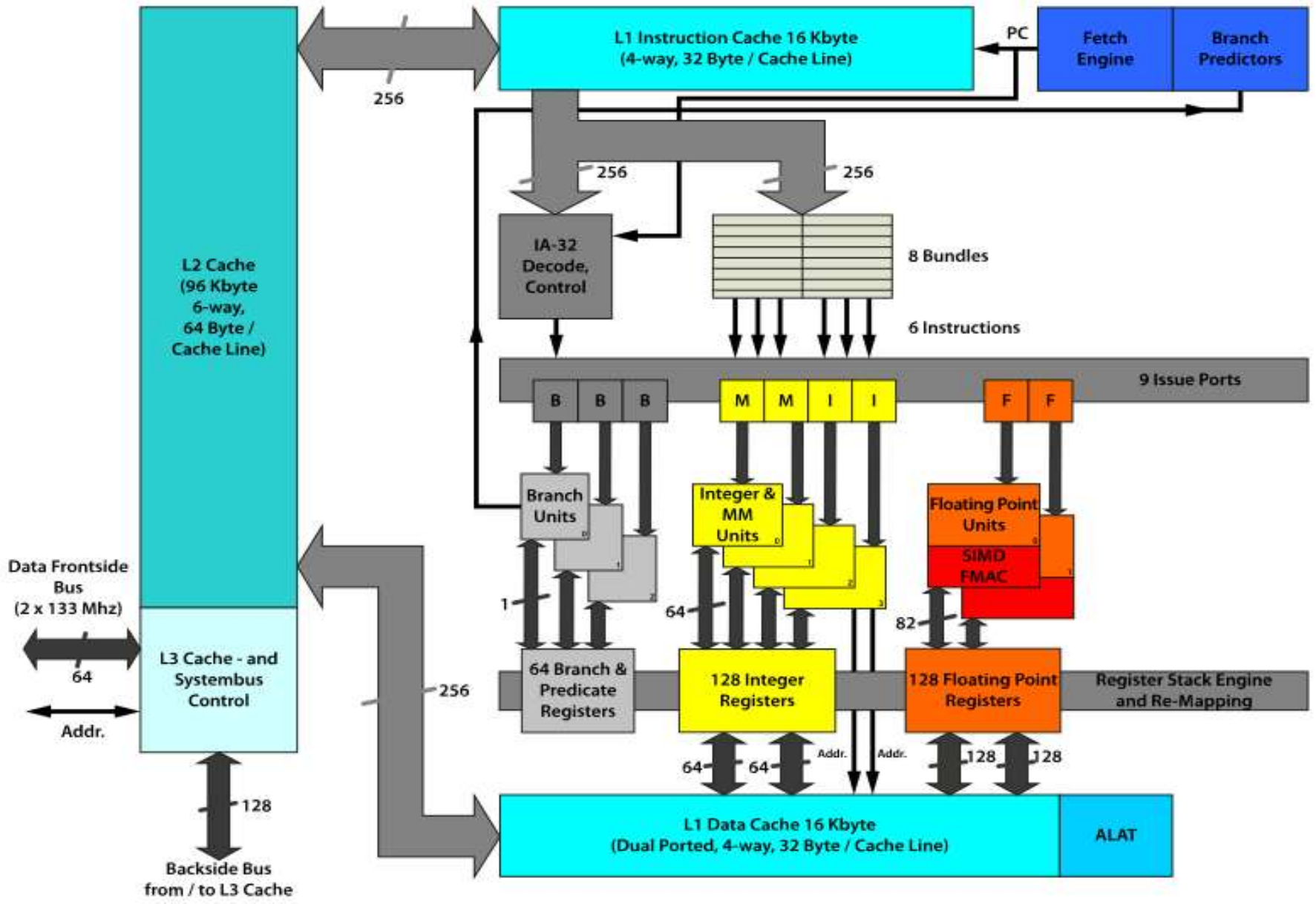
<u>IA-64 architektúra:</u>

Az Intel a VLIW architektúra továbbfejlesztéseként alkotta meg az „új ISA”-t, azaz az IA-64 architektúrát. Ez volt az Intel első 64 bites architektúrája. A tervek között szerepelt az x86 kivezetése is, viszont az IA-64 nem volt kompatibilis az x86-ra írt szoftverekkel. Bár a Windows XP-t még megírták IA-64-re is, a drága gyártásnak és a többmagos, 64 bites, de visszafelé kompatibilis x86_64-es processzorok megjelenésének köszönhetően a fejlesztést 2016-ban végleg leállították.

Maga a VLIW architektúra egy modern, előremutató és megbízható rendszer volt, a fejlesztés leállításának okai részben üzletiek voltak: amíg az Intel és a HP az IA-64-et fejlesztette, az AMD megalkotta az x86_64 architektúrát, ami kompatibilis volt az x86-os szoftverekkel. Nem kizárható, hogy a jövőben még visszatérnek a VLIW CPU-k.

Legnagyobb problémái:
- Hosszú fejlesztési idő (a szuperskalárok is fejlődtek!)
- A fejlesztésnek meg kell térülnie ➡ magas piaci ár
- Kompatibilitási problémák
- Nehéz jó Compiler-t írni, nem képes kihasználni teljesen az architektúrát
- Többmagos CPU-k megjelenésével és a változó frekvencia bevezetésével a szuperskalárok már jobb alternatívát nyújtottak

Végrehajtási modell (Intel Itanium):
- 128 db 64 bites regiszter (FX), 128 db 82 bites regiszter (FP)


L1 Instruction Cache 16 Kbyte
(4-way, 32 Byte / Cache Line)
PC
Fetch Engine
Branch Predictors
256
256
256
L2 Cache
(96 Kbyte
6-way,
64 Byte /
Cache Line)
IA-32
Decode,
Control
8 Bundles
6 Instructions
9 Issue Ports
B
B
B
M
M
I
I
F
F
Branch Units
Integer & MM Units
Floating Point Units
SIMD FMAC
1
64
82
64 Branch & Predicate Registers
128 Integer Registers
128 Floating Point Registers
Register Stack Engine and Re-Mapping
64
64
Addr.
Addr.
128
128
Data Frontside Bus
(2 x 133 Mhz)
64
Addr.
L3 Cache - and Systembus Control
128
Backside Bus
from / to L3 Cache
256
L1 Data Cache 16 Kbyte
(Dual Ported, 4-way, 32 Byte / Cache Line)
ALAT