

# Korszerű számítógép architektúrák II.

Készítette: Simon Péter<sup>1</sup>

2022. január 1.

<sup>1</sup>Hallgatói jegyzet Dr. Sima Dezső előadásai alapján

# Tartalomjegyzék

<b>1. Bevezetés</b>	<b>6</b>
1.1. A tárgy célja . . . . .	6
1.2. Csíkszélesség . . . . .	6
1.3. Pentium 4 processzorcsalád . . . . .	6
1.4. Intel Core processzorok fejlesztése . . . . .	6
1.5. Gyártási technológia . . . . .	7
1.6. Utasításkészlet fejlődése . . . . .	7
1.7. A mikroarchitektúra fontosabb újításai . . . . .	7
1.8. Fejlődés IPC-ben kifejezve . . . . .	8
1.9. Frekvencia vs tranzisztorok száma . . . . .	8
<b>2. Intel Core 2 család</b>	<b>10</b>
2.1. Kategorizálás . . . . .	10
2.2. Kliens processzorok fejlődése . . . . .	10
2.2.1. Magok száma . . . . .	10
2.2.2. Memória csatornák . . . . .	10
2.2.3. Memória vezérlő elhelyezése . . . . .	11
2.2.4. Memória sebessége . . . . .	11
2.2.5. Grafika . . . . .	11
<b>3. HEDT processzorok fejlődése</b>	<b>13</b>
3.1. Bevezetés . . . . .	13
3.2. PCIe vonalak . . . . .	13
3.3. Magszámok . . . . .	13
3.4. Memóriacsatornák . . . . .	14

3.5. Overclocking . . . . .	14
3.6. Fogyasztás . . . . .	14
3.7. Grafika . . . . .	14
<b>4. Nagy teljesítményű, két foglalatos szerver processzorok fejlődése</b>	<b>15</b>
4.1. Szerver platformok osztályozása . . . . .	15
4.2. Méret, tranzisztorok . . . . .	15
4.3. Fogyasztás . . . . .	15
4.4. Lábak száma . . . . .	15
4.5. Magok száma . . . . .	16
4.6. Intel Cascade Lake 9200-AP . . . . .	16
4.7. Magszámok és tranzisztorok növekedése . . . . .	16
4.7.1. Kivételek . . . . .	17
4.8. Memória sebességek fejlődése . . . . .	17
<b>5. Mobil és táblagép processzorok fejlődése</b>	<b>19</b>
5.1. Bevezetés . . . . .	19
5.2. Fogyasztás csökkentése . . . . .	19
5.2.1. Intel Atom . . . . .	19
5.2.2. AMD . . . . .	19
5.2.3. A frekvencia csökkentése . . . . .	20
5.3. Piaci részesedés . . . . .	20
5.4. Szélesség fejlődése . . . . .	22
<b>6. Disszipációkezelés</b>	<b>23</b>
6.1. A disszipációkezelés fontossága . . . . .	23
6.2. A disszipáció . . . . .	23
6.2.1. Kiszámítása . . . . .	23
6.3. TDP . . . . .	24
6.4. Intel processzorok disszipációjá a fejlődés során . . . . .	24
6.5. Disszipációcsökkentő kezdeményezések . . . . .	25
6.5.1. Energy Star . . . . .	25
6.5.2. Az AMD hatékonysági kezdeményezése . . . . .	26

6.5.3. Szerverek energiahatékonysága . . . . .	26
6.6. ACPI . . . . .	26
6.7. A disszipációkezelés technikái . . . . .	27
6.7.1. Kezelés áramköri szinten . . . . .	27
6.7.2. Kezelés platform szinten . . . . .	28
6.7.3. Kezelés CPU szinten . . . . .	28
6.8. Az Intel Turbo Boost techológiája . . . . .	33
6.8.1. Bevezetés . . . . .	33
6.8.2. Első generáció . . . . .	33
<b>7. Az AMD Zen processzorai</b>	<b>35</b>
7.1. Bevezetés . . . . .	35
7.2. Áttekintés . . . . .	35
7.3. A Zen család tervezési elvei . . . . .	36
7.3.1. Többlapkás felépítés . . . . .	36
7.3.2. Moduláris kialakítás . . . . .	37
7.3.3. A CPU-k és GPU-k három szintű hierarchikus tervezése . . . . .	37
7.4. A Zen magok fejlődése . . . . .	39
7.5. A CCX-ek fejlődése . . . . .	40
7.6. A grafikai magok fejlődése . . . . .	41
7.6.1. A Vega CU felépítése . . . . .	42
7.7. A Zen processzorok építőelemei és felépítése . . . . .	42
7.7.1. Első generáció . . . . .	42
7.7.2. Második generáció . . . . .	43
7.7.3. Harmadik generáció . . . . .	43
7.8. Infinity Fabric . . . . .	44
7.8.1. Scalable Control Fabric . . . . .	44
7.8.2. Scalable Data Fabric . . . . .	44
7.8.3. Példa: első generációs, GPU nélküli desktop Zen CPU . . . . .	45
7.8.4. Példa: első generációs EPYC szerver CPU . . . . .	45
7.8.5. Példa: első generációs 2S EPYC szerver CPU . . . . .	47
7.8.6. Második generációs IF . . . . .	47

7.9.	MCM tokozás . . . . .	47
7.10.	A Zen processzorcsalád tagjai . . . . .	48
7.11.	Példa: Zen 3 alapú, GPU nélküli Ryzen 5000 . . . . .	48
<b>8.</b>	<b>ARM processzorok</b>	<b>49</b>
8.1.	Terminológia . . . . .	49
8.2.	Áttekintés . . . . .	49
8.2.1.	Üzleti modell . . . . .	49
8.2.2.	Licenszelés . . . . .	50
8.2.3.	Az ARM dominanciájának okai . . . . .	50
8.3.	Az ARM ISA fejlődése . . . . .	50
8.4.	Számítási képességek fejlődése . . . . .	51
8.4.1.	Fixpontos SIMD kiterjesztés . . . . .	51
8.4.2.	Lebegőpontos műveletvégzés . . . . .	52
8.4.3.	SVE regiszterkészletre alapuló SIMD kiterjesztések . . . . .	52
8.4.4.	SVE 2 . . . . .	52
8.5.	Az ARM CPU-k áttekintése . . . . .	53
8.5.1.	Cortex CPU-k . . . . .	53
8.5.2.	Cortex profilok . . . . .	53
8.6.	A Cortex-A profilú CPU-k fejlődése (v7 és v8) . . . . .	54
8.6.1.	Az ARMv7-8 CPU-k fejlődése . . . . .	54
8.6.2.	Az ARM MPCore kiegészítés . . . . .	54
8.6.3.	A big.LITTLE technológia . . . . .	55
8.6.4.	Cortex-A processzorok teljesítmény növekedése . . . . .	55
8.6.5.	Az ARM Cortex-A processzorok elterjedése . . . . .	55
8.7.	A Cortex-A profilú CPU-k fejlődése (v8.2) . . . . .	55
8.7.1.	Különbségek az ARMv8-hoz képest . . . . .	56
8.7.2.	Fejlesztési roadmap . . . . .	56
8.7.3.	A Cortex-X1 processzor . . . . .	56
8.8.	Az ARMv9 ISA . . . . .	57
8.8.1.	Főbb fejlesztési területek . . . . .	57
8.8.2.	Vektor és digitális jelfeldolgozás (SVE2) . . . . .	57

8.8.3. ML . . . . .	58
8.8.4. Többszálú programozás támogatása . . . . .	58
8.8.5. Biztonság . . . . .	58
8.9. Az ARMv9 mikroarchitektúra . . . . .	58
8.9.1. A DSU . . . . .	58
8.9.2. A DynamIQ klaszter fejlődése . . . . .	58
8.9.3. Memory Tagging Extension . . . . .	59
8.9.4. Hozzáférési szélesség növelése . . . . .	59
8.10. ARMv9-A-t megvalósító processzorok . . . . .	59
<b>9. Mobil processzorok architektúrája</b>	<b>60</b>
9.1. Bevezetés . . . . .	60
9.2. ARM licenszek . . . . .	60
9.3. Az ARM alapú mobil processzorok fejlődése . . . . .	60
9.4. Az Intel Atom és az AMD Cat család . . . . .	61
9.5. Windows táblagépekre fejlesztett Core 2 processzorok . . . . .	61
9.6. ARM processzorok az Apple Mac eszközeiben . . . . .	61
9.7. Mobil processzor architektúrák fejlődése . . . . .	61
9.7.1. Szimmetrikus többmagos processzorok . . . . .	62
9.7.2. big.LITTLE architektúrák . . . . .	62
9.7.3. DynamIQ magklaszterek . . . . .	64
9.8. Magok szélességének fejlődése . . . . .	65
9.8.1. A Samsung Mongoose családjának fejlődése . . . . .	65

# 1. fejezet

## Bevezetés

### 1.1. A tárgy célja

A tárgy célja ismertetni a jelenleg domináns architektúrákat (Intel Core, AMD Zen, ARM), speciális témaköröket (pl. disszipáció) és a mobil architektúrákat.

### 1.2. Csíkszélesség

A processzor csíkszélessége a hagyományos (planár) MOSFET tranzisztorokat használó processzoroknál a tranzisztor kapu szélességét jelenti. Az Intel által bevezetett 3D FinFET tranzisztorok jellemzéséhez már több paraméterre lenne szükség, de a publikációkban továbbra is egy számot használtak. Ezért gyártónként eltérően kell értelmezni a gyártástechnológiát. Pl. az Intel 10 nm technológiája nagyjából a Samsung vagy a TSMC 7 nm-esének felel meg.

### 1.3. Pentium 4 processzorcsalád

A Core 2 család elődje a Pentium 4 processzorcsalád volt, ami valójában 3 generációt jelent (Willamette, Northwood, Prescott). A Pentium 4 fontos innovációkat tartalmazott, alapja a Netburst mikroarchitektúra. Második generációja hozta be először a többszálás architektúrát, a harmadik pedig a 64 bites architektúrát és a több processzormagot (Pentium D).

2000-ben 7 éves élettartamot vártak a Netburst-től, de 2004 körül a bejelentett architektúrákat visszavonták, mivel a várt 10 GHz-es frekvencia helyett csak 3.6-at értek el, a disszipáció miatt. A Prescott processzorok 1 cm<sup>2</sup>-en 100 W-ot adtak le, amivel elérte a léghűtéses rendszerek határát.

### 1.4. Intel Core processzorok fejlesztése

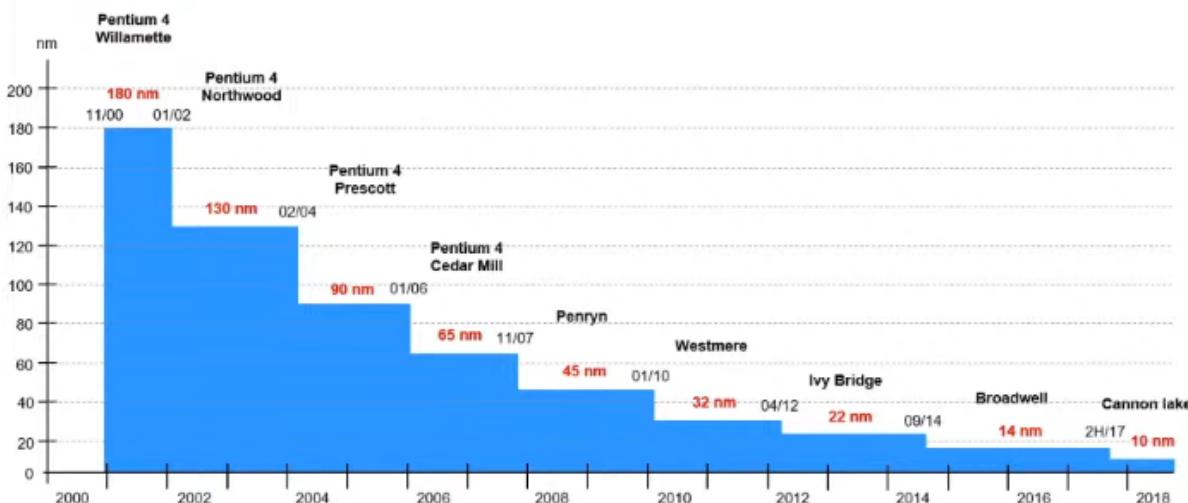
A fejlesztők ezentúl másik irányban folytatták a tervezést, a nyers teljesítmény növelése helyett az 1 W-ból kihozható teljesítmény maximalizálására törekedtek. 2006 környékén a mobil processzorok megjelenésével egy újabb tervezési irány jelent meg: a minél alacsonyabb fogyasztás, a jobb üzemiidő elérése miatt.

A célok elérése érdekében a Pentium 4-ig használt egyfázisú fejlesztési modellt két fázisúra (Tick-Tock)

cserélték. Az első fázisban döntően a csíkszélesség csökkentésére fókusztáltak, a másodikban pedig a mikroarchitektúrát fejlesztették. Ez a fejlesztési modell a Core 2 processzorcsaládtól kezdődött, és a Skylake-ig tartott, amit viszont nem követett csíkszélesség csökkentés, így a fejlesztés visszaállt egy fázisúra.

## 1.5. Gyártási technológia

A gyártástechnológia fejlődése a 1.1. ábrán látható. Látható, hogy 22-14 nm környékén lelassult a fejlődés.



1.1. ábra. Intel processzorok csíkszélessége

Az Intel ennek ellenére 2019-ben két évenkénti csíkszélesség csökkentést jósolt 2029-ig.

## 1.6. Utasításkészlet fejlődése

Az utasításkészlet legfontosabb fejlesztése a vektorfeldolgozás volt, először 64 biten, majd a későbbiekben 128, 256 és újabban 512 bitre szélesítették.

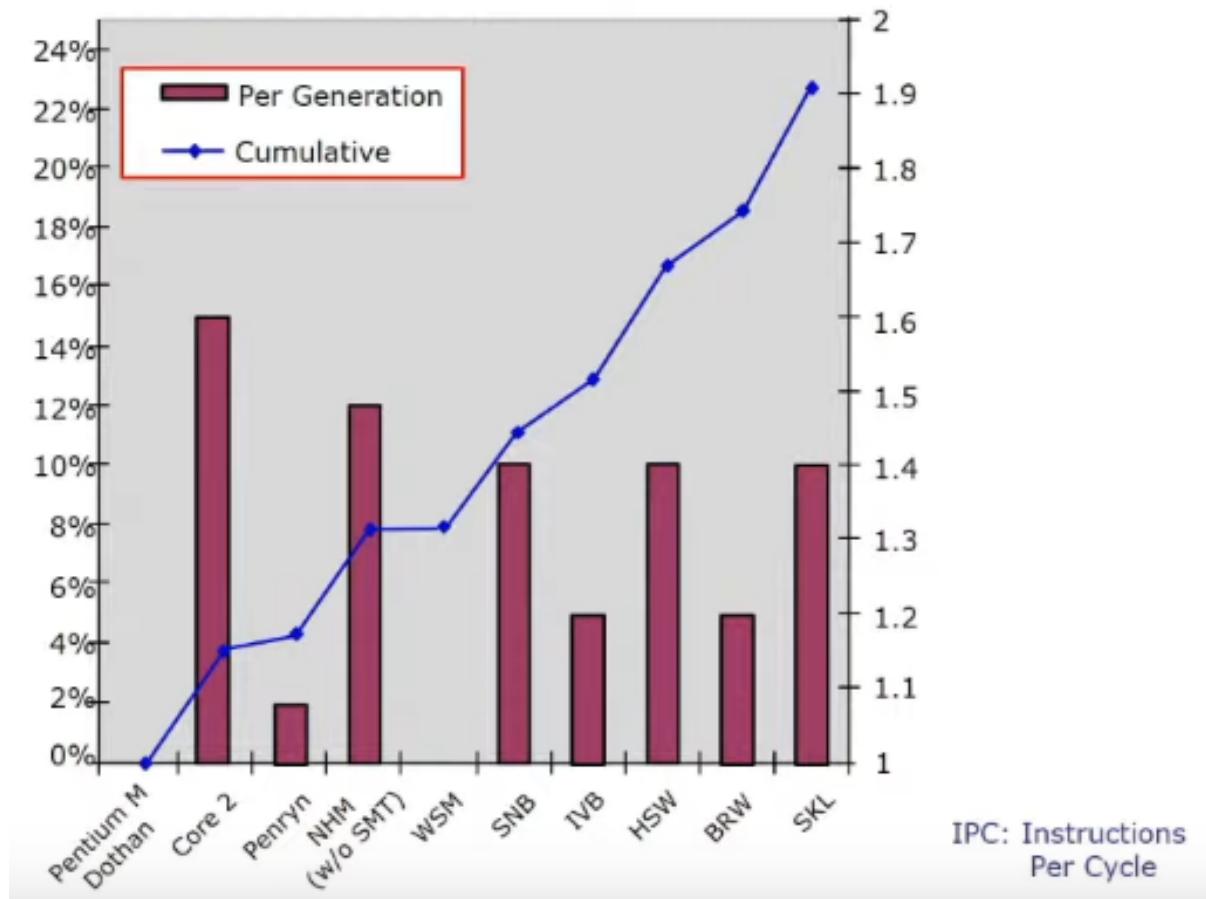
## 1.7. A mikroarchitektúra fontosabb újításai

A Core 2 processzorok 3 helyett 4 széles magokat használtak, kb. 30%-al nagyobb teljesítményt eredményezve. Ezzel lehagyta az AMD-t. A Nehalem architektúránál jelent meg az integrált memóriavezérlő és a 3. szintű cache, valamint újra a többszálú végrehajtás. A Sandy Bridge legfőbb újdonsága a lapkára integrált videokártya volt. A Skylake 5 széles magokat alkalmazott és megjelentek az egy tokba integrált CPU, GPU és nagy teljesítményű memória lapkák. 2017-ben az AMD konkurenciájára válaszul az Intel növelte a magok számát 6-ra, 8-ra, majd 10-re.

Egy másik fontos fejlődési irány a disszipáció kezelése volt, erről később részletesebben.

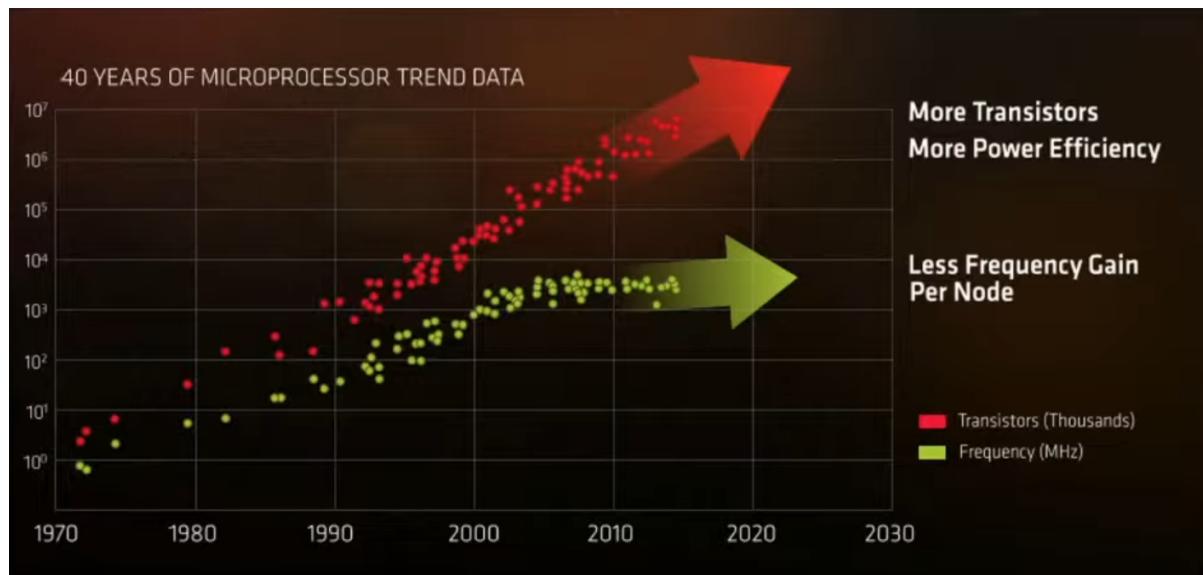
## 1.8. Fejlődés IPC-ben kifejezve

Az ábrán látható, hogy az architekturális váltást hozó processzorok jóval nagyobb (kb. 10%) teljesítmény növekedést hoztak, mint a technológiai váltást hozók.



## 1.9. Frekvencia vs tranzisztorok száma

A következő ábra jól mutatja, hogy az utóbbi kb. 20 évben megállt a frekvencia növekedése, a fejlődés fő iránya a tranzisztorok számának növelése és az energiahatékonyság.



## 2. fejezet

# Intel Core 2 család

### 2.1. Kategorizálás

A Core 2 családot 4 csoportra oszthatjuk:

- szerverek (e3, e5, e7, Platinum, Gold)
- high-end desktopok (i7, i9)
- desktopok, laptopok (i3, i5, i7), összefoglalva kliensek
- mobilok (Atom - 2016-ban visszavonva)

A magok száma kategóriánként változik: szerverek max. 28, HEDT max. 18, desktopok 2-10 + grafika, mobilok max. 10 + grafika. A magok számával a lapkaméret is nő, a szerverek nagyobbak a klienseknél és a mobiloknál.

### 2.2. Kliens processzorok fejlődése

#### 2.2.1. Magok száma

A többmagos fejlődést a Pentium D indította el, de ez még nem valódi kétmagos processzor volt, mivel két, egymagos lapkát tokoztak egybe. Ezután a Core 2 valósított meg 2 magot egy lapkán. A Core 2 Quad 2 db 2 magos lapkát tokozott egybe. A Nehalemmel már egy lapkán 4 mag lehetett.

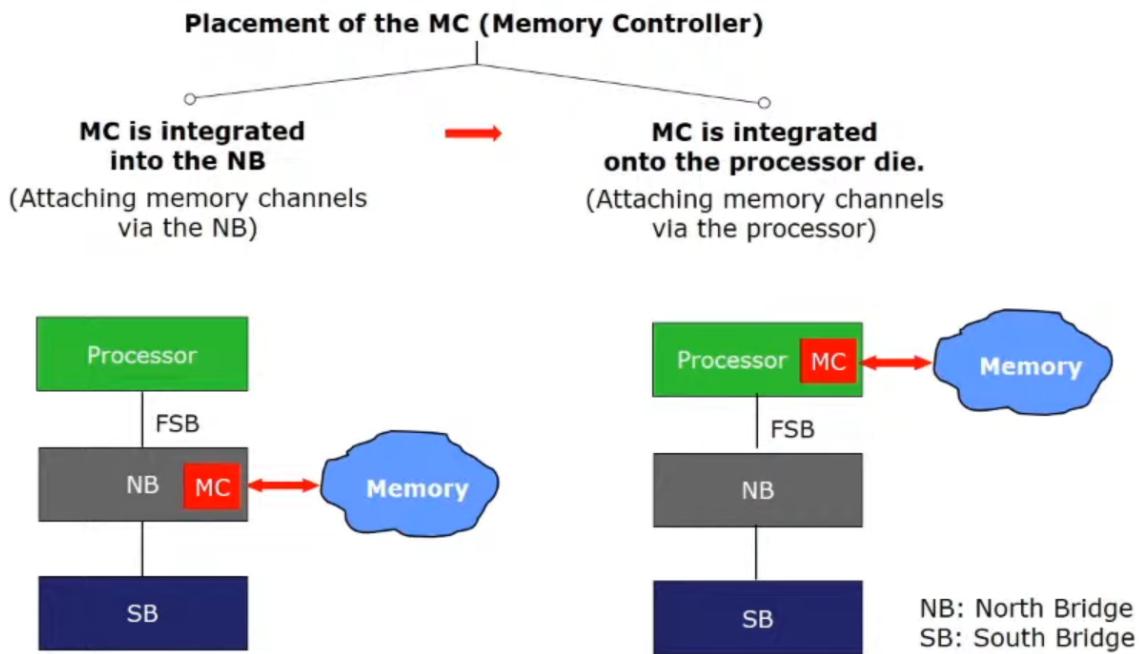
Később megjelent a grafika, egy harmadik, egybetokozott grafikai maggal. Ezután sokáig nem történt semmi, a magok száma maximum 4 maradt. 2017-ben, hogy konkurenciát nyújtsanak az AMD-nek, emelték a magok számát. A 10 nm-es rendszerek megjelenésekor az Intel nem követte a magszám növekedés irányát, először 2, aztán 4 magot alkalmazott.

#### 2.2.2. Memória csatornák

A memória csatorna szám végig 2 maradt. Bár a magok számának növekedésével gyorsabb memória hozzáférésre is szükség volt, a memóriák fejlődése ellensúlyozta azt, hogy nem lett több csatorna.

### 2.2.3. Memória vezérlő elhelyezése

A Nehalem előtt a memóriavezérlőt az északi hídra helyezték. Hátránya, hogy az északi híd és a processzor közötti kapcsolatot tereli a memória forgalma. Ez főleg több processzoros rendszerekben negatívum, mivel az északi híd ekkor komolyan korlátozza a memória sávszélességét. Tehát a processzorszám növekedésével a memória nem skálázódik. Ezért a Nehalem-től a memória közvetlenül a processzorhoz csatlakozik (a lapkán lévő vezérlőhöz), ami rövidebb utakat és tehermentesített északi hidat jelent (2.1. ábra). Előny, hogy processzorunként külön memória kapcsolat létezhet, azaz skálázódik a processzorszámmal.



2.1. ábra. Memória kontrollek elhelyezkedése

Később az északi híd egybeolvadt a délivel, amit periféria vezérlőnek neveztek el.

### 2.2.4. Memória sebessége

A DDR (Double Data Rate) memóriák átviteli sebessége kétszerese az órajelének, tehát a DDR4-2400 (MT/s) átviteli sebességű memória frekvenciája 1200 MHz. Az átviteli sebesség jelentősen fejlődött, 10 év alatt megháromszorozódott.

### 2.2.5. Grafika

A grafikai magok generációjának számozása 5-től kezdődik a Westmere-nél, mivel előtte is volt grafika, csak az északi híd látta el a feladatát. Westmere-nél még csak egybe volt tokozva, de Sandy Bridge-tól már egy lapka.

A grafikai technológiai szint jelölésénél a GR2 egy grafikai szeletet tartalmaz, a GT3 kettőt, a GT4 pedig hármat. A GT1 rész szeletet jelöl (kevesebb végrehajtó egység mint a többiben).

A végrehajtó egységek száma 6-ról 96-ra nőtt az évek során.

A Westmere és a Sandy Bridge még nem támogatott OpenCL-t, de Ivy Bridge már igen.

A teljesítmény növekedése a Sandy Bridge-től Skylake-ig kb 7x-es.

## A Haswell grafikai végrehajtó egysége

Egy szelet 20 végrehajtó egységet tartalmaz, két fél szeletre osztva. A két fél szelet egy közös adat cache-en dolgozik és egy buszrendszeren kapcsolódik. Egy végrehajtó egység (EU) négy funkcionális egységet tartalmaz:

- Send - adat küldés/fogadás
- Branch - elágazáskezelés
- 2x SIMD (Single instruction multiple data) feldolgozó

Minden SIMD egység négy darab egyszeres pontosságú adaton tud MAD (Multiply-Add) műveleteket végrehajtani. Tehát minden EU  $2 \times 4 \times 2 = 16$  utasítást tud ciklusonként elvégezni. Az EU 7 szálon többszálú, minden szálnak 128 db 32 bites regisztere van. Ezen kívül képes lebegőpontos és transzcendens matematikai műveletek végrehajtására.

A Haswell grafikai újdonsága, hogy kapott egy kiegészítő eDRAM-ot is, ezeket az egységeket neveztek Iris Pro-nak.

## 3. fejezet

# HEDT processzorok fejlődése

### 3.1. Bevezetés

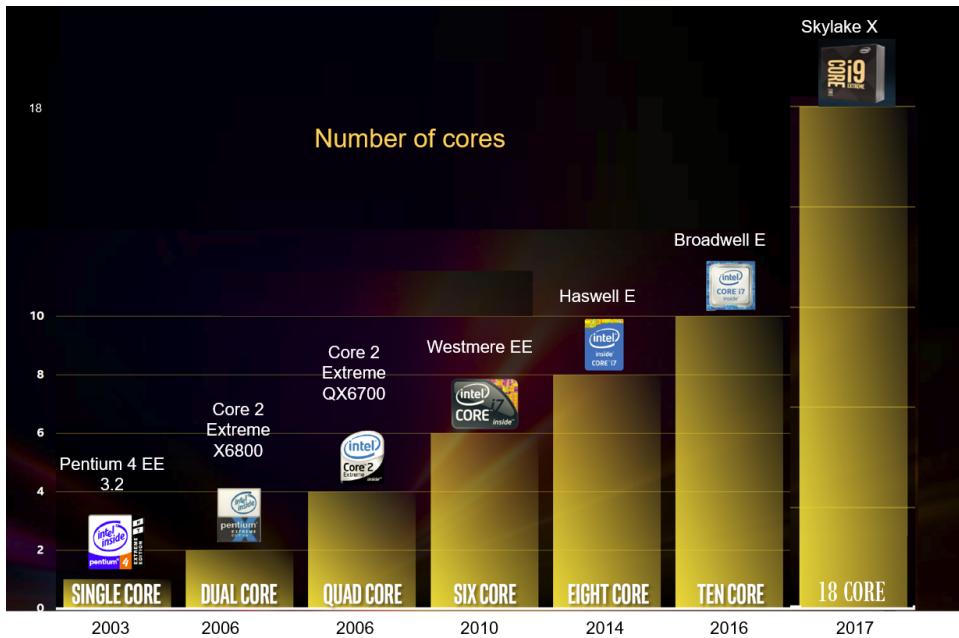
A HEDT (High-End Desktop) processzorok a nagyteljesítményű asztali gépek processzorait jelentik. Ilyenek az Intel i7 és i9 Extreme Edition modellek. Döntően a játékosok és tartalom kreatörök igényeit igyekeznek kiszolgálni. Ezeknek a felhasználóknak nagy teljesítményre és jó minőségű grafikára van szükségük, ezért az ilyen processzoroknak több, akár 4 grafikus kártyát is támogatniuk kell.

### 3.2. PCIe vonalak

Ehhez több PCIe vonalra van szükség (kártyánként 8 vagy 16). A Core 2 Extreme (2006 körül) család (4x8) 32 PCIe vonalpárt tartalmazott, amik az északi hídra csatlakoztak. Ez csak a Sandy Bridge-el változott meg, ahol már a lapkára csatlakoztak. Az Ivy Bridge előtt a PCIe 2.0 változatát támogatták a processzorok, Ivy Bridge-től pedig 3.0-t.

### 3.3. Magszámok

Mivel ezek a feladatok (pixelek feldolgozása) jól párhuzamosíthatók, több mag és így több memória csatorna van a HEDT processzorokban. A magoknak csak a technológiai korlátok szabtak határt. Kezdetben (Core 2 Extreme) ez 2 magot jelentett, majd fokozatosan 18 magra nőtt.



3.1. ábra. Intel HEDT processzorok magszámának fejlődése

### 3.4. Memóriacsatornák

A kezdetekben a viszonylag kevés mag könnyen kiszolgálható volt 2 memóriacsatornával, de ahogy növekedtek a magok számai, úgy nőtt a memóriacsatornák száma is, először 3-ra, majd 4-re.

### 3.5. Overclocking

A processzorok órajelét egy szorzó állítja elő, ami megszorozza a buszfrekvenciát (100/133 MHz) egy olyan faktorral, amit a Power Control Unit ad (pl. 20). A HEDT processzorok szorzója nem rögzített, tehát a hozzáértő felhasználók növelni tudják a processzor sebességét, nagyobb disszipáció árán.

### 3.6. Fogyasztás

A HEDT processzorok nagy fogyasztással rendelkeznek a sok mag és memóriacsatorna miatt (130-160 W). A processzorok teljesítményét TDP-ben szokás megadni (Thermal Design Power). A hűtési rendszert erre az értékre kell tervezni.

### 3.7. Grafika

A HEDT processzorok nem tartalmaznak integrált grafikát, mivel az ilyen PC-kben jellemzően nagy teljesítményű, diszkrét grafikus kártyák dolgoznak.

## 4. fejezet

# Nagy teljesítményű, két foglalatos szerver processzorok fejlődése

### 4.1. Szerver platformok osztályozása

A szerver rendszerek feloszthatók egy processzoros (UP) és több processzoros platformokra. A több processzoros platformok feloszthatók két foglalatos (2S/DP), négy vagy nyolc foglalatos (4S, 8S) és nyolcnál több foglalatos rendszerekre. A piac nagy részét a két processzoros szerverek teszik ki (kb. 80%), ezért részletesebben ezekkel foglalkozunk.

### 4.2. Méret, tranzisztorok

A 2S szerver processzorok nagy méretűek, sok tranzisztort tartalmaznak. A 2000-es évek közepén a Pentium 4 szerver processzorok még viszonylag kevés tranzisztorból álltak (125 millió), ezért elég volt kb 1 cm<sup>2</sup>-es lapka. A magok számának növekedésével egyre több tranzisztor kellett, a Skylake (2017) processzoroknál már 8 milliárd. Még újabb CPU-knál nincs információ. Ezzel együtt a lapkaméret is tovább nőtt, a mai lapkáknál 8 cm<sup>2</sup> körüli.

Mivel a lapkák gyártása nagyon komplex, nagyobb méretű lapkáknál több hiba előfordulhat, ezért alacsonyabb lesz a kihozatali arány. A lapkák méretét így a gazdasági hatékonyság is meghatározza.

### 4.3. Fogyasztás

A HEDT processzoroknál is nagyobb fogyasztással kell számolni a szerver processzoroknál, 130-tól 200 W felettig mennek.

### 4.4. Lábak száma

A szerver processzorok sok lábbal kapcsolódnak a foglalatokhoz, amik precíz gyártást igényelnek.

## 4.5. Magok száma

A szerver alkalmazások általában jól párhuzamosíthatók, így a processzorok a lehető legtöbb magot implementálják. A Pentium 4 szerver processzoránál még csak 1 mag volt, de később 28-ig növekedett a magok száma.

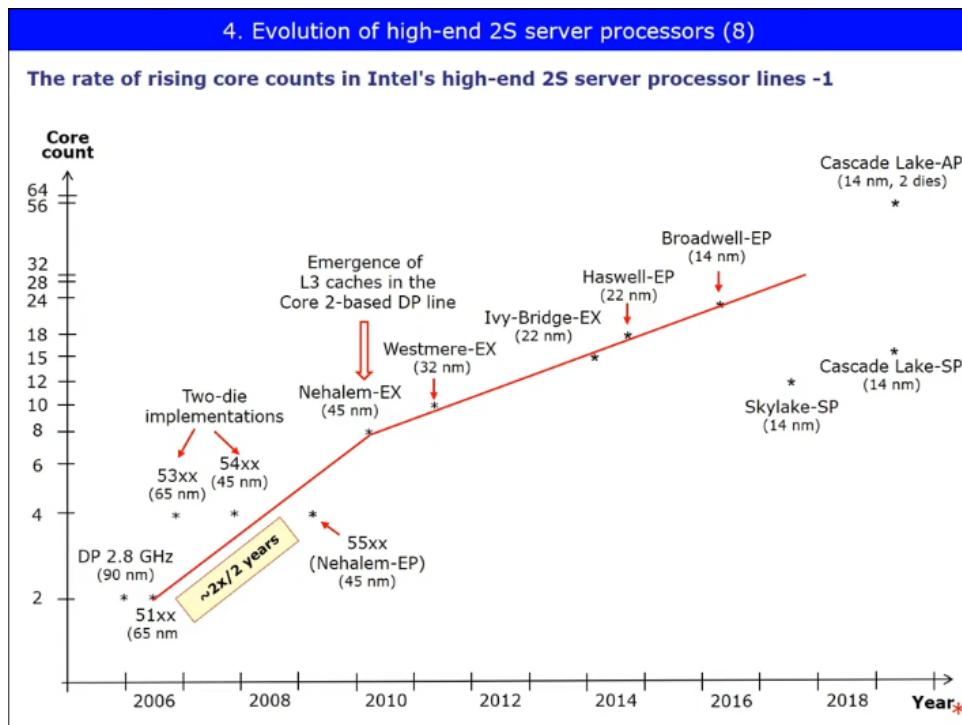
## 4.6. Intel Cascade Lake 9200-AP

Ez a 2x28, azaz 56 magos processzor az Intel válasza volt az AMD 64 magos EPYC Rome processzoraira. Az AMD 2017-ben adta ki ezt a processzort, ami valós konkurenciát állított az Intelnek, ami 2015 környékén még nagyrészt egyeduralkodó volt a szervereknél. Ezért az Intel két 28 magos lapkát tett egy foglalatba, és BGA (Ball Grid Array) tokozással látta el, ami forrasztásra készült. A forrasztás miatt nem voltak külön megvásárolhatók, csak OEM-ek kapták meg.

A két lapka egybecsomagolása miatt egy két foglalatos rendszer gyakorlatilag 4 processzort jelent.

## 4.7. Magszámok és tranzisztorok növekedése

A kétfoglalatos szerver CPU-k esetében kb 2 évente duplázdódt a magok száma (4.1. ábra).

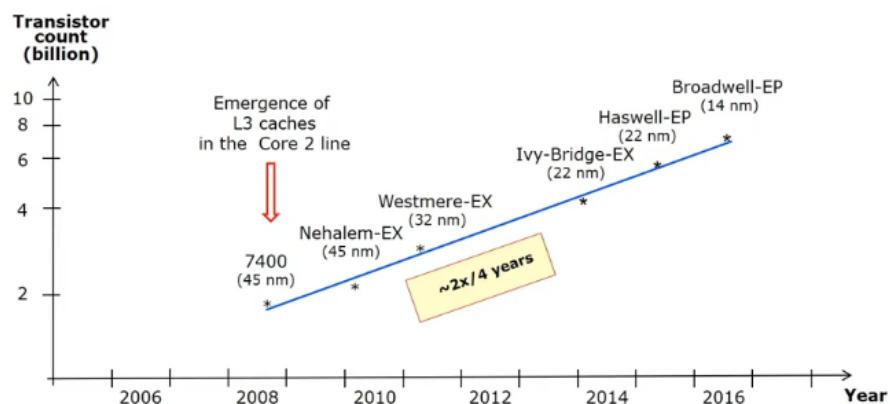


4.1. ábra. Intel 2S szerver processzorok magszámának fejlődése

Ez összhangban van a Moore-szabálytal, ami szerint a tranzisztorok száma két évente duplázdik. Az utóbbi években viszont lelassult a fejlődés, már csak négy évente duplázdott a magok száma. Ennek egyik oka az L3 cache megjelenése, ami elfoglalja a lapka jelentős részét, mivel nagyon sok tranzisztor kell a működéshez. A technológi változása miatt a Moore törvénnyel ellentében az Intel már csak 4 évente tudta dupláznai a tranzisztorok számát (4.2. ábra). Ennek az oka, hogy a sok tranzisztor disszipációját kezelni kell, ami korlátozza a fejlődést.

#### 4. Evolution of high-end 2S server processors (10)

Evolution of transistor counts of Intel's Core 2 based HE server processor dies



As the above Figure shows, transistor counts rise slower than would be according to Moore's law. This can presumably be attributed to heat issues caused by the limited power budget.

\*

4.2. ábra. Intel 2S szerver processzorok tranzisztorszámának fejlődése

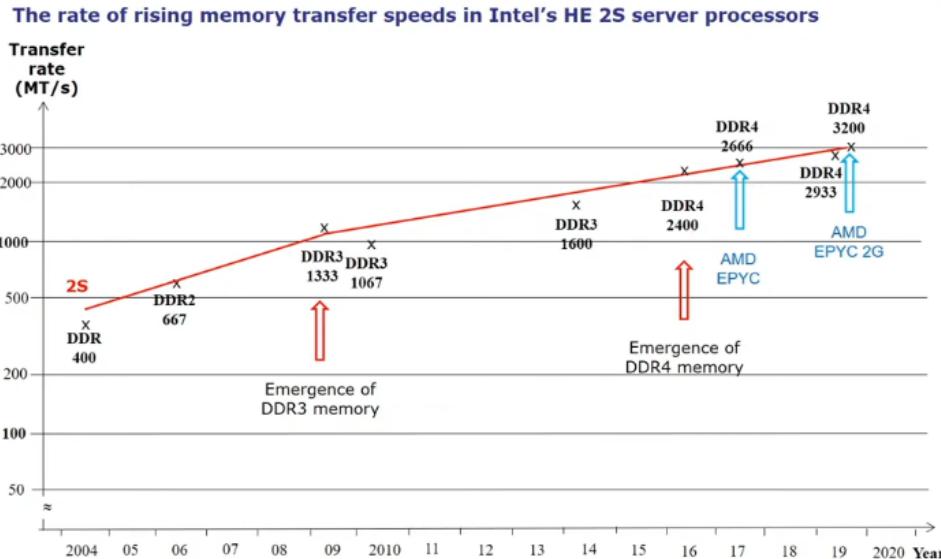
#### 4.7.1. Kivételek

Néhány processzor az ábrán nem illik a fejlődési görbüre, ezek az AMD konkurenciára adott válaszoknak köszönhetők.

#### 4.8. Memória sebességek fejlődése

A memória sebességeknél adatátviteli rátával jellemzők őket, mivel a frekvencia nem azonos az átviteli sebességgel a DDR (Double Data Rate) memóriák esetében.

#### 4. Evolution of high-end 2S server processors (11)



\*

4.3. ábra. Memória sebességek fejlődése

A ???. ábrán látható, hogy a DDR és DDR2-es memóriák gyorsabban fejlődtek, mint a DDR3 és DDR4 memóriák. Az első szakaszban egy duplázódáshoz 4 év kellett, DDR3 óta pedig már 8. Ennek oka a DDR3-as memóriák jóval komplexebb felépítése, ami lassította a fejlődést. A magok száma tehát gyorsabban növekedett, mint a memóriák sebessége, ami konfliktusforrás.

Szerverek esetében elvárás, hogy az egymástól függetlenül működő magokat azonos memória sávszélességgel kell ellátni. Tehát a sávszélességet lineárisan skálázni kell a magszámmal együtt. A foglalatonkénti sávszélesség kiszámítása:

$$BW/core = n_M * w * T_M/n_C \quad (4.1)$$

ahol:

- $n_M$ : memória csatornák száma
- $w$ : memória csatorna szélessége (pl. 8 byte)
- $T_M$ : memória átviteli rátája (pl. 2.4 GT/s)
- $n_C$ : magok száma

Probléma, hogy a transzfer ráta lassabban nő, mint a magok száma, tehát  $T_m/n_C$  folyamatosan csökken. Megoldás a memória csatornák számának növelése. Ez azonban elég komplex feladat.

A magonkénti memória sávszélességnél referenciának tekinthető a Pentium 4 DP rendszer, a fejlesztési cél ehhez az értékhez tartani.

## 5. fejezet

# Mobil és táblagép processzorok fejlődése

### 5.1. Bevezetés

Az okostelefonok 2006 körül jelentek meg, a táblagépek ennél is később, 2010-ben. A fő prioritás az üzemidő, tehát alacsony fogyasztású processzorokat kell fejleszteni. Ez ellentétes az addig uralkodó fejlesztési iránytól (legnagyobb teljesítmény wattoinként).

### 5.2. Fogyasztás csökkentése

Az alacsony fogyasztást kétféleképpen lehet elérni:

- keskeny mikroarchitektúrákkal
- alacsony órafrekvenciával.

Az ARM Cortex család korai tagjai ezt általában 2 (3 a Cortex A15 esetében) széles architektúrával (egyszerre 2 utasítás vágrehajtására képes) érték el.

#### 5.2.1. Intel Atom

Az Intel viszont a Core 2 családtól kezdve 4 szélességű architektúrát implementált, a nagyobb teljesítmény/fogyasztás arány elérése érdekében. Következmény, hogy az Intel és AMD hagyományos mikroarchitektúrái nem voltak versenyképesek az ARM-el szemben. Ezért kifejlesztettek egy új mikroarchitektúrát, ami 2 szélességű. Így jött létre 2008-ban az Intel Atom családja, amire alapozva megjelentek az Intel okostelefon platformjai.

#### 5.2.2. AMD

2011 környékén az AMD a Bulldozer mikroarchitektúrára épülő processzorokat gyártotta, ami végül nem vált be. Ezzel párhuzamosan bevezették a kisebb fogyasztású Cat mikroarchitektúrát, az Atomhoz hasonlóan 2 szélességgel. Ez is a mobil piacot célozta.

### 5.2.3. A frekvencia csökkentése

A disszipáció két részből áll: statikus és dinamikus. Statikus a zárt tranzisztorokban szivárgó áram által kellett disszipáció, dinamikus pedig a működő tranzisztorok által kellett. Következmény, hogy a dinamikus disszipáció egyenesen arányos az órafrekvenciával, mivel magasabb frekvencián több töltéskisülés keletkezik. Ezen kívül négyzetesen arányos a magfeszültséggel (Ohm-törvény). Tehát a dinamikus disszipáció:

$$D_d = \text{const} * f_c * V^2 \quad (5.1)$$

Mivel a frekvenciához arányosan nagyobb feszültség kell, ezért a disszipáció a frekvencia növelésével köbönként növekszik. Ezt mutatja a 5.1. ábra.

TDP (W)	No. of cores	Graphics	No. of graphics EU	eDRAM	Base frequency (GHz)
4.5	2	HD 515	18	--	1.2
15	2	HD 540	48	64 MB	2.2
15	2	HD 520	24	--	2.6
28	2	HD 550	48	64 MB	3.3
35	4	HD 530	24	--	2.8
45	4	HD 530	24	--	2.9
65	4	HD 530	24	--	3.4
91	4	--	--	--	4.2

5.1. ábra. Az Intel Skylake család órajelei és fogyasztásai

A Skylake család gyengébb tagjait táblagépekbe szánták, ezeknél az alacsony fogyasztás érdekében gyengebb grafikát, kevesebb magot és alacsony órajelet használtak.

### 5.3. Piaci részesedés

A fejlesztések ellenére az Intelnek és AMD-nek nem sikerült betörniük a mobil eszközök piacára.

Smartphone application processors worldwide market share 2015 (revenue)	
Qualcomm (USA)	42 %
Apple (USA)	21 %
MediaTek (Taiwan)	19 %
Samsung (S. Korea)	
Spreadtrum (China)	

5.2. ábra. Az okostelefon processzorok piaci részesedése 2015-ben

Bár az Intel a táblagépeknél elért valamekkora részesedést, ezt úgy érte el, hogy fizetett a gyártóknak az Atom processzor használatáért.

Tablet application processors worldwide market share 2015 (revenue)	
Apple (USA)	31 %
Qualcomm (USA)	16 %
Intel (USA)	14 %
MediaTek (Taiwan)	
Samsung (S. Korea)	

5.3. ábra. A táblagép processzorok piaci részesedése 2015-ben

A gyártóknak fizetett összegek miatt az Intelnek 2 év alatt 7 milliárd dollár vesztesége keletkezett. A nagy veszteségek miatt 2016-ban az Intel visszavonult a mobil piacról, a már bejelentett rendszereket pedig törölték. Ezzel együtt 12000 embert bocsátottak el.

Az AMD 2015-ban még megjelentetett egy új mobil családot, de a 2016-os kínálatban már nem szerepeltek mobil processzorok. Helyette 2017-től a Zen architektúrát fejlesztették.

Az Nvidia szintén gyártott mobil processzorokat a 2010-es évek elején, de 2016-ban ők is kivonultak a

piacról.

## 5.4. Szélesség fejlődése

A keskeny szélesség és alacsony frekvencia követelmények a processzorok fejlődésével kevésbé lett k lényegesek, mivel nagyon sok fejlesztés történt a disszipáció csökkentésére. 2010 környékén még jellemzően 2 széles rendszerek voltak a mobilokban, aztán a teljesítmény növelése érdekében 3 szélesre növelte a processzorait az ARM, az Apple, a Qualcomm és a Samsung is. A fejlődés itt nem állt meg, az ARM 2018-ban 4 széles rendszereket hozott ki. Szintén 4 széles a Samsung M1 magja is. Ezután még 6, 7 és 8 széles architektúrákat is bejelentettek. Az Apple pl. nagyon korán (2013-ban) 6 széles magot mutatott be (Jim Keller fejlesztése).

## 6. fejezet

# Disszipációkezelés

### 6.1. A disszipációkezelés fontossága

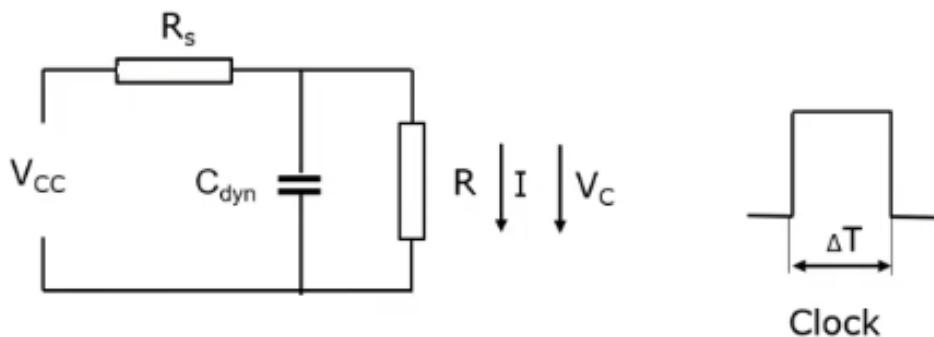
A processzorok fejlesztésének két iránya van: teljesítmény növelése és a fogyasztás csökkentése. A fejlesztés fókusza folyamatosan a disszipáció csökkentése felé tolódott, mivel rájöttek, hogy a teljesítmény fő korlátja a disszipáció.

### 6.2. A disszipáció

A disszipáció két komponensből áll: dinamikus (működés közbeni) és statikus. A tápegység szempontjából a processzor egy szort kapacitás, amit a felfutó órajelen fel kell tölteni, a lefutón pedig egy ellenálláson keresztül le kell vezetni. A töltés-kisülés fogyasztása a dinamikus disszipáció. A statikus disszipáció a lezárt tranzisztorok szivárgási áramából adódik. A teljes disszipáció a kettő összege.

#### 6.2.1. Kiszámítása

A dinamikus disszipáció kiszámításához a processzor aktív tranzisztorait egy áramkörrel modellezhetjük (6.1.).



6.1. ábra. Aktív tranzisztorok modellje

Ebből a dinamikus disszipáció kiszámítása:

$$D_d = 2 * C_{dyn} * V_{cc}^2 * f_c \quad (6.1)$$

Az órafrekvenciával lineáris, mivel a töltés-kisülések mennyisége függ tőle.

### 6.3. TDP

Thermal Design Power, azaz tervezési hőérték. Egy számításigényes alkalmazás futtatása közbeni maximális disszipációt jelenti. Fontos, hogy nem teljesen pontos érték. Az OEM-ek ez alapján tervezik a hűtés rendszereket, aminek legalább ezt a TDP értéket el kell vezetnie úgy, hogy a tranzisztorok hőmérséklete nem megy kb. 80-90°C fölé. A kategóriák jellemző TDP értékeit mutatja a 6.2. ábra.

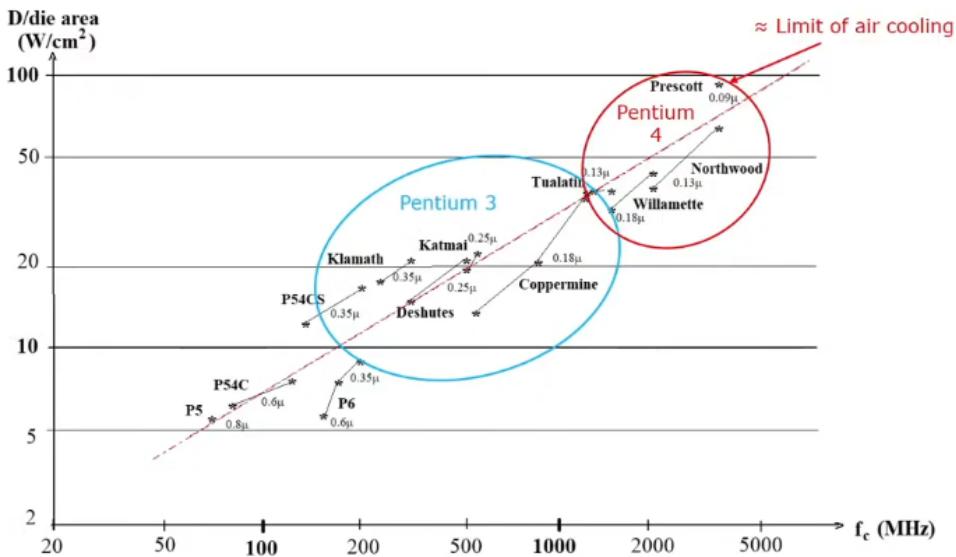
Processor category	TDP	Intel's usual tags
<b>Servers</b>	≈85-200 W	
<b>HEDs</b>	≈100-150 W	X
<b>Desktops</b>	<b>High perf.</b>	≈70-95 W K
	<b>Mainstream</b>	≈50-65 W S
	<b>Low power</b>	≈35-45W T
	<b>High perf.</b>	≈45 W H/HQ
<b>Notebooks</b>	<b>Mainstream</b>	≈25-35 W U
	<b>Ultra-thin</b>	≈15 W U
<b>Tablets</b>	≈5 W	Y/m

6.2. ábra. Processzor osztályok jellemző disszipációja

A TDP korlátozza az alapfrekvenciát. A Turbo működéséhez jóval több watt szükséges.

### 6.4. Intel processzorok disszipációja a fejlődés során

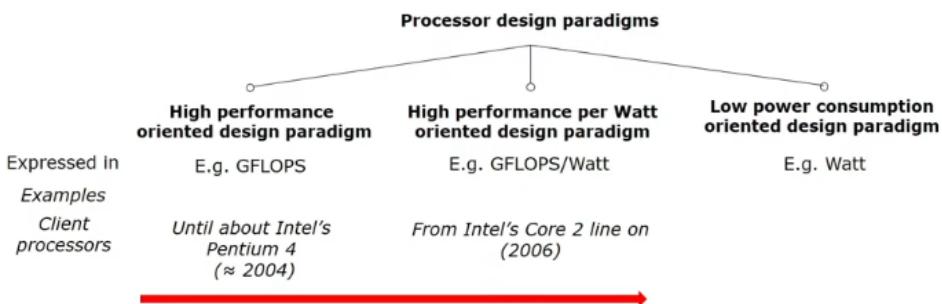
A 6.3. ábrán az Intel processzorok lapkamérethez viszonyított disszipációjának növekedése látható az órajel növekedésével.



6.3. ábra. Disszipáció változása az órajel függvényében

Látható, hogy a Pentium 4 Prescott mag már közel 100 wattot adott le  $\text{cm}^2$ -enként, ami a léghűtés fizikai határa. Az Intel 2000-ben azt várt a Pentium 4 családtól, hogy 10 évig gyártásban marad és 10 GHz órajelet fog elérni. Ehelyett 2004-ben 4 GHz-nél megakadt a fejlesztés. A Prescott család bizonyos tagjait kényetlenek voltak visszavonni túlmelegedési problémák miatt.

Ezzel az Intel 2003-tól a nyers teljesítményről áthelyezte a hangsúlyt a teljesítmény per watt, azaz a hatékonyság növelésére. Összefoglalva a tervezési paradigmák változását mutatja a 6.4. ábra.



6.4. ábra. Processzor tervezési paradigmák

### 6.5. Disszipációcsökkentő kezdeményezések

A disszipáció kezelésére megszületett az Energy Star kezdeményezés, az AMD pedig célul tűzte ki, hogy 2014 és 2020 között 20x-osára emeli a hatékonyságot.

### 6.5.1. Energy Star

Az amerikai környezetvédelmi hatóság indította 1992-ben, az energiatudatosság elterjesztésére a processzoroknál. Több verziója volt, a legfontosabb az 5., amit 2009-ben adtak ki, a Pentium 4 problémái

után. Lényege, hogy a processzorokra kiszámítják az éves inaktív állapotú energiafogyasztást, és ha egy modell a határérték alatt van, akkor használhatja az Energy Star jelölést. A különböző inaktív állapotok súlyozása kategóriánként eltér.

### 6.5.2. Az AMD hatékonyiségi kezdeményezése

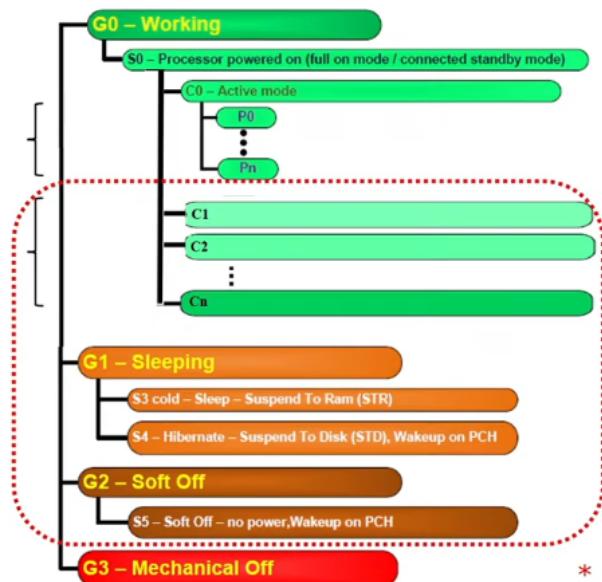
2014-ben jelentették be, hogy növelni szeretnék a teljesítmény per watt arányt. 2009-től 2014-ig 10x-es hatékonyág növekedést értek el, ehhez képest szerettek volna további 25x-ös javulást 2020-ra. 2020-ban bejelentették, hogy a Zen 2 alapú Ryzen processzoraikkal elértek a célt, 32x-es növekedéssel.

### 6.5.3. Szerverek energiahatékonyisége

A szervereknél kiemelten fontos a hatékonyág, mivel egy adatközpont több MW áramot is fogyaszthat. A szuperszámítógépeket hatékonyág szerint is rangsorolják, erre szolgál a Green500-as lista.

## 6.6. ACPI

Az inaktív állapot definiálása az Energy Star-hoz az ACPI (Advanced Configuration and Power Interface) szabvány segítségével lehetséges. Ez egy nyílt szabvány, ami definiálja a számítógép állapotait. Az állapotot globális (Gi), teljesítmény (Pi), inaktív (Ci) és alvó (G3) állapotok jellemzik (6.5. ábra).



6.5. ábra. ACPI állapotok

Globális állapotok:

- G1: OS által kezdeményezett leállás, kontextus mentéssel
- G2: OS által kezdeményezett leállás, kontextus mentés nélkül, tápellátás megmarad
- G3: mechanikai kikapcsolás

## 6.7. A disszipációkezelés technikái

A disszipáció kezelése sokféleképpen történhet, a tranzisztorok tervezésétől az áramkörökön és a processzoron át egészen a platform és operációs rendszer szintekig.

### 6.7.1. Kezelés áramköri szinten

Két jelentős technológia:

- óra kapuzás
- tápfeszültség kapuzás

#### Óra kapuzás

Az éppen inaktív áramkörök elé egy AND kapu kerül, ami inaktivitás esetén nem küld órajelet az áramkörre, így csökkentve a fogyasztást. A statikus disszipáció viszont továbbra is fennmarad. Implementációja történhet finom vagy durva szemcsézettséggel. A finomnál kis áramköri egységeket külön-külön kapuznak, míg a durva szemcsézs több modult kapoz egyszerre. Az óra kapuzást korán, már 1996-ban is alkalmazta a DEC cég az Alpha processzoraiban (lebegőpontos egységeket kapuzták). Mivel a technológia jól bevált, hamar elterjedt, az Intel a Pentium 4-től használja, ma már teljesen általános.

#### Tápellátás kapuzása

Az óra kapuzás továbbfejlesztése, nagyobb áramköri egységekre használják mint pl. mag vagy memória kontroller. Az egység inaktivitása esetén a komplett tápellátást lekapcsolja az egységről, így a dinamikus és a statikus disszipációt is megszüntetve. A többmagos processzorokban a processzormagok külön-külön kapuzhatók.

A tápfeszültség kapuzás előfeltétele a Turbo Boost technológiának, mivel a Turbo órajel a processzor fogyasztása és a TDP közötti hőtartalékot felhasználva emeli meg az órajelet. Ehhez elegendő hőtartalékot kell biztosítani, amit nagyban elősegít a kapuzás.

A tápellátás kapuzást az Intel vezette be a Nehalem processzorokkal 2008-ban. Később az AMD és az IBM is átvette.

Implementáció szerint először (Nehalem) csak a magokat kapuzták, később több egységre is kiterjedt mint pl. L3 cache, GPU, memória kontroller. Az Intel a Westmere, Sandy Bridge és Ivy Bridge folyamatosan terjesztette ki a kapuzást, az AMD a Bulldozer családdal gyakorlatilag minden alapegységet kapuzott.

A tápellátás megszüntetése történhet a pozitív pólus vagy a föld elvágásával is.

#### Integrált tápegeységek - Intel FIVR

Az Intel a Haswell családdal továbbfejlesztette az energiaellátást, a tápegeység (voltage regulator) bizonyos részeinek processzorlapkára helyezésével. Ez a FIVR (Fully Integrated Voltage Regulator), ami eltérő feszültséggel tudja ellátni a processzor különböző részeit, így a kapuzás fölöslegessé vált.

Ezt a technológiát a Haswell és Broadwell processzorok után felfüggesztették, mivel az integrált tápegeység a processzor disszipációjához adott hozzá, így csökkentve az elérhető órajelet. A Skylake-el kezdődően tehát ideiglenesen visszatértek a teljesítmény kapuzásra.

## Feszültség lekapcsolás előkészítése

Mielőtt egy magról lekapcsolnánk a tápot, a gyorsítótárat vissza kell írni a következő szintű gyorsítótárba, vagy ha az nem elérhető, memóriába. A processzor végrehajtási állapotát (kontextus, azaz a regiszterek állapota) is el kell menteni egy következő szintű cache-be, vagy a memóriába, esetleg SRAM-ba. A táp lekapcsolással együtt a magról lekapcsolják az órajelet és az órajel generátort is.

### A kontextus mentése

A kontextus mentése történhet:

- memóriába (AMD használta pl. Bulldozer, Piledriver, Excavator, Bobcat, Jaguar)
- következő szintű gyorsítótárba (AMD Jaguar, Puma)
- magonkénti SRAM-ba (Intel processzorok)

### 6.7.2. Kezelés platform szinten

Platform szinten bonyolult a disszipáció kezelés, mivel részt vesz benne a processzor, az IO eszközök, a BIOS és az OS is. Az első ilyen megoldások az Intel-től származnak (SL), később a Microsofttal összefogva fejlesztettek (APM, nem terjedt el). Később több gyártó is bekapsolódott, így alakult ki a nyitott ACPI szabvány (1996-tól máig használják).

### Az ACPI fejlődése

Az ACPI 1.0 verziót a Windows 95, 98, 2000 és NT 4.0 rendszerek támogatták. Az XP és a Server 2003 rendszerek már az ACPI 2.0-t használták, amiben már jelen volt a DVFS (Dynamic Voltage and Frequency Scaling) és nagyban elősegítette a disszipáció kezelését. A többmagos és többszálas rendszerek támogatása az ACPI 3.0-ban jelent meg, ezt a Vista, 7 és Server 2008 operációs rendszerek használták. A 4.0 kevésbé jelentős, de az 5.0 egy új szemléletet vezetett be: Collaborative Processor Performance Control, aminek során az operációs rendszer csak átadja a processzornak, hogy a felhasználó a teljesítményt vagy az alacsony fogyasztást preferálja, majd ezután a többi döntést a processzor hozza meg hardveresen. A fejlődés folytatódott, de a 6-os verziónak nem voltak jelentős újításai.

### 6.7.3. Kezelés CPU szinten

Kétféle kezelési lehetőség:

- aktív magok kezelési technikái:
  - energiaszükséglet csökkentése:
    - \* DVFS - dinamikus feszültség és frekvencia kezelés
    - \* CPPC
    - \* AVFS - adaptív feszültség és frekvencia kezelése
  - Turbo Boost: a TDP hőtartalékát felhasználva megemeli a processzor az alap frekvenciáját ideiglenesen
- inaktív magok kezelési technikái (ACPI alapú, C állapotban keresztül): a kihasználatlan magot egyre mélyebb alvó állapotba helyezi a rendszer.

## DVFS - Dynamic Voltage and Frequency Scaling

### Egymagos rendszerek

A DVFS (egymagos, egyszálas processzoroknál) alapelvei:

- a dinamikus fogyasztás kiszámítása:

$$D_d = \text{const} * V_{cc}^2 * f_c$$

- a szükséges magfeszültség ( $V_{cc}$ ) ahhoz, hogy stabil órajelet lehessen generálni közel egyenesen arányos az órafrekvenciával

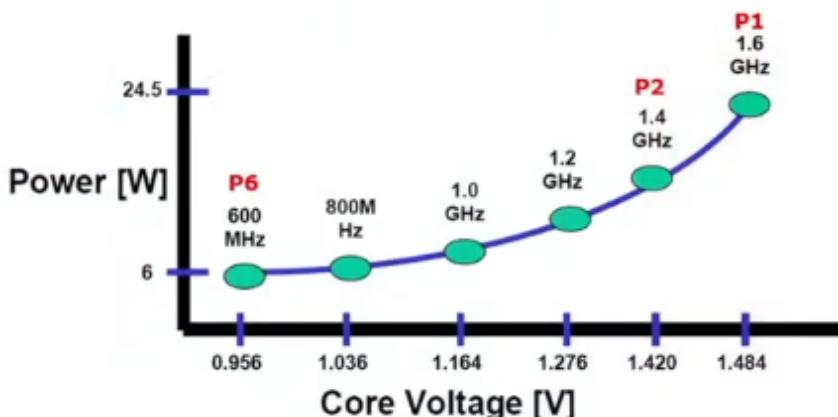
$$V_{cc} = \text{const} * f_c$$

- tehát a dinamikus disszipáció közel köbösen aránylik a frekvenciához:

$$D_d = \text{const} * f_c^3$$

Következmény, hogy a disszipáció csökkentésére nagyon hatékony eszköz a frekvencia csökkentése. Fontos, hogy a frekvencia csak addig csökkenthető, amíg ez nem rontja a felhasználói élményt.

A DVFS a P (performance) állapotokon alapul, amiket az ACPI 2. verziója tartalmazott. A P állapotok munkapontok, amik a magfrekvencia és órafrekvencia kettősöket adják meg (6.6. ábra).



6.6. ábra. Az Intel Pentium M processzor munkapontjai

Látható, hogy a legalacsonyabb állapotban csak 6 watt a disszipáció, a legmagasabbban pedig már 24.5 watt. A cél tehát a lehető legalacsonyabb fogyasztású P állapotban tartani a processzort, ha ez megtehető a felhasználói élmény rontása nélkül. A legmagasabb teljesítményű és fogyasztású állapot a P1.

Itt nagyon fontos szerepet kap az OS, mivel neki kell eldöntenie, hogy melyik az a P állapot, ami még nem rontja érezhetően a teljesítményt. Ehhez az OS vizsgálja a CPU kihasználtságát: megnézi, hogy egy adott időablakot az ütemező hány százalékban használ ki. Pl.: ha a CPU mag 2 GHz-en működik, de a kihasználtság csak 50%, akkor minden második óraciklus elvesztegetett. Ezért a frekvencia lehívhető 1 GHz-re, érezhető lassulás nélkül. Ez alapján az OS kiválasztja a P állapotok közül az optimális munkapontot. Ha megtörtént a kiválasztás, az OS a CPU-val egy regiszter interfésszen keresztül közli a döntést. Ezt az interfészét az ACPI standard határozza meg. Ezután ezt a regisztert lekérdezi a PCU (Power Control Unit) és beállítja a frekvenciát. Ez a rendszer igényvezérelt skálázást tesz lehetővé.

## Többmagos rendszerek

Többmagos rendszereknél az OS a taskokat szálakhoz, és nem magokhoz rendeli. Ezért a kihasználtságot is szálakra vizsgálja. A DVFS lépései több mag és több szál esetén:

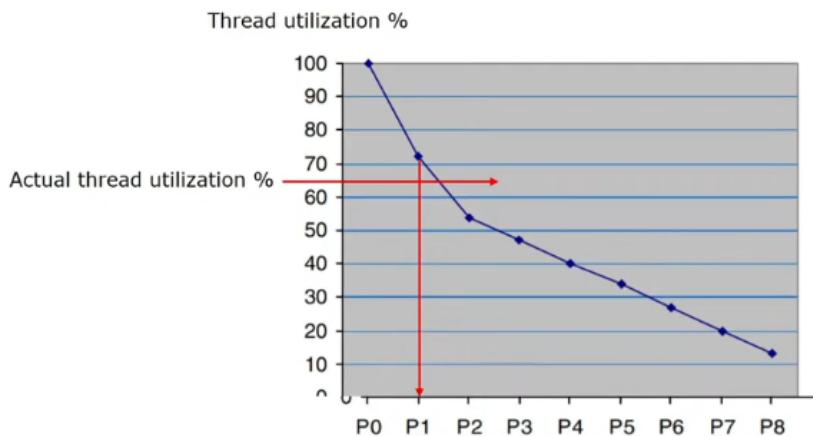
- szál kihasználtság meghatározása
- kihasználtság alapján legjobb P állapot kiválasztása
- ha több szál van egy magon, koordinálni kell a kiválasztott P állapotokat (egy mag csak egy feszültségen működhet)
- az igényelt P állapot kommunikálása a processzorral
- a PCU beállítja az igényelt magfeszültséget és órajelet

## Szál kihasználtság meghatározása

Először azt figyelték, hogy az idő hány százalékában nincs ütemezendő szál, viszont ez körülményes volt. Ezért 2006-tól az Intel bevezetett szálanként két gépi 64 bites regisztert, amiket számlálóként használ. A számlálókat csak akkor lépteti a processzor, ha az adott szálhoz tartozó logikai processzor aktív (C0) állapotban van. Az egyik számlálót az alapfrekvencia ütemében inkrementálják, a másikat pedig a pillanatnyi órajel ütemében. A kettő aránya megadja a processzor kihasználtságát.

## P állapot meghatározása

A kihasználtság alapján meghatározható a legmegfelelőbb P állapot (6.7. ábra).



6.7. ábra. P állapot meghatározása

## Több szál koordinálása egy magon

Ha egy mag több szálat futtat, minden a teljesítmény igényesebb szálhoz kell igazítani a mag állapotát. Ha több mag van egy közös órajelről táplálva, szintén a legnagyobb teljesítmény igényű szálat futtató maghoz kell igazodni. Az újabb rendszerek már képesek magonként különböző P állapotokat kezelní, így a magok közötti koordináció nem szükséges.

## Kiválasztott P állapot kommunikációja

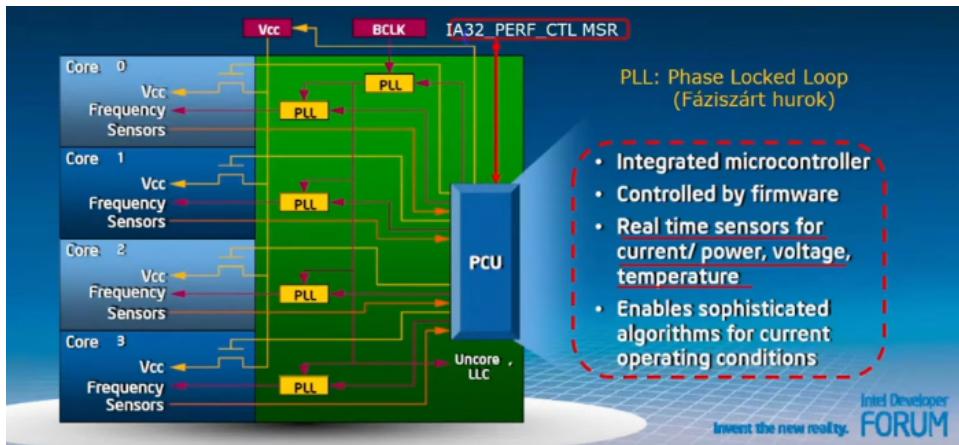
Erre a performance control (IA32\_PERF\_CTL) belső regiszterek szolgálnak, amibe az OS írhat. A regiszterben az alsó kettő byte szolgál a feszültség és a frekvencia megadására. A frekvencia megadása szorzófaktorral történik, pl. 133 MHz busz frekvenciánál 34-es szorzó 34x133 MHz-t jelent. Ennek a regiszternek a kódolása nem publikus.

## PCU működése

A performance control regiszter kiolvasása után a PCU beállítja a szükséges paramétereket.

### DVFS az Intel Nehalem processzorokon

A Nehalem mikroarchitektúrára épülő CPU-knál vezették be először a PCU használatát. Működése a 6.8. ábrán látható.



6.8. ábra. PCU az Intel Nehalem processzorokban

### A DVFS továbbfejlesztése

A DVFS-nek két továbbfejlesztett változata van:

- CPPC: az OS csak iránymutatással látja el a PCU-t, a precíz feszültség és órajel szabályzást a PCU végzi.
- AVFS: a DVFS biztonsági zónáját csökkenti, a teljesítmény növelése vagy a disszipáció csökkentése érdekében.

A CPPC-t az Intel a 2015-ös Skylake óta használja (SpeedShift), az AMD pedig a 2019-es Zen 2 alapú Ryzen 3000 óta. Az AVFS egy régebbi rendszer, az IBM 2007 óta használja, a Samsung 2015-től (Galaxy S6), az AMD pedig a régebbi processzoraiban.

### CPPC - Collaborative Processor Performance Control

A CPPC-t az ACPI 5.0 szabvány definiálja, az ACPI 2.0 alapú DVFS-t váltja le ez a technológia. Ehhez szintén szükséges OS támogatás, ami a Windows 10 2015 őszi frissítésével jelent meg.

A DVFS-nél az OS nézi a kihasználtságot és küldi el a szükséges P állapot paramétereit. Hátránya, hogy a kihasználtság vizsgálata nagy időablakban történik, így a rendszer lassú, csak ritkán tudja módosítani a P állapotokat. A gyorsításához a lehető legjobban ki kell küszöbölni az OS-t, erre szolgál a CPPT. Ez a gyorsulás látszik a 6.9. ábrán.

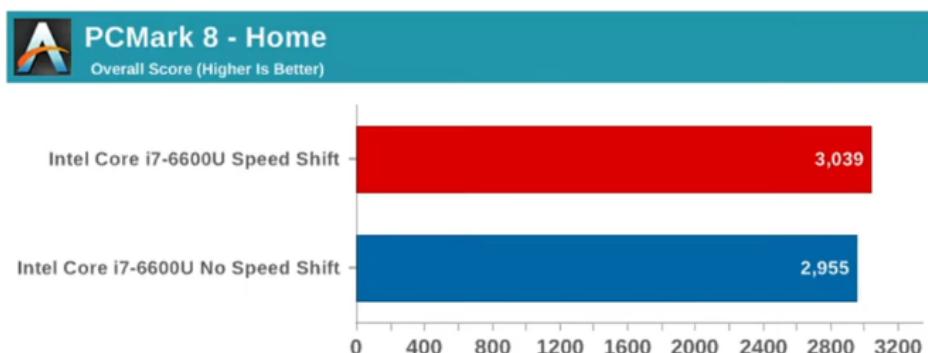


6.9. ábra. Intel SpeedShift (CPPC) és DVFS frekvencia átállási ideje

A CPPC-nél az OS csak felhasználói igényeket továbbít, mint pl. a teljesítmény vagy a fogyasztás preferálása. Ezután a vezérlést átadja a PCU-nak, ami önállóan beállítja a szükséges paramétereket, valamint végrehajt optimalizációkat is. Az optimalizálásokat több algoritmus segítségével hajtja végre. Az operációs rendszer felügyeli a PCU működését és hiba esetén beavatkozhat.

További fejlesztés még a DVFS-hez képest, hogy a diszkrét munkapontok helyett bármilyen szorzó beállítható. Ezzel finomabb szabályozás valósítható meg.

Az alacsonyabb felfutási időnek köszönhetően a CPPC segítségével nagyobb teljesítmény érhető el (???. ábra).



6.10. ábra. Benchmark eredmények CPPC-vel és anélkül

A teljesítmény javulása viszont nem számottevő (kb 1.5%). Más alkalmazásoknál kb. 20% javulás érhető el. A további gyorsításért az Intel a Kaby Lake processzorokban javított a SpeedShift technológián, a korábbi 35 ms-os felfutási időt 10 ms-re csökkentették.

Az AMD CPPC Implementációja már egy korszerűsített CPPC változatot valósít meg, ami az ACPI 5.1 szabványból származik. A felfutási időt az AMD 1 ms környékére csökkentette.

## AVFS

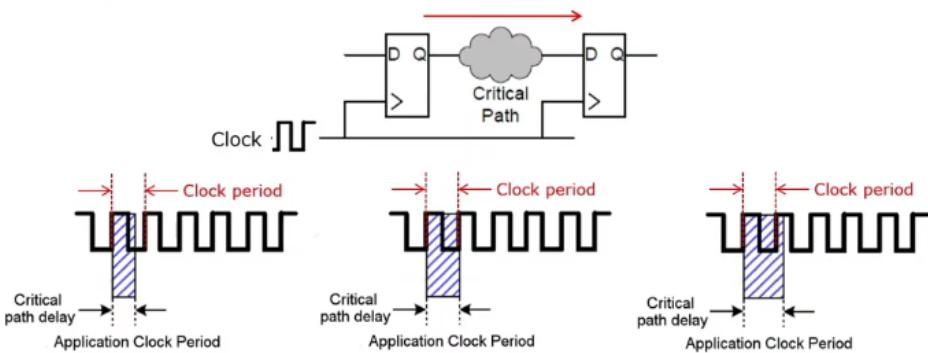
A DVFS-nél az OS előre meghatározott frekvencia-feszültség párosokból választ működési pontokat. Ezeket a működési pontokat minden CPU fejlesztése során mérésekkel határozzák meg (adott frekvenciához melyik az a legkisebb feszültség, amin még stabilan működik a CPU), majd ezt egy biztonsági értékkal növelik (guard band), hogy rosszabb körülmények is jó legyen a CPU. Ilyen körülmények pl.:

- gyártási eljárás miatti eltérések
- processzor belséjében tápfeszültségek eltérései

- processzor hőmérséklete
- processzor öregedése

A gyártási eljárás figyelembe vétele nél azokkal a legyártott darabokkal kell számolni, amiknek a legmagasabb a minimum magfeszültsége. Probléma, hogy mindenek miatt a biztonsági sávot nagyon nagynak kell megválasztani.

Az AVFS erre jelent megoldást, csökkenti a szükséges biztonsági sávot. Alapja, hogy nem a legrosszabb esetet veszük figyelembe, hanem helyben történik a minimális feszültség meghatározása. A helyi mérés kritikus út monitorokkal valósul meg, tehát a kritikus áramköri részeken áthaladó jeleket méri a processzor és hasonlítja össze az órajelrel. Ha a jel hamarabb érkezik meg, mint a következő órajel, akkor még növelhető a frekvencia vagy csökkenthető a feszültség. Ha pont egyszerre, akkor elérte a jó beállítást. Ha pedig később érkezik meg a vizsgált jel, mint az órajel, akkor csökkenteni kell a frekvenciát, vagy növelni a feszültséget. Ezeket az eseteket mutatja a 6.11. ábra.



6.11. ábra. Órajel és frekvencia beállítása kritikus út monitorozással

Az AVFS a kritikus út monitorok jelzései alapján be tud állítani egy, a gyártásnál meghatározottnál jóval kisebb biztonsági sávot. Ezzel jelentős mértékben csökkenthető a fogyasztás, vagy növelhető a teljesítmény.

Az AVFS kétféleképpen használható fel: AVS és AFS. Az AVS (Adaptive Voltage Scaling vagy undervolting) a csökkentett biztonsági sávot az előre meghatározott P állapotok feszültség értékeinek meghatározására használja, így csökkentve a fogyasztást. Az AFS (Adaptive Frequency Scaling vagy overclocking) viszont a P állapotokhoz tartozó frekvenciákat növeli meg, így növelte a teljesítményt.

## 6.8. Az Intel Turbo Boost technológiája

### 6.8.1. Bevezetés

Az Intel a Nehalem-nél (2008) vezette be a Turbo Boost első verzióját. Ezután két másik verziót hoztak ki: 2.0 (Sandy Bridge, 2011) és 3.0 (Broadwell-E, 2016).

### 6.8.2. Első generáció

A Turbo Boost 1.0 nagyrészt a TDP-re alapoz. Ha a processzor aktuális disszipációja alacsonyabb, mint a TDP pl. kis számítási igényű alkalmazások futtatásánál, a Turbo Boost megemeli az aktív magok órajeleit. Így ha pl. négy mag közül csak kettő aktív, az inaktív magok által nem elhasznált disszipáció átadható az aktív magoknak, amik így magasabb teljesítményen működhetnek.

A Turbo Boost-al együtt jelent meg a PCU is. Feladatai:

- DVFS kezelése
- Turbo Boost vezérlés
- a teljes disszipáció és a hőmérséklet ellenőrzése, szükség esetén a frekvencia csökkentése

A Turbo Boost aktiválásához négy feltételnek kell teljesülnie:

- az OS igényli a legmagasabb teljesítményű (P0) állapotot
- az aktuális fogyasztás kisebb mint a TDP
- az aktuális áram egy megadott érték alatt van
- a lapka hőmérséklete egy megadott érték alatt van

Ha ezek teljesülnek, a PCU megnöveli a frekvenciát és így a teljesítményt. Ehhez tudnia kell a PCU-nak, hogy milyen Turbo frekvenciát válasszon. Az első generációban ez az aktív magok függvénye (kevesebb aktív mag = alacsonyabb Turbo órajel és fordítva). A maximális Turbo Boost órajeleket a gyártás során határozzák meg, szorzófaktorok formájában. Ezeket a szorzókat az MSR 1ADH regiszterben tárolja a processzor (Model Specific Register).

## 7. fejezet

# Az AMD Zen processzorai

### 7.1. Bevezetés

Az AMD a korábbi lemaradás után az utóbbi években újra felzárkózott az Intelhez és több kategóriában megverte őket teljesítményben. A Zen elnevezés vonatkozhat általában a Zen mikroarchitektúrára épülő processzorokra és az első generációs Zen CPU-kra is. A fejezetben ezért az első generációs Zeneknél explicit jelöljük a generációt. A gyártó a mobil elnevezést a laptop processzorokra használja.

### 7.2. Áttekintés

Az 1960-as években a Fairchild Semiconductor vállalatból több cég kivált, de ezek közül csak az Intel és az AMD nem olvadt be más cégekbe. Napjainkban ez a kettő határozza meg az x86 alapú processzorgyártást.

Az AMD először az Inteltől licenszelt processzorokat gyártott, 1996-ban a K5 processzorokkal jelentek meg a saját fejlesztésű termékeik. A K8 a világ első 64 bites processzora volt (2003). Ezután több, nem túl sikeres processzort gyártottak (Bulldozer, Cat), az újabb siker 2017-ben jött a Zen családdal.

Az AMD jelentőségteljes fejlesztései közé tartozik az x86-os utasításkészlet 64 bites kiterjesztése. Ekkor az Intel és a HP 64 bites VLIW architektúrával próbálkozott (Itanium), az AMD viszont a meglévő x86-os architektúra kiterjesztésében látta a jövőt. Ebből az AMD került ki győztesen. A 64 bit megvalósításához ki kellett terjeszteni a regisztereket és az utasításokat is ki kellett bővíteni.

Az AMD másik fontos fejlesztése a DCA (Direct Connect Architecture) multiprocesszoros megoldás. Ezzel a korábban az északi hídon lévő memóriavezérlőt integrálták a processzorlapkára, kiküszöbölvé a memória szűk keresztmetszetét (főleg szervereknél volt probléma). A DCA része még a HyperTransport link, amiből minden processzorra kerül három és a processzorok összekötését szolgálják.

A fejlesztéseknek köszönhetően az AMD 2003-ban megjelent processzorai teljesítményben jobbak voltak az Intelénél. Így egyre nagyobb piacot tudott magának szerezni. Az Intel csak később vette át ezeket az újításokat, de végül ők is rákényszerültek az x86 64 bites kiterjesztésére (2004). A lapkára integrált memóriavezérlőt viszont csak 2008-ban hozta be az Intel a Nehalem processzorokkal.

Az AMD piaci részesedése viszont 2006-tól csökkenni kezdett, mikor az Intel kihozta a Core 2 processzorait és az AMD elő került teljesítményben. Ez a tendencia 2017-ben fordult meg, az AMD Zen processzorok megjelenésével. A Zennel nagy mértékű teljesítmény és hatékonyság növekedést értek el, 2021 elején egymagos teljesítményt nézve az AMD Ryzen 9 5900X volt a leggyorsabb asztali processzor.

A szervereknél szintén az AMD EPYC processzorok vezetik a benchmarkokat. Az Intel eléggyé elmarad, viszont közben megjelentek az ARM alapú szerverek, amik teljesítményben majdnem elérik az AMD-t. Az ARM architektúrák teljesítmény felvételben kedvezőbbnek tűnnék, tehát valós konkurencia az x86-nak.

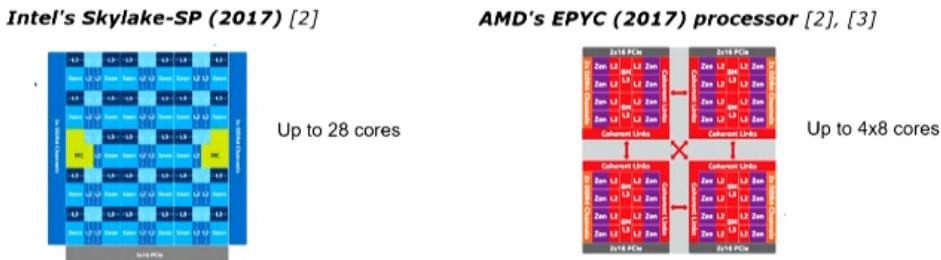
## 7.3. A Zen család tervezési elvei

Az AMD 300 mérnöke dolgozott a Zen mikroarchitektúrán, hogy teljesítményben legyőzzék az Intelt. A megvalósítási alapelvek:

- többlapkás megközelítés
- lego-jellegű rendszer, három különböző lapkatípusból lehet összerakni a processzorokat:
  - CPU lapkák
  - APU lapkák - számítási és grafikai feldolgozást is megvalósít
  - IO lapkák
- három szintű hierarchikus tervezés
- a lapkák összekötéséhez szükséges egy új kapcsolóhálózat kifejlesztése, ez lett az Infinity Fabric (IF)
- a lapkákat egybe kell tokozni, ehhez egy két dimenziós tokozást alkalmaztak (MCM - Multi Chip Module)

### 7.3.1. Többlapkás felépítés

Korábban a processzorokat monolitikusan terveztek, azaz minden mag egy lapkára került. Ilyen az Intel Skylake-SP (2017) is, ami 28 magot tartalmaz egy lapkán. Az AMD EPYC (2017) viszont többlapkás megvalósítást használ, négy, egyenként nyolc magos lapkát tartalmaz egy tokozásban (7.1).



7.1. ábra. Monolitikus és többlapkás felépítés

A többlapkás felépítés fő előnye, hogy kisebb lapkák gyártásával kisebb a hibaarány, így nagyobb lesz a kihozatal, tehát a gyártás költséghatékonyabb. Az AMD becslése szerint egy 32 magos processzornál a többlapkás felépítés gyártási költsége mindössze 60%-a az egylapkás felépítésűnek. További előny, hogy ugyanazokból az alapelemekből több processzort is ki lehet alakítani, valamint a memória csatornák és az IO skálázható a magok számával.

Hátrány viszont, hogy egy cache koherens összekapcsoló hálózat kell, ami csökkenti a teljesítményt és növeli a fogyasztást. Ezen kívül a tokozás növeli a gyártási költségeket.

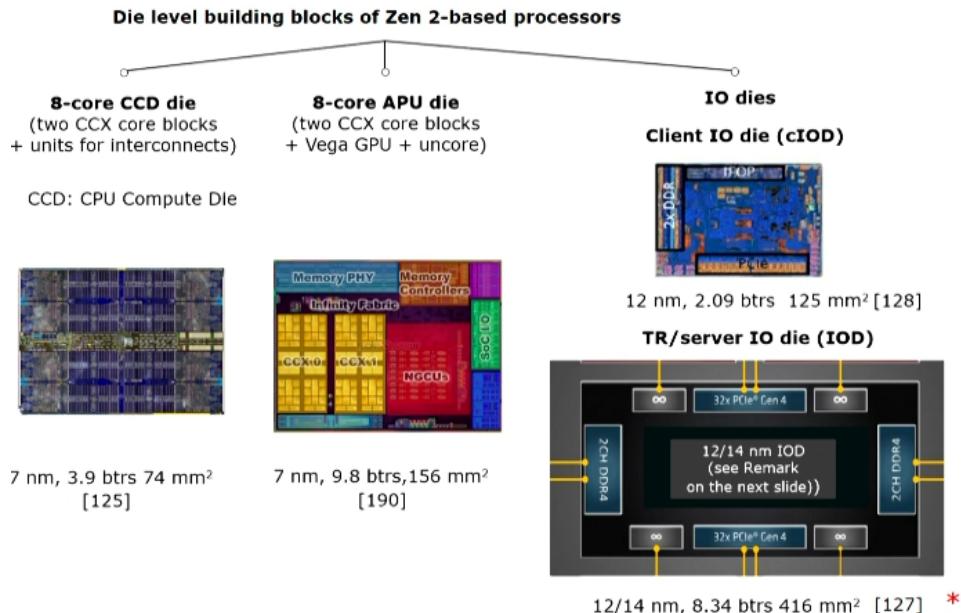
A többlapkás felépítést az Intel is tervez átvenni, valamint az Nvidia is végez kutatásokat a GPU-k többlapkás felépítésével kapcsolatban.

### 7.3.2. Moduláris kialakítás

A teljes Zen család három fajta lapkából épül fel (CPU, APU és IO).

A CPU lapkák nem tartalmaznak GPU-t. Az első generációban Zeppelinnek, később CCD-nek (CPU Compute Dies) hívták. Az APU lapkák CPU-t és integrált GPU-t is tartalmaznak, az IO lapkák pedig az IO-ért felelnek. IO lapkáknál megkülönböztetünk szerverekben és HEDT gépekben használt IOD-okat (IO Dies) és kliensekben használt cIOD-okat (Client IO Dies).

Az asztali processzorok, a HEDT-ek és a szerverek nem tartalmaznak grafikát, több lapkából állnak és tartalmaznak IO lapkát is. Ezzel szemben a laptopokba és asztali gépekre szánt APU-k egy lapkás SoC-ök, GPU-t is tartalmaznak. A különböző lapkatípusokat mutatja be a 7.2. ábra.



7.2. ábra. A Zen 2 processzorokat alkotó lapkák

A mobil és desktop APU-k egy APU lapkából állnak, legfeljebb 8 maggal (Ryzen 4xxU/H/HS és Ryzen 4xxG/GE). A GPU nélküli desktopok (Ryzen 3xxx/X) 1 vagy 2 CCD lapkát és 1 cIOD lapkát tartalmaznak (max 16 mag). 24 és 32 magos HEDT processzoroknál 4 darab 6 vagy 8 aktivált magos CCD és 1 IOD lapka van a foglalatban, 64 magnál pedig 8 darab 8 magos CCD és 1 IOD (Threadripper 39xxX). Szervereknél 24 és 32 mag esetében 4 darab 4 vagy 6 magos CCD és 1 IOD, 48 és 64 magnál pedig 8 darab 4 vagy 6 magos CCD és 1 IOD dolgozik.

### 7.3.3. A CPU-k és GPU-k három szintű hierarchikus tervezése

Az AMD a CPU és GPU lapkákat hierarchikus módon tervezte. Jelentése, hogy az alap építőkocka a mag, a magok összerakásából funkcionális egységek építhetők, a funkcionális egységekből pedig lapkák.

Ezek az építőkockák:

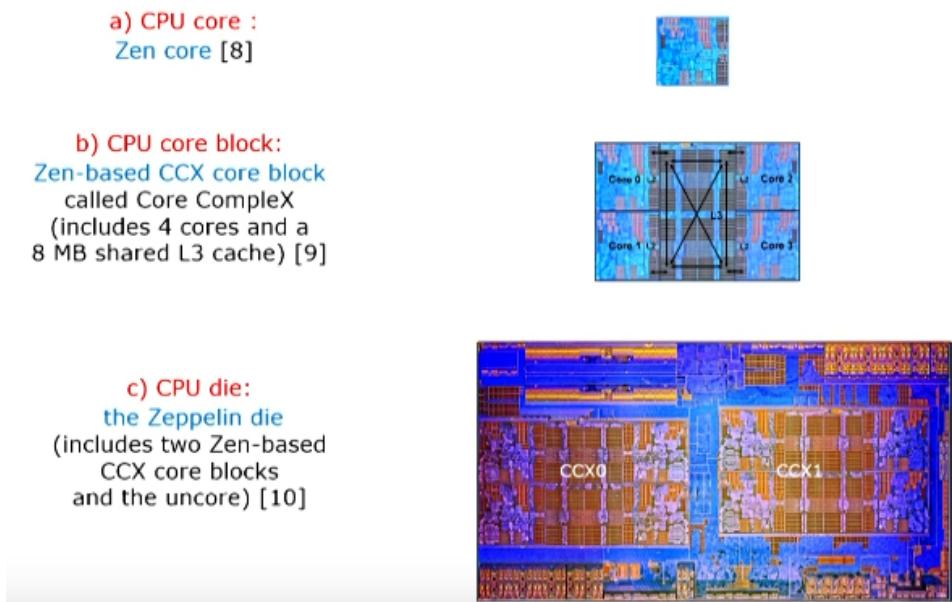
- CPU: CPU magok, CPU mag blokkok, CPU lapkák
- Grafika: GPU magok, GPU, APU lapka (egy GPU és egy vagy két CPU mag blokk)
- IO: csak kétféle lapka (HEDT/szerver és kliens)

## A CPU építőelemei

A CPU magok a Zen processzorok alap építőelemei. Ezekből alakulnak ki a CPU blokkok, amik 4 vagy 8 magot tartalmaz. A magoknak saját L1 és L2 gyorsítótárai vannak, az L3 cache viszont osztott és szeletelt. A szeletek egymás mellett párhuzamosan képesek dolgozni. A magcsoportok neve CCV (Core CompleX). A harmadik szint a lapka szint, a lapka egy vagy két CCX-et tartalmaz. Az első generációs CPU-kat Zeppelin lapkának hívták, később CCD-re (Core Complex Dies) nevezték át.

Az Intel ezzel szemben nem magcsoportokat használt, hanem szimmetrikus többmagos rendszereket épített. 2020-ban az Intel viszont a Lakefield családdal áttért a magblokkok használatára és a big.little felépítésre, többlapkás felépítést használva.

A 7.3. ábrán látható egy példa a CPU hierarchikus felépítésére.



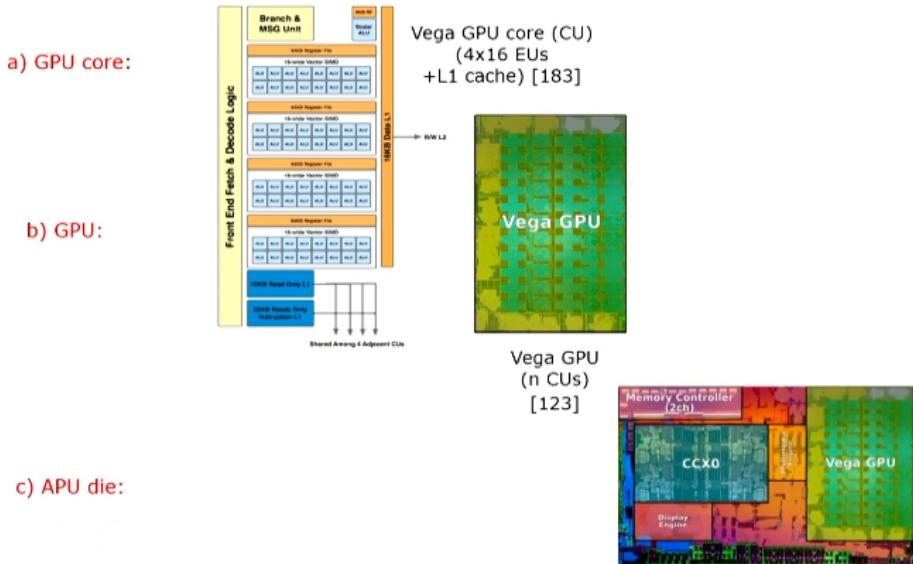
7.3. ábra. A Zen 1 processzorok felépítése

## A GPU építőelemei

Itt az alap építőelem a GPU mag, ami megegyezik az AMD diszkrét videokártyáiban használt Vega CU-kkal (Compute Unit). A második szint a Vega GPU-k amik 3-11 Vega CU-t tartalmaznak és egyéb szükséges egységeket. Harmadik szinten az APU lapka van, ami egy Vega GPU-t és egy vagy két CCX-et tartalmaz.

Az APU-nak a GPU-n és CCX-en kívül szüksége van egyéb blokkokra a működéshez, ezeket összefoglalva uncore-nak nevezzük.

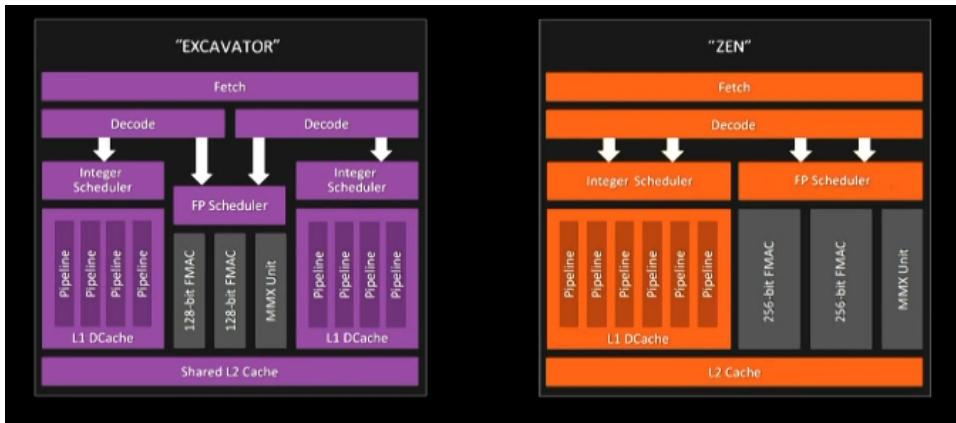
Egy APU felépítése látható a 7.4. ábrán. A 7.4. ábrán látható egy példa a CPU hierarchikus felépítésére.



7.4. ábra. Az APU SoC felépítése

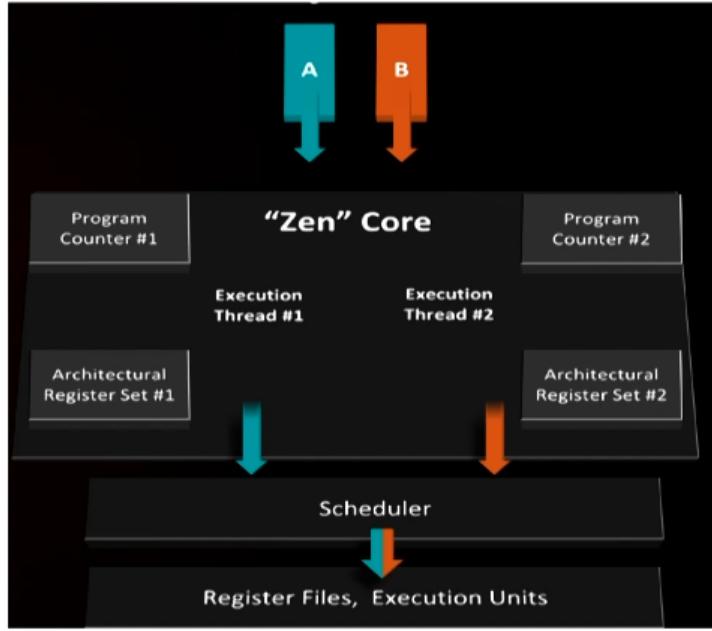
## 7.4. A Zen magok fejlődése

A Bulldozer processzorokhoz képest több téren előrelépést jelentett. A Bulldozer magban két egész feldolgozó egységet integráltak egy lebegőpontos feldolgozóval és ezt a modult tekintették két magosnak. Ez nem vált be, ezért a Zen visszatárt a hagyományos mag felépítéshez (7.5).



7.5. ábra. Különbség a Bulldozer alapú és a Zen architektúrák között

A Bulldozernek gyengéje volt még, hogy nem alkalmazott többszálúságot, a Zen viszont már megvalósította (7.6).



7.6. ábra. Többszálúság a Zen magokban

A harmadik nagy újítás egy olyan műszaki csomag volt, ami segített a disszipáció csökkentésében és a teljesítmény növelésében. Az AMD ezt SenseMI-nak nevezte el és többek között adaptív feszültségszabályzást, Turbo Boostot, frekvencia növelést, neurális hálós elágazásbecslést és intelligens utasítás lehívást valósított meg.

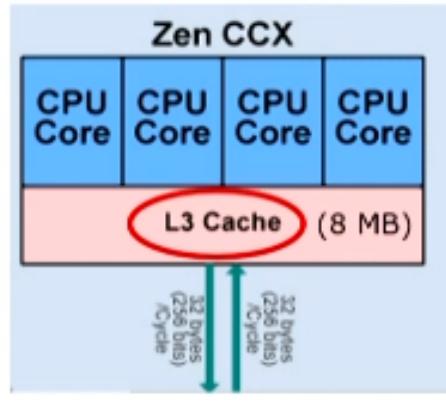
Eredmények:

- 14nm technológia a korábbi 28 helyett
- 52%-os egyszálas teljesítmény javulás
- 3.7x-es javulás teljesítmény/wattban
- többszálúság

A Zen 1 után megjelent a Zen 2 és a Zen 3, 2022-re ígérik a Zen 4-et. A Zen 2-nél 8-ról 16 MB-ra emelték az L3 cache-t és több, kisebb lapkát használtak fel. A Zen 3 még tovább növelte az L3 méretét 32 MB-ra és itt már 4 magos komplexeket használtak.

## 7.5. A CCX-ek fejlődése

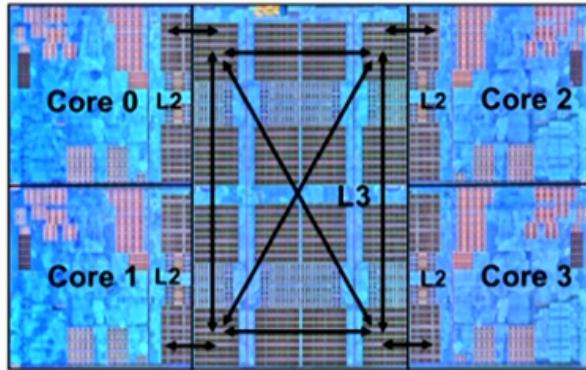
A Zen 1-nél 4 magos CCX-eket alkalmaztak (7.7. ábra)



(2017)

7.7. ábra. Zen 1 CCX felépítése

A magok saját L1 és L2 cache-el rendelkeztek. A CCX 8 MB méretű közös L3 cache-t biztosít a magoknak, a külső komponensekkel pedig egy 32 byte széles, kétirányú linkkel kapcsolódik. Az L3 cache négy darab 2 MB-os szeletre van osztva, amik cím átlapolással címezhetők. A szeletek crossbar kapcsolattal kapcsolódnak, azaz bármelyik mag képes elérni bármelyik szeletet azonos időben (7.8. ábra). Ez többnyire exkluzív cache, tehát az L2-ből kimiradó adatokat tárolja (áldozat cache).



7.8. ábra. Zen L3 cache felépítése

Az ARMv8 ISA szintén magkomplexumokra épül, de megosztott L2 cache-t használ a Zennel ellentétben. Később az ARM (8.2) már szintén privát L2 cache-t használt a magkomplexumoknál. Ezknél már megjelent a big.little architektúra is, azaz kis és nagy magokat kombinálhatnak a magkomplexumok. Az AMD megvalósítása jobb volt mint az ARM 8.0, de elmarad a 8.2-től.

A Zen 2 architektúra megnövelte az L3 cache méretét, két CCX magblokkot tartalmaz és minden magblokknak 4 magja és 16MB L3 cache-e. Itt a két cache összekötését meg kellett oldani, a Zen 3 viszont már közös 32 MB L3 cache-t biztosít a 8 magnak. Ez csökkenti az elérési időket és a mag-mag futási időket.

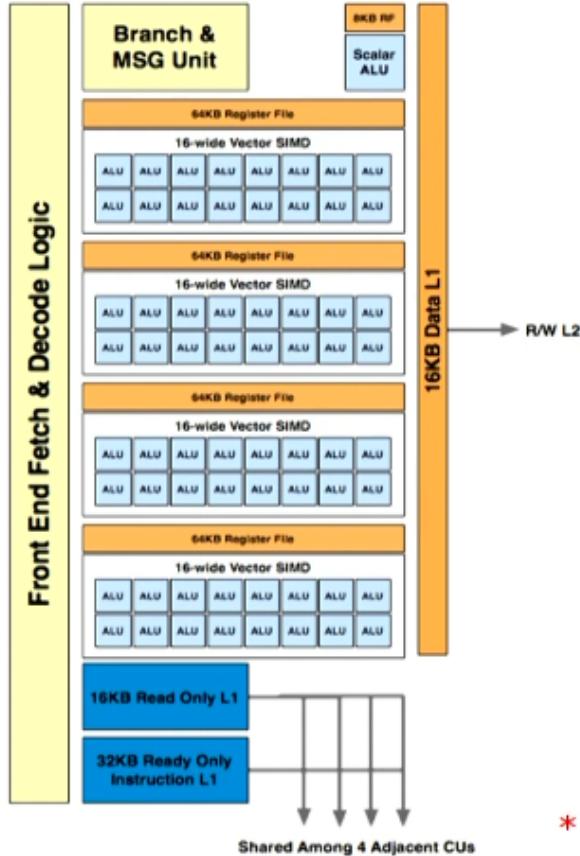
## 7.6. A grafikai magok fejlődése

Az AMD a Vega számítási egységeit használta fel a Zen processzorokban. A GPU-k változó számú CU-kat (Compute Unit) tartalmazhatnak. A Vega CU architektúrája az AMD GCN (Graphics Core Next)

GPU család 5. generációja, 2017-ben vezették be. Ez már támogatja a mesterséges intelligenciához használt 8 vagy 16 bites adatformátumot is.

### 7.6.1. A Vega CU felépítése

Minden CU-ban négy SIMD feldolgozó egység van, amikben egyenként 16 ALU van (7.9). A SIMD egységek egymástól függetlenül működnek és ugyanazt az utasítást hajtják végre 16 szálon.



7.9. ábra. A Vega CU felépítése

Minden SIMD egység egy 64 KB-os regiszter fájllal rendelkezik, az ALU-k pedig csak lebegőpontos műveleteket hajtanak végre, ezért kell még egy 32 bites, fixpontos utasításokat végző ALU, ami megosztott a négy SIMD egység között. Az ALU-k lebegőpontos, 32 vagy 64 bites műveleteket tudnak végrehajtani. Ismerik a fused és unfused megoldásokat, a fusednál pl. egy 32 bites szorzás eredménye 64 bit.

Egy Vega GPU több CU-ból áll (3-11 db).

## 7.7. A Zen processzorok építőelemei és felépítése

### 7.7.1. Első generáció

Az első generációban kétféle blokkból épültek fel a processzorok:

- 8 magos Zeppelin lapka (2 db 4 magos CCX + uncore)

- 4 magos APU lapka (1 db CCX + uncore)

Ebből a különböző processzor típusok felépítése:

- Mobil APU-k: 1 db APU lapka
- GPU nélküli desktop: 1 db Zeppelin lapka
- HEDT-ek: 2 Zeppelin lapka
- 1S/2S szerverek: 4 Zeppelin lapka

A Zeppelin lapkákat az IF (Infinity Fabric) köti össze és a memóriavezérlők is a lapkákkal együtt skálázódnak.

### 7.7.2. Második generáció

Itt már három blokkból lehetett válogatni:

- 8 magos, GPU nélküli CCD lapka (két CCX + uncore)
- 8 magos APU lapka (két CCX + Vega + uncore)
- IO lapkák (cIOD - kliens IO lapka vagy IOD - server és HEDT IO lapka)

Ebből felépülő processzorok:

- Mobil és Desktop APU-k: 1 APU lapka
- GPU nélküli desktop: 1 vagy 2 CCD lapka + 1 kliens IO lapka
- HEDT: 4 vagy 8 CCD + 1 IOD lapka
- 1S/2S szerverek: 4 vagy 8 CCD + 1 IOD lapka

### 7.7.3. Harmadik generáció

Építő blokkok:

- 8 magos, GPU-t nem tartalmazó CCD lapka (egy CCX + uncore)
- 8 magos APU lapka (CCX + Vega GPU + uncore)
- cIOD és IOD

Ebből épülő processzorok:

- Mobil és Desktop APU-k: 1 APU lapka
- GPU nélküli desktop: 1 vagy 2 CCD lapka + 1 kliens IO lapka
- 1S/2S szerverek: 4 vagy 8 CCD + 1 IOD lapka

## 7.8. Infinity Fabric

Az Infinity Fabric az AMD Zen architektúrához tervezett cache koherens kapcsolati rendszer. Feladatai:

- lapkák összekötése cache koherens módon
- bizonyos vezérlési funkciók biztonsítása

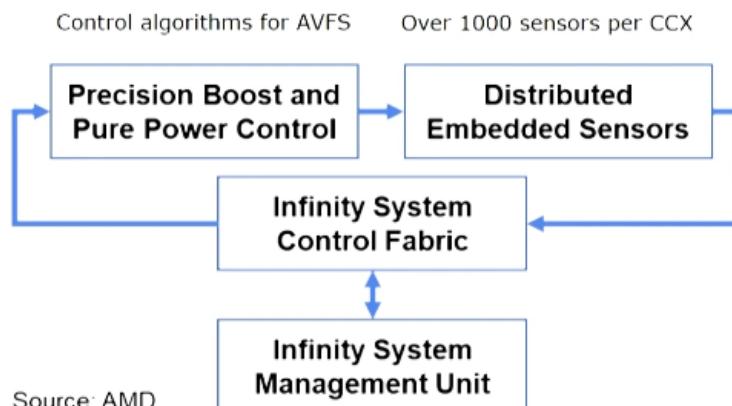
Alapja a korábban használt HyperTransport buszrendszer (hasonló, mint az Intel QPI - Quick Path Interconnect). A feladatok ellátásához két alapvető részből áll az Infinity Fabric: Scalable Control Fabric és Scalable Data Fabric.

### 7.8.1. Scalable Control Fabric

Felelősségi körei:

- disszipáció kezelése
- biztonság
- minőségi kérdések

Tartalmaz központi és távoli elemeket. A távoli elemek jellemzően szenzorok mint pl. hőmérséklet, frekvencia vagy feszültség szenzorok. Ezeket az AMD egy zárt hurokban használja, egy központi vezérlés mellett (7.10).



7.10. ábra. A Control Fabric működése

A szenzorok száma nagyon nagy ( $>1000$ ).

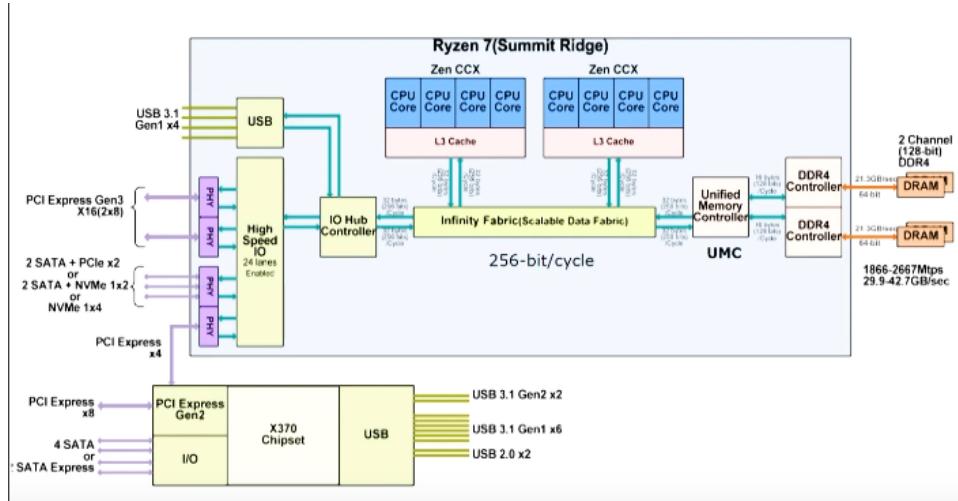
### 7.8.2. Scalable Data Fabric

A HyperTransport busz továbbfejlesztett változata. Feladatai:

- lapkán belül a különböző egységek összekapcsolása (CCX, memória és IO vezérlők)
- lapkák összekapcsolása (IFOP On-Package linkek)
- 2S szervereknél a tokok összekötése (IFIS: IF Inter-Socket linkek)

### 7.8.3. Példa: első generációs, GPU nélküli desktop Zen CPU

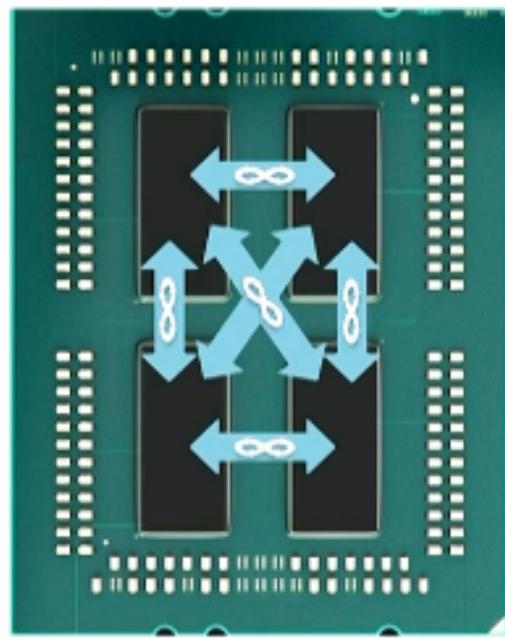
A Ryzen 1000 két CCX blokkot és egy IO vezérlőt tartalmaz, amiket az Infinity Fabric köt össze, 256 bit széles adatátvitelt garantálva (7.11).



7.11. ábra. A Ryzen 1000 IF felépítése

### 7.8.4. Példa: első generációs EPYC szerver CPU

Az EPYC szerverprocesszorok egy tokozásban több lapkát tartalmaznak, amiket az Infinity Fabric (IFOP) köt össze ( minden lapkát minden lapkával).



The 4 dies are  
8-core Zeppelin dies

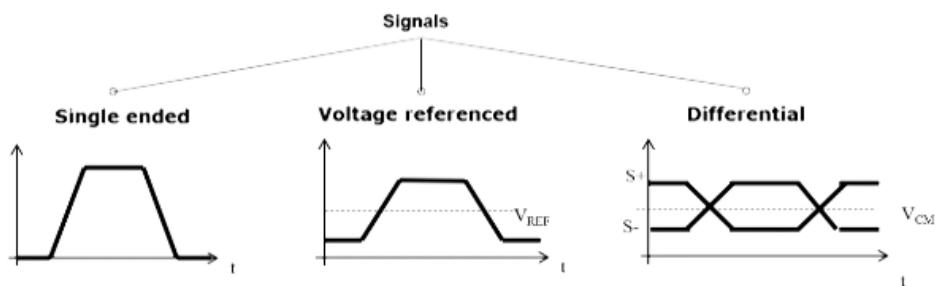
7.12. ábra. Négy Zeppelin lapka összekötése IFOP linkkel

Ezek a linkek kétirányú 32 bites kapcsolatot valósítanak meg, föld-referenciált (single-ended) jelekkel, differenciált órajellel. Az energiaigénye átvitelenként 2 pJ, egy memóriafrekvencia során pedig 4 átvitel történik.

### Jelfajták

A buszok esetében három jelrendszer különböztetünk meg (7.13):

- Föld referenciált - single ended: TTL, LVTTL és SDRAM használja
- Feszültség referenciált: FSB, DDR használja
- Differenciál erősítő minden két bemenetén jel van: QPI, PCIe, egyéb soros buszok

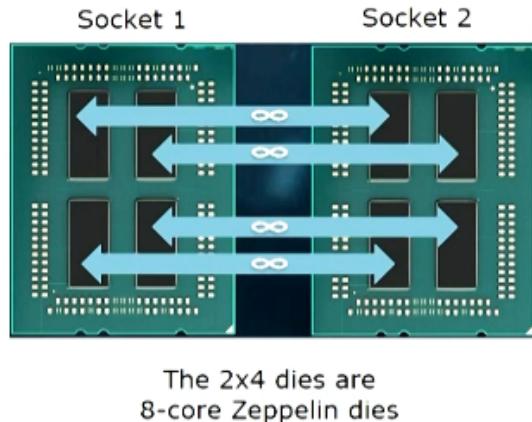


7.13. ábra. Buszok jelrendszeri

Az ábrán balról jobbra egyre kisebb löketű és egyre jobb biztonságú jelek láthatók, ezért az új rendszerek szinte kizárolagosan a differenciált jelet használják. Ennek viszont a hátránya a nagyobb teljesítmény igény.

#### 7.8.5. Példa: első generációs 2S EPYC szerver CPU

2S szervereknél az IFIS (Infinity Fabric Inter-Socket) köti össze a tokokat, minden lapkát lapkát minden lapkával.



7.14. ábra. IFIS link felépítése 2S EPYC processzoroknál

Az IFIS kapcsolatonként 16 vezetékpár ból épül fel, amiknek hosszabb távot kell megtennie ( $>20$  mm). A nagyobb távolság miatt több zavar probléma léphet fel, ezért itt már áttértek a megbízhatóbb, differenciált átvitelre. Ennek a következménye a nagyobb teljesítmény igény: 11 pJ/bit.

#### 7.8.6. Második generációs IF

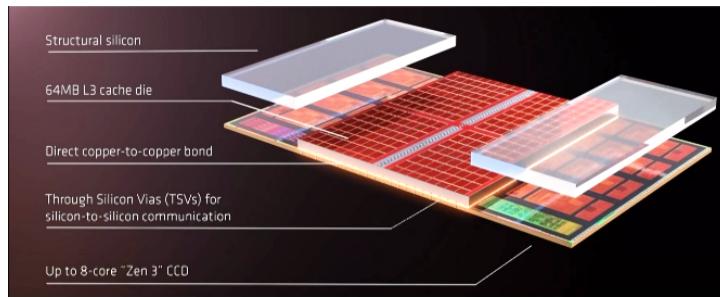
Az Infinity Fabric második generációjában növelték az átviteli szélességet 512 bitre, a működést függetleníteték az órafrekvenciától és 27%-kal csökkentették a teljesítmény igényt.

### 7.9. MCM tokozás

Egy tokozás háromféle lehet:

- 2 dimenziós (horizontális lapka elhelyezés): a lapkák egymás mellé kerülnek, közöttük elektromos kapcsolat áll fent (pl. Intel EMIB, AMD MCM)
- 3 dimenziós (vertikális tokozás): a lapkák egymás fölé kerülnek és itt is megvan az összeköttetés. Előnye a nagyobb sűrűség, hátránya a rosszabb hűtés. (pl. Intel Foveros)
- 2D és 3D kombinációja: Foveros+EMIB

Az MCM (Multi-Chip Module) egy két dimenziós tokozási eljárás. 2021 augusztusában az AMD jelentette, hogy ők is készítenek 3D tokozási eljárást. A megoldásnak köszönhetően 192 MB-ra tudják emelni az L3 cache-t (7.15).



7.15. ábra. Az AMD által bejelentett 3D tokozás

## 7.10. A Zen processzorcsalád tagjai

A Zen családon belül különböző márkanevekkel látják el az eltérő kategóriák processzorait:

- mobil és kliens processzorok: Ryzen
- HEDT: Ryzen és Threadripper
- szerverek: EPYC

A Ryzenen belül sorozatok különböznek el: 3, 5, 7 és 9. Ezek a teljesítményre vonatkoznak és a sorozat határozza meg a magszámot is. A belépő szinten 4-6 mag, magasabb teljesítményű desktopoknál 16, HEDT-eknél pedig 64 mag jellemző.

A processzorok elkülöníthetők GPU-t tartalmazó (mobil, desktop) és nem tartalmazó (desktop, HEDT, szerver) CPU-kra. A GPU-t nem tartalmazó CPU-k több lapkából állnak (kivéve 1. generációs GPU nélküli Ryzenek), a GPU-val szereltek viszont egylapkás SoC-ok.

Tokozás szempontjából eltérnek a desktopok és a mobilok. Míg a desktopokat PGA (Pin Grid Array) tokozással látják el, a mobilok BGA (Ball Grid Array) forrasztott tokozást használnak. A szerverek és desktopok tokozása fizikailag azonos, de nem kompatibilisek egymással.

A Pro jelzéssel ellátott processzorok biztonsági, megbízhatósági és egyéb, főleg vállalati felhasználásban hasznos funkciót tartalmaznak.

## 7.11. Példa: Zen 3 alapú, GPU nélküli Ryzen 5000

A processzor a Vermeer architektúrára épül, 2020 végén jött ki. Játékos célokat használ, a frekvencia szorzója nem rögzített. Két változata van:

- 8 magot és két lapkát tartalmazó tartalmazó kisebb verzió (CCD + cIOD)
- három lapkát tartalmazó nagyobb verzió (2xCCD + cIOD)

Mind AM4 tokozással rendelkeznek és 400 vagy 500-as chipkészlettel működnek. A Ryzen 9 5950X vette át elsőnek az Inteltől a teljesítmény koronát, eközben viszonylag olcsók és fogyasztásban is jobbak az Intelknél.

Benchmarkok alapján az egyszálas, többszálas és valós felhasználási esetekben is kb. 20 év után az AMD a Zen 3 processzorokkal átvette a vezetést az Inteltől.

Jelenleg a Zen 3 alapú AMD processzorok a szerverekben is átvették a vezetést az Inteltől.

## 8. fejezet

# ARM processzorok

Az ARM az Intel és az AMD mellett a harmadik nagy szereplő, a mobil világban egyeduralkodó.

### 8.1. Terminológia

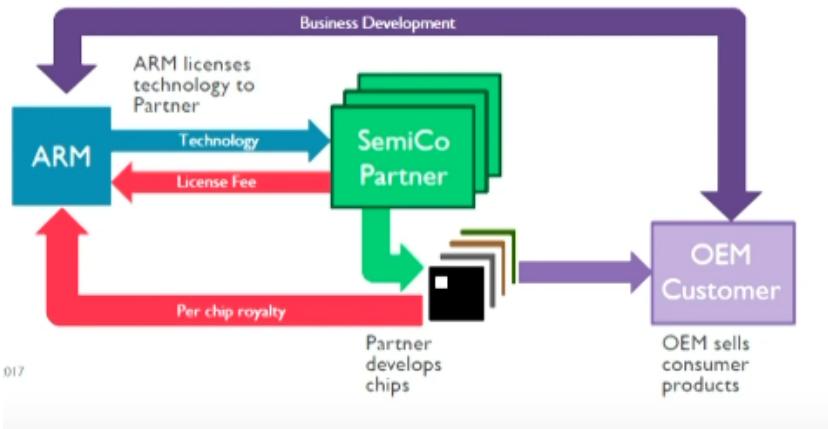
Az ARM nem processzorgártó cég, hanem CPU-kat, GPU-kat, NPU-kat és egyebeket fejlesztenek és a szellemi tulajdonról értekesítik. Ezzel szemben az Intel, AMD, Qualcomm stb. komplett processzorokat fejlesztenek és forgalmaznak, részben az ARM szellemi tulajdonra alapján. Ezért az ARM-re CPU - és nem processzor - fejlesztőként hivatkozunk.

### 8.2. Áttekintés

Az ARM egy brit félvezető fejlesztő cég (1983-ban alapították, Acorn RISC Machine, de 1990-től az Apple és a VLSI-vel közös projekttel Advanced RISC Machines), amit 2016-ban megvett egy japán bank, a Softbank. 2021-ben az Nvidia bejelentette, hogy megveszi az ARM-et, de ez még folyamatban van. A cég alapvetően tervezéssel foglalkozik, kis áramfogyasztású processzor komponenseket és ezekhez kapcsolódó tervezési eszközöket fejleszt. Ezek szellemi termékek, a bevételük ezeknek a licenszeléséből származik.

#### 8.2.1. Üzleti modell

Az ARM a technológiát licenszeli a félvezető gyártó partnereinek, akik ezért licenszdíjat fizetnek. A félvezető gyártó a legyártott processzorokat eladja az OEM gyártóknak (közben royalty-t fizetnek az ARM-nek), az OEM gyártók pedig végfelhasználói termékeket értékesítenek. Az ARM figyelembe veszi a partnerek visszajelzéseit.



8.1. ábra. Az ARM üzleti modellje

### 8.2.2. Licenszelés

Az ARM két licenszet forgalmaz:

- Cortex licensz: az ARM egy komplett mikroarchitektúrát értékesít, konfigurálható opciókkal
- Architektúra licensz: az ARM csak az ISA-t adja át a vevőnek, a vevő dolga megtervezni a saját mikroarchitektúráját (pl. Apple és Samsung)

### 8.2.3. Az ARM dominanciájának okai

Az ARM az x86-al szemben egy RISC architektúra, a RISC pedig Load/Store architektúra, tehát az operandusokat a regiszterekből olvassa be. Az x86 viszont CISC és megenged operáns olvasást közvetlenül a memóriából. Következmények:

- a CISC utasítások nagyon hosszúak lehetnek a memória címzése miatt (base reg, offset, stb.)
- a RISC-eknél az utasításoknál elég a regisztereken címezni, a Load/Store-nál pedig a memóriát
- a CISC-ek utasításhossza változik, a RISC fix (általában 32 bit)
- a CISC-ek több áramot fogyasztanak, mivel az utasítás lehívás és a dekódolás komplexebb

A mobil világban nagyon fontos az alacsony fogyasztás, ezért a RISC alapú ARM architektúrák előnye. Az ARM processzorok további fejlődésben vannak, újabban a szervereknél is kezdenek teret nyerni. 2020-ban, mikor az ARM-et megvette az Nvidia, az Nvidia CEO-ja kijelentette, hogy a desktopoknál is a legfontosabb architektúra az ARM lesz a jövőben.

## 8.3. Az ARM ISA fejlődése

Az ARM (RISC) architektúra kezdetben kevés utasítást tartalmazott, de később az utasítások száma meghaladta az x86-ost is. Napjainkig 9 különböző ARM ISA verziót hoztak ki. Az első két verzióban még csak 26 bites címteret használt. Az ARMv3 volt az első olyan ISA verzió, ami már 32 bites címteret használt, ez tekinthető az ARM kiindulópontjának. Az ARM ISA jellemzői (korai változatok):

- 32 bites RISC architektúra
- 32 bites fixpontos vagy logikai adatokkal képes dolgozni
- 16 regisztert tartalmaz (13 általános + 3 dedikált)

A fejlődés négy fő irányban történt:

- számítási képességek növelése
- programkód méretének csökkentése
- bytekód végrehajtási sebesség növelése
- biztonság fokozása

A korai ARM rendszerek egyszerű, beágyazott alkalmazásokhoz készültek, ahol fontos volt a kód mérete és a bytekód végrehajtás sebessége is kritikus. A 2000-es években az ARM utat talált az alkalmazói processzorokba (mobilok), ahol sokkal általánosabb feladatokat kellett végrehajtania. Megjelent tehát a grafika és így a lebegőpontos számításokra való igény, megjelentek a SIMD utasítások is a számítási teljesítmény növelése érdekében. Fontossá vált a biztonság is. A v7-től kezdve a korábban fontosnak tartott szempontok értelmüket vesztették.

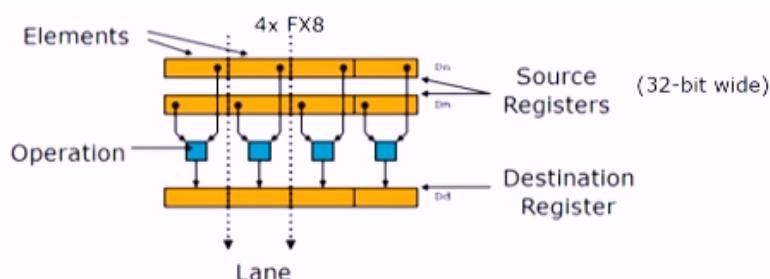
## 8.4. Számítási képességek fejlődése

Az ARMv3 egy egyszerű, alap architektúra, amit a fejlődés során kiterjesztettek. Egy utasításkészlet kiterjesztéséhez két dologra van szükség: egy regiszterkészlet definiálására és a regisztereken végrehajtható utasítások meghatározására. Az ARN fejlesztése három fő logikai lépésből állt:

- a műveletek kiterjesztése fixpontos SIMD utasításokkal (2002, ARMv6)
- másodlagos regiszterkészlet bevezetése skalár lebegőpontos, SIMD lebegőpontos és FX/FP lebegőpontos műveletekhez (2000, ARMv5)
- SVE regiszterkészlet (nagyon széles, 2KB) bevezetése, erre alapozva utasításkészlet kiegészítése (2016, ARMv8.2)

### 8.4.1. Fixpontos SIMD kiterjesztés

Ezzel 32 bit szélesen lehetett végrehajtani 8 vagy 16 bites műveleteket (8.2).



8.2. ábra. Fixpontos SIMD utasítások végrehajtása

Az ezzel elért teljesítmény növekedés csekély volt, ezért később kiterjesztették a NEON bővítménnyel.

Az ARMv8 ISA-jának két végrehajtási módja volt: 32 és 64 bites. Az utóbbihoz kiterjesztették a regiszterkészletet, a 32 bites működés a kompatibilitást szolgálta.

#### 8.4.2. Lebegőpontos műveletvégzés

A NEON kiterjesztés a multimédiás alkalmazások gyorsítását szolgálta. Lehetővé tette, hogy 64 vagy 128 bites hosszú adatokon történjen a műveletvégzés.

#### 8.4.3. SVE regiszterkészletre alapuló SIMD kiterjesztések

Az SVE-t (Scalable Vector Extension) 2016-ban mutatták be, az ARMv8.2 alapú ISA-ban. Csak a 64 bites verzióban volt aktív. Alapvető célja a HPC (High Performance Computing), tehát a tudományos jellegű, képfeldolgozó és média alkalmazások támogatása. FX és FP műveleteket is támogat. A regiszterkészlet nem volt rögzítve, hanem 128-bittől egészen 2 Kb hosszú regiszterek használhatók voltak, tehát többféle fixpontos és lebegőpontos adattípust is támogatott. Ezeket hívták Z regisztereknek.

Legfontosabb jellemzője, hogy a programozásban nem kell figyelembe venni a hardveres regiszterek hosszát. Így a programozóknak könnyebb kódot írnia, mivel eltérő végrehajtóegység méretű processzorokra ugyanazt a kódot tudja használni. A megírt binárisok hordozhatóbbak és automatikus vektorizáció lehetséges a compilerekben.

A megvalósításhoz predikátum regisztereket vezettek be, amiket maszkokként használtak. Ez megmutatja, hogy az adat melyik része aktív és melyik nem. 16 predikátum regiszter áll rendelkezésre. A predikátum regisztereket a programkódban be kell állítani.

#### 8.4.4. SVE 2

Az előző részben leírt SVE-t szinte nem használták. Később bemutatták a továbbfejlesztett változatát, az SVE2-t (ARMv9, 2021). A cél a HPC helyett egy széleskörű alkalmazás palettát támogatni és a NEON-t kiváltani.

ARM ISA		ISA extension	The SVE register set based SIMD extensions			
Name	Year		Available register set	Scalar data types	Vector data types	Instruction subset
Armv8.2	2016	SVE	SVE register set 32 registers each (1...16)x128-bit wide	---	FX 8/16/32/64/128 FP 16/32/64 (1...16)x128-bit wide SIMD data (FMA available)	Small basic instruction subset aiming at HPC workloads
Armv9	2021	SVE2				Vastly enhanced instruction subset aiming at general-purpose workloads

8.3. ábra. Az SVE és SVE2 összehasonlítása

Az SVE-t csak a Fujitsu A64FX processzora implementálta. Ennek oka, hogy az eredeti SVE nagyrészt HPC feladatokra volt kitalálva, de a mobilok nem futtatnak ilyen alkalmazásokat. Ezzel szemben az SVE2 elsősorban a szervereket célozza meg, ahol gyakoriak az ilyen feladatok.

## 8.5. Az ARM CPU-k áttekintése

Az ARM CPU-k csoportosítása:

- Korai ARM CPU-k
  - ARMv1 és v2 (1985-1990): 26 bites címbuszú rendszerek, nem tárgyaljuk őket. Pl.: ARM1-ARM3
  - ARMv3-v6 (1991-2004): 32 bites rendszerek. Pl.: ARM6xx-ARM11xx
- ARM Cortex CPU-k (ARMv7-v9): 2004-től, ma is ezek az uralkodók. 8.0-tól 64 bites
- ARM Neoverse CPU-k (ARMv9): 2021-től, szerverekhez készültek, teljesen új család

### 8.5.1. Cortex CPU-k

A Cortex CPU-k a jelenlegi mobil világ alapját alkotják. 2004-ben vezették be, elsőként a Cortex-M3 beágyazott processzorban.

### 8.5.2. Cortex profilok

A Cortex családot 3 különböző profilra osztották, felhasználás szerint:

- Cortex-A: alkalmazások, a mai mobil processzorok alapja
- Cortex-R: kritikus, valós idejű rendszerek
- Cortex-M: mikrokontroller, nem annyira kritikus rendszerek

#### Cortex-A profil

Két utasításkészletet támogat: az A32-t és a T32-t. A T32 egy 16 bites utasításokat tartalmazó részhalmaza az A32-nek. Az ARMv8-A verzióval megjelent a 64 bites kiterjesztés, a korábbi 32 bites alkalmazásokkal kompatibilis módon. A 32 bites alkalmazásokat AArch32 végrehajtási módban, A32 utasításkészlettel futtatja, a 64 biteseket pedig AArch64 módban, A64 utasításkészlettel. Ezzel együtt az AArch64 módban megszüntették a Thumb 16 bites utasításkészlet támogatását. Az ARMv9-el a 32 bites utasításokat sem támogatták tovább.

#### Cortex-R profil

Beágyazott rendszerekhez készült, szintén az A32 és T32 utasításkészleteket támogatja.

#### Cortex-M profil

Mélyen beágyazott rendszerekhez, mikrokontrollerekhez készült (pl. autóipar). Ez csak a T32 16 bites utasításokat támogatja, hogy minél kisebb legyen a program.

## **SecurCore profil**

2007-ben a családot kiegészítették a SecurCore profillal, smart card és egyéb biztonsági alkalmazásokra kifejlesztve.

A továbbiakban csak az A profilú processzorokkal foglalkozunk.

## **8.6. A Cortex-A profilú CPU-k fejlődése (v7 és v8)**

2004-ben jelentek meg az első Cortex-A CPU-k, 2011-től támogatták a big.LITTLE felépítést (ARMv7-A ISA). Ezt követték az ARMv8.0-A-t megvalósító, 64 bites CPU-k (2012-ben jelentették be). A következő generáció (ARMv8.2-A, 2017) már támogatta DynamIQ magcsoportokat, ami a korábbi big.LITTLE magcsoportokat váltotta. 2021-ben megjelent az ARMv9-A, ahol jelentősen továbbfejlesztették a DynamIQ-t.

### **8.6.1. Az ARMv7-8 CPU-k fejlődése**

Három teljesítmény osztályt határoztak meg:

- nagy
- közepes
- alacsony teljesítményű, kis fogyasztású

Az eltérő teljesítményű rendszerekhez eltérő lapkaméretek is tartoznak. A kis teljesítményű rendszerek viszonylag kis helyen megvalósíthatók, míg a nagy teljesítményűekhez több helyre van szükség.

A Cortex-A CPU-k fejlődését négy szempont szerint vizsgáljuk:

- szóhossz
- L2 cache jelenléte
- egy vagy több magos-e a CPU
- big.LITTLE támogatás

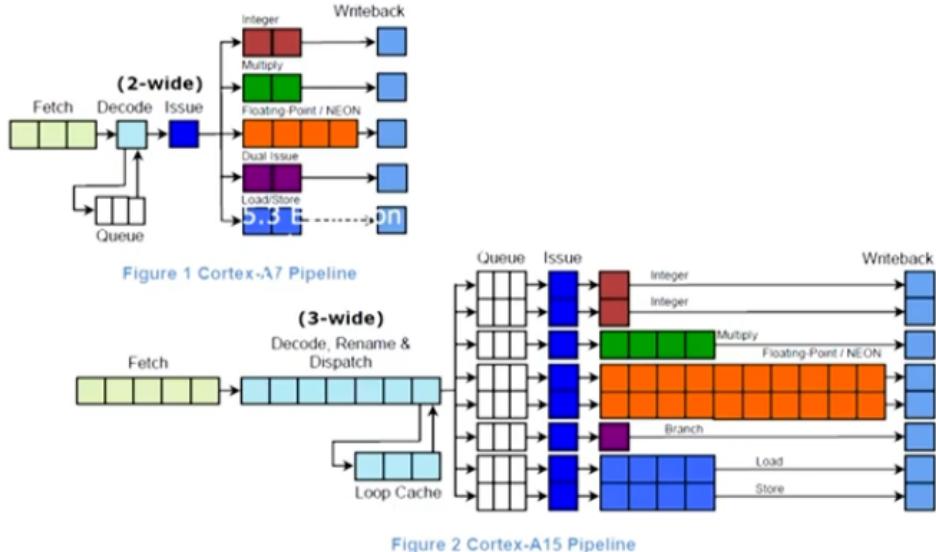
Szóhossz tekintetében a v7 rendszerek csak 32 bitet, míg a v8.0 CPU-k 32 és 64 bitet is támogatnak. A Cortex-A5 esetében még L2 cache se volt, később (A7-A9) tervezési opcióból jelen volt az L2 cache, majd az A15-től már kötelező tartozék volt. A magok számát tekintve korábban csak egymagos rendszerek léteztek, az A9 viszont már létezett többmagos változatban is (Cortex-A9 MPCore). A többmagos rendszerek big.LITTLE felépítést követtek, később az egymagos verziót elhagyták. Ma már minden ARM CPU többmagos.

### **8.6.2. Az ARM MPCore kiegészítés**

2004-ben, a korai ARM processzorokkal jelent meg az MPCore tag, az ezzel ellátott ARM 11-et multiprocesszoros rendszerként árulták. Ennek ellenére ez csak egy egyszerű, többmagos processzor volt. Később az ARM Cortex-A9 modellből is létezett MPCore változat, ezt viszont már több magos rendszerként hirdették. Az MPCore nélküli változatok egymagosak voltak. A későbbiekben elhagyták az MPCore elnevezést, mivel minden ARM Cortex-A CPU többmagos volt.

### 8.6.3. A big.LITTLE technológia

2011-ben hozta nyilvánosságra az ARM, megvalósítása a 32 bites rendszereknél kezdődött (A7, A15). Ma már minden mobil erre alapoz. A LITTLE mag 2 széles végrehajtó egységet tartalmaz, a big pedig 3 széleset.



8.4. ábra. big és LITTLE magok a Cortex-A7 processzorban

Egy egyszerű big.LITTLE rendszer tartalmaz két big és két LITTLE magot (A15 és A7), amiket egy cache koherens hálózat köt össze (Cache Coherent Interconnect - CCI-400). Alapesetben a LITTLE mag működik, ha nő a teljesítmény igény, átkapcsol a big magokra.

### 8.6.4. Cortex-A processzorok teljesítmény növekedése

Az ARM processzorok hatékonysága 2007-től 2016-ig a duplájára nőtt (IPC alapján). A tényleges teljesítmény ugyanezen az időtávon 5-szörösére emelkedett, a fogyasztás pedig mindenkorban a felére csökkent.

Az Intel processzorok ezalatt viszont kisebb hatékonyság növekedést értek el.

### 8.6.5. Az ARM Cortex-A processzorok elterjedése

Az ARM Cortex-A8 és A9 processzorok jelentek meg először széleskörben a mobil eszközöknél.

## 8.7. A Cortex-A profilú CPU-k fejlődése (v8.2)

Az ARMs v8.2 egy jelentős előrelépést jelentett az ARM processzor fejlődésében. 2017-ben jelentek meg először, a Cortex-A55 és A75 processzorokkal (egyidőben az AMD Zenkel). Az 5-ös a nagy hatékonyságú, a 7-es pedig a nagy teljesítményű modell.

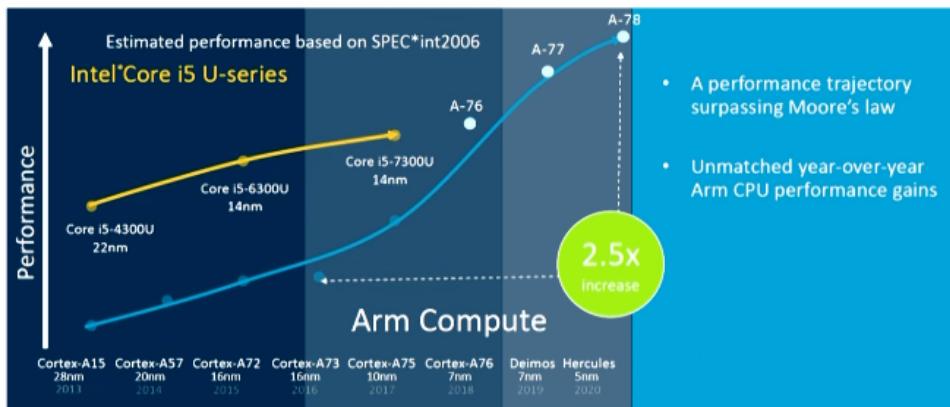
A modellcsalád kiegészült a Cortex-X1 processzorral, ami egy nagyon nagy teljesítményű processzor.

### 8.7.1. Különbségek az ARMv8-hoz képest

- a magok frontendje szélesedett
- a futószalagok száma növekedett

### 8.7.2. Fejlesztési roadmap

Az ARM 2017-ben egy fejlesztési roadmapet tett közzé, amikor bejelentették a Deimos és Hercules módelleket (később az A-77 és A-78 nevet kapták). Ekkor az ARM szerint a processzoraik teljesítményben felül fogják műlni az Intel-t.



8.5. ábra. Az ARM által jóslott fejlődés

### 8.7.3. A Cortex-X1 processzor

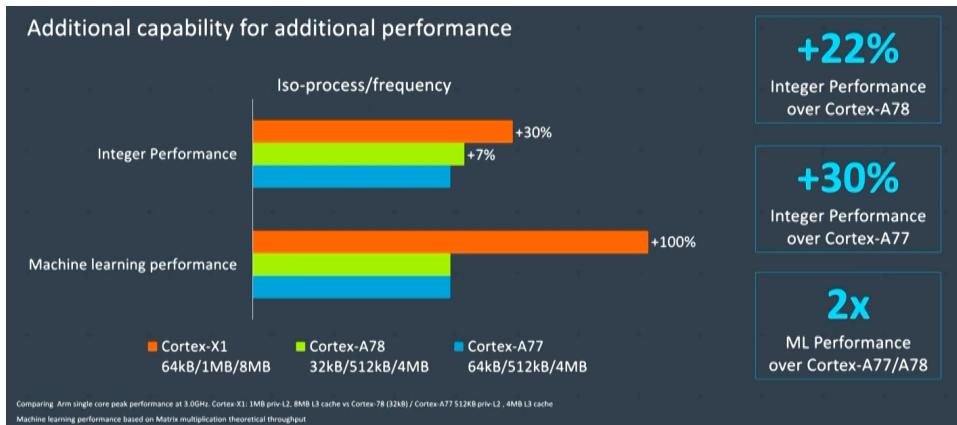
A Cortex-X1 processzort 2020-ban jelentették be, a Cortex-A78-al közös alapokra épül. Tervezése az Austin-i központban történt. A bejelentése egy új program keretében történt, ez volt a Cortex-X Custom, ami a Built on ARM Cortex Technology evolúciója. A program keretében lehetőség van a partnerrel közösen testreszabni a készülő processzorokat, mint pl. meghatározni a ROB méretét. A készülő processzorok megtartják a Cortex nevet és az ARM márkát.

#### Különbségek az A78-tól

- az A78 egy kiegyszűlyozott fejlesztés, az X1 viszont a teljesítményt helyezi előtérbe a fogyasztással szemben
- az A78 4 utasítású frontenddel szemben 5 utasításúval rendelkezik
- 2 helyett 4 SIMD futószalag van
- a cache-ek mérete is megnőtt
- a ROB méret is növekedett

#### Eredmények

A Cortex-X1-el nagyjából 20-30%-os teljesítmény növekedést tudtak elérni.



8.6. ábra. Az ARM Cortex-X1 teljesítmény növekedése

## 8.8. Az ARMv9 ISA

Az ARM 2021 márciusában jelentett be, az első processzorok 1-2 hónappal később jöttek ki. Az első ilyen processzor a szerverekbe szánt Neoverse volt.

Az ARM célja az ARMv9-el a teljes felhasználási terület lefedése, a szenzoroktól a szerverekig. Az ARM architektúra 9. verziója 10 évvel a 8-as után jelent meg. Az új verzió visszafelé kompatibilis a korábbi architektúrákkal.

### 8.8.1. Főbb fejlesztési területek

- vektor és digitális jelfeldolgozás (SVE2)
- mély tanulás
- többszásas programozás hatékonysága
- biztonság

### 8.8.2. Vektor és digitális jelfeldolgozás (SVE2)

Az ARM már korábban is támogatott vektoros kiterjesztéket, ez volt az SVE (Scalable Vector Extension), ami az ARMv8.2 utasításkészlet egy opcionális kiterjesztése volt. Ez főleg a tudományos alkalmazásokat támogatott, általános felhasználásra nem igazán volt alkalmas.

Ennek a hiányosságait küszöböli ki az SVE2, ami a teljes alkalmazói spektrumot lefedi és így a célja a Neon leváltása.

#### Az SVE és SVE2 előnyei

- lehetővé tesznek változó hosszú, kvantált vektorméreteket
- lehetővé teszik a szóhossz független programozást - ennek segítségével a programozó ugyanazokat az utasításokat tudja használni a vektorhossztól függetlenül. Ezt predikátum regiszterekkel éri el az architektúra, ahol a bitek kijelölök az aktív adatmezőket. Következmény, hogy a kód más végrehajtó egységen is futtatható.

### **8.8.3. ML**

A gépi tanulást az élet minden területén használjuk, ennek egy lényeges eleme a neurális hálók használata. Az ARMv9 ezért támogatja a fél pontosságú lebegőpontos számformátumot és a Google által kifejlesztett BFloat típusokat. Ezen kívül támogatja a skalárszorzat számításokat is.

### **8.8.4. Többszálú programozás támogatása**

Többszálú programoknál fennállhatnak versenyhelyzetek (race condition), ezek megoldására két mehanizmus lehetséges:

- lockok - az adott memóriaterülethez egyszerre csak egy szál férhet hozzá
- tranzakciós memória - a rendszer ellenőrzi és javítja a konfliktusokat, hasonlóan az elágazásbecslés visszalépéséhez

A tranzakciós memóriát már más gyártók korábban is használták, az ARM is vezette a v9-el.

### **8.8.5. Biztonság**

A biztonság növelése érdekében vezetették a megbízható számítási architektúrát és a memória jelzőbitek (memory tagging) használatát.

## **8.9. Az ARMv9 mikroarchitektúra**

Az ARM 2011-ben vezette be a big.LITTLE architektúrákat, ami két magcsoportot tartalmazott, klaszterenként max. 4 maggal. Ezt váltotta a v8.2-ben a DynamIQ dinamikus mag klaszter, ahol már max. 8 kis mag lehet egy klaszterben, valamint a magokat tetszőlegesen lehet kombinálni, azzal a megkötéssel, hogy max. 4 nagy mag lehet. A big.LITTLE-ellentében a DynamIQ architektúrában már saját L2 cache-t kaptak a magok.

Egy DynamIQ klaszter két részből áll:

- magok
- DSU - DynamIQ Shared Unit

### **8.9.1. A DSU**

- viszonylag nagy L3 cache-t tartalmaz
- a rendszerhez 2x128 bites vagy 1x256 bites buszon kapcsolódik
- a disszipációkezelés magonként történik

### **8.9.2. A DynamIQ klaszter fejlődése**

Az ARMv9 processzoroknál fejlesztették a DynamIQ klasztert is, a következő változások történtek:

- 3 magtípus lehet használni
- szeletelt L3 cache és egyéb DSU egységek
- memória címkézés (Memory Tagging Extension) támogatása
- szélesebb, skálázható csatlakoztatás a DSU-nál

Az új magtípusok:

- nagy teljesítményű mag
- nagy hatékonyságú big mag
- nagy hatékonyságú LITTLE mag

Tehát megjelent egy még gyorsabb magtípus. Ezzel az ARM több felhasználási területet akar lefedni, pl. a laptopokat.

A DSU egyik nagyon fontos újítása, hanem egy szeletelt megvalósítás jelenik a korábbi monolit felépítés helyett. A szeleteket egy gyűrű kapcsolat fogja össze, két darab egyirányú gyűrűvel. Míg a monolitikus egység egy belépései ponttal rendelkezik, a szeletelt cache-nél a szeletek önálló életet élnek, minden szelet önállóan tud kommunikálni. Ennek előnye, hogy a szeletek párhuzamosan tudnak működni, ami gyorsítja a teljesítményt. További előny, hogy a szeletek kisebbek, így kisebb a hozzáférési idő. Hátrány viszont, hogy ha a keresett adat egy távolabbi szeletben van, tovább tart az elérése. A rendszer tehát NUCA (Non-Uniform Cache Architecture) felépítést használ.

A szeletelésen kívül megnövelték még az L3 cache méretét.

### 8.9.3. Memory Tagging Extension

A memória gyakran biztonsági problémák forrása. Az ebből adódó sérülékenységek elkerülésére az MTE bevezet zárákat és kulcsokat. A zárak hozzá vannak rendelve a memóriához, 16 byte-onként egy 4 bites zárat definiálnak. A hozzáférés során a virtuális címekhez kulcsokat rendelnek. A kulcs csak akkor férhet hozzá egy területhez, ha megegyezik a zárral.

A kulcsok megoldásához felhasználták, hogy az ARMv8-A nem használja a pointerek legfelső byte-ját, ezt használják a kulcsra.

### 8.9.4. Hozzáférési szélesség növelése

Korábban 2x128 vagy 1x256 bitet használtak a busz interfészhez, a 9-es verzió viszont megnégyszerezi ezt a hozzáférési szélességet.

## 8.10. ARMv9-A-t megvalósító processzorok

A Cortex családban három v9-es processzor jelent meg: Cortex-A510, Cortex-A710 és Cortex-X2. Az előző generációkhöz képest 10-30%-os teljesítménynövekedést, 20-30%-os hatékonyság növekedést és 2x gépi tanulási teljesítményt tudnak.

A legnagyobb teljesítményű ARMv9-es processzor az ARM Cortex X2. Az ARM Cortex-A710 és az X2 egymagos processzor, az A510 pedig kétmagos. Az ARMv9-A alapú magklaszterek ezeket az egységeket tudják felhasználni.

## 9. fejezet

# Mobil processzorok architektúrája

### 9.1. Bevezetés

A modern processzorok sok különálló egységet tartalmaznak, ezek közül a CPU-val foglalkozunk. Mobil processzorok alatt a mobiltelefonok és a táblagépek processzorait értjük.

Csak kevés gyártó tervez mobil processzorokat és a legtöbbjük nem rendelkezik gyártási kapacitással, kivéve a Samsungot. A többi gyártó processzorait jellemzően a TSMC vagy a Samsung gyártja le.

A mobil processzorok többsége ARM alapú, de léteznek x86 alapú mobil processzorok is. x86 alapú processzorokkal az Intel és az AMD próbálkozott, sikertelenül (2016-ban meg is szüntették a gyártást).

### 9.2. ARM licenszek

Az ARM processzorok gyártóinak két lehetősége van:

- Cortex licensz felhasználása
- Architektúra licensz felhasználása

Cortex licensz esetén a vásárló egy komplett mikroarchitektúra tervet kap meg, konfigurálható opciókkal. Az architektúra licensz viszont csak arra jogosítja fel a vásárlót, hogy az utasításkészletet használja.

### 9.3. Az ARM alapú mobil processzorok fejlődése

Az ARM ISA fejlődésével együtt az ARM mobil processzorok is fejlődtek. Ennek az egyik fontos állomása volt a v7-ről v8-ra átállás, ami behozta a 64 bites támogatást. Az áttérés során a cégek előtt két út állt:

- egyesek megtartották a hagyományos szimmetrikus többmagos felépítést
- mások egyúttal áttértek a big.LITTLE architektúrára is

Az Apple és a MediaTek szimmetrikus megvalósítást alkalmazott, a Samsung és a Huawei pedig big.LITTLE architektúrát kezdett használni. A 64 bites áttérés tehát nagyjából egyszerre történt a big.LITTLE átállással, 2013 és 2016 között.

Nem sokkal ezután történt az átállás az ARMv8.2-re (2018), ami a dinamikus magklaszterek használatával jelentős teljesítmény növekedést eredményezett. Ezzel egyidőben jelentek meg az ARM szerverprocesszorok is.

## 9.4. Az Intel Atom és az AMD Cat család

Az Intel és AMD hagyományos processzorai a nagy fogyasztás miatt nem voltak alkalmasak a mobil felhasználásra, ezért az Intel 2008-ban, az AMD pedig 2011-ben új processzorokkal állt elő. Az AMD piacra lépése nagyon későinek számított.

Az Intel Atom processzora a hagyományos processzorokkal ellentétben 2 és 3 széles volt, ennek oka az alacsonyabb fogyasztás.

Ezek ellenére egyik cég se tudott érdemben betörni a mobil piacra, ezért 2015-ben befejezték ezeket a fejlesztéseket.

## 9.5. Windows táblagépekre fejlesztett Core 2 processzorok

2013-tól a Microsoft megjelent a Surface táblagépekkel, amik 2 az 1-ben gépek voltak. Intel processzorokkal szerelték őket, először 2, aztán 4 maggal. A táblagépek között nagyjából 10%-os részesedést értek el.

Az elmúlt néhány évben a Windows tabletok is megjelentek Qualcomm ARM processzorokkal. A céljuk olyan 2 az 1-ben gépek megalkotása, amik nagyon hosszú üzemiidővel rendelkeznek (Always on, always connected). Fő problémájuk a szoftver kompatibilitás.

## 9.6. ARM processzorok az Apple Mac eszközeiben

Az Apple 2006-tól az Inteltől vásárolta a processzorait. 2020-ban viszont bejelentették, hogy saját processzort fog fejleszteni a MacOS alá. Az új processzor, az Apple M1 jó teljesítménnyel rendelkezik.

## 9.7. Mobil processzor architektúrák fejlődése

A fejlődésnél négy fő szakaszt különböztetünk meg:

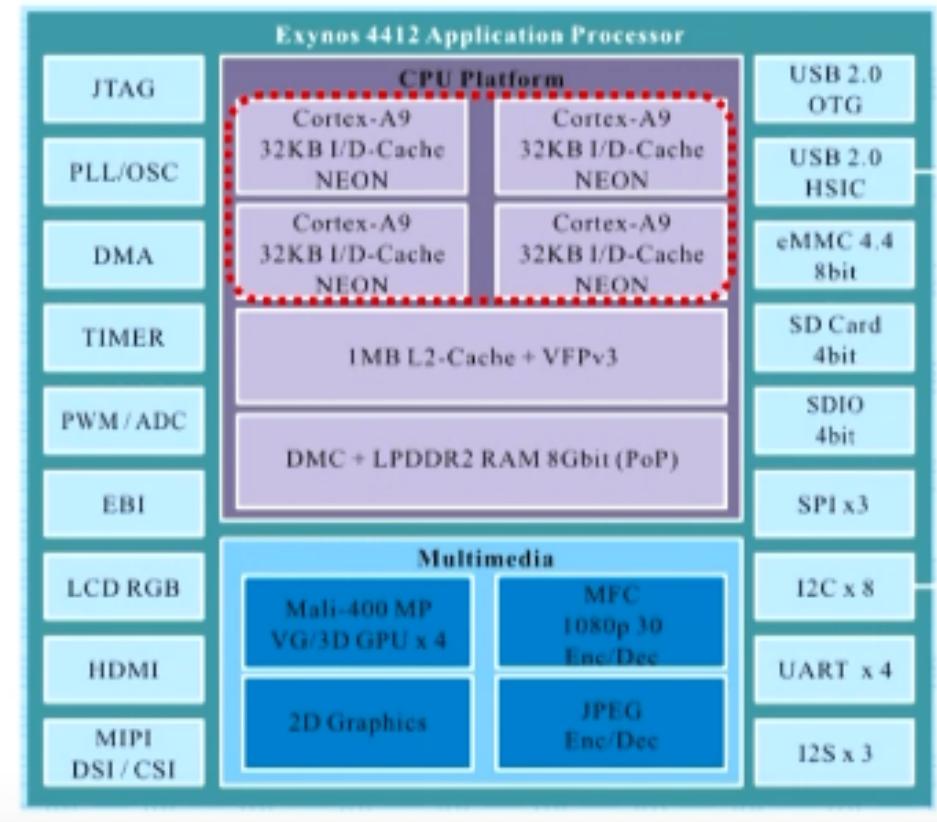
- egymagos processzorok
- szimmetrikus többmagos processzorok
- big.LITTLE processzorok
- DynamIQ klaszter alapú processzorok

Az egymagos processzorok használata kb. 2003-tól 2010-ig tartott, általában 32 bitesek voltak. A többmagos processzoroknál először szimmetrikus felépítést alkalmaztak, 2011-2013-ban, amik szintén 32 bitesek voltak. A big.LITTLE architektúra 2013-2016-ig tartott, itt megtörtén a 64 bites átállás. DynamIQ processzorokat 2018-tól gyártanak.

Az egymagos processzorokat nem tárgyaljuk.

### 9.7.1. Szimmetrikus többmagos processzorok

A desktop processzorok többmagossá válása után kb. 6 évvel, 2011-ben jelentek meg a mobilknál is a többmagos felépítések. Cél a teljesítmény növelése. Az egyik első ilyen rendszer a Samsung Exynos 4412 volt.



9.1. ábra. A Samsung Exynos 4412 felépítése

A szimmetrikus többmagos processzorok fő fejlesztési iránya a magok számának növelése volt, a kezdeti 2-ről 4-re, majd 8-ra emelkedett.

Az általános felépítés kettő vagy négy magos magklasztereket jelentett, a klaszteren belül egy közös L2 cache-el. 8 magos rendszereket 2x4 magos klaszterrel készítettek.

### 9.7.2. big.LITTLE architektúrák

A kifejesztésük fő motivációja a fogyasztás csökkentése. Az ötlet onnan eredt, hogy az okostelefon használata során nagyon gyakran változik a teljesítmény igény. Ezért kétféle magot hoztak létre, kis fogyasztású, kis teljesítményű és nagyobb fogyasztású de nagyobb teljesítményű.

#### Feladatok elosztása

Az így létrejövő heterogén többmagos rendszerekben a feladatokat többféle módon is fel lehet osztani a magok között:

- mester-szolga elv

- feladattovábbítás a dedikált végrehajtók felé
- más típusú processzor felhasználása

A mester-szolga elvnél van egy megkülönböztetett mester mag és egy vagy több szolga mag. Ekkor a mester mag látja el a feladatok kiosztását is. Az IBM, Sony és a Toshiba próbálkozott ilyen elvű processzorral (Cell processzor), de a bonyolult programozás miatt nem volt túl működőképes.

A második esetben a processzorban dedikált végrehajtóegységek vannak, mint pl. GPU. minden utasítás ahhoz az egységekhez kerül, ami a legalkalmasabb a végrehajtására.

Az utolsó működést tipikusan big.LITTLE architektúráknál használják, ekkor különböző típusú CPU-k működnek együtt.

### **big.LITTLE alapelvek**

A big.LITTLE processzorokban kettő vagy több magklaszter dolgozik, amik architekturálisan megegyező magokat tartalmaznak. A magklaszterek cache koherens módon vannak összekötve. Ehhez kell egy szoftver, ami a klasztereket vezérli.

### **Működési elv**

A big.LITTLE technológia különböző módokon implementálható, mi az exkluzív klaszter kapcsolást vizsgáljuk a működési elv leírásánál. Feltételezzük még, hogy egy klaszter minden magja azonos munkaponton dolgozik.

- egy kernel rutin figyeli a teljesítmény igényt és kiválasztja a megfelelő munkapontot
- ha a teljesítmény igény kicsi, a kis klaszter van használatban
- ha a teljesítmény igény megnő, klaszter átkapcsolás történik a nagy magokra
- ha újra elég a kisebb teljesítmény, a klaszter visszavált a kis magokra

Egyszerre csak egy klaszter működhet, ezért ez az exkluzív klaszter kapcsolási rendszer.

### **Megvalósítás**

A technológia hardverkötélezetére:

- a kis és nagy klasztereknek architekturálisan azonosnak kell lennie
- a gyors átkapcsoláshoz szükséges egy cache koherens megoldás
- a megengedett magszám klaszterenként általában 4
- megfelelő szoftvertámogatás szükséges, amit a processzorgyártó ad át a gyártóknak, kernel kiegészítések formájában

A big.LITTLE architektúrákat három szempont szerint vizsgáljuk:

- klaszterek és magok száma
- ütemezési megoldások

- feszültség és frekvencia ellátás

Mag klaszterek szerint két vagy három klaszteres rendszereket különböztetünk meg. Az egyes klaszterekben 1, 2, 3 vagy 4 mag lehet.

Az ütemezési megoldások vizsgálatánál egy 2 klaszteres processzort tekintünk. Az ütemező működhet klaszter vagy mag alapon. Mindkét esetben kétfajta implementáció lehet: exkluzív és inkluzív. Exkluzívnál vagy egyik, vagy másik klaszter van hozzárendelve egy adott feladathoz. Inkluzívnál megengedett, hogy a két klaszter egyszerre működjön nagy teljesítmény igény esetén. Általában exkluzív implementációt használnak, mivel az inkluzívát nehezebb implementálni és viszonylag kis gyorsulást eredményez.

A mag alapú ütemezés is történhet exkluzív vagy inkluzív módon. Exkluzív mag allokációnál a magokat kis-nagy párokba szervezik. Az ütemező a terheléstől függően a nagy vagy kis magot választja ki. Inkluzívnál viszont minden mag külön-külön aktiválható, ezért ez egy globális ütemező. Ma ez a legelterjedtebb.

A felhasználói igény döntően a frekvenciát dönti el, ami meghatározza a szükséges feszültséget. A szabályzás üzemeltethet minden magot azonos feszültségen és frekvencián, szabályozhatja magonként a frekvenciát, közös feszültséget tartva, vagy magonként különböző frekvencia és feszültség beállításokat alkalmazhat. A leghatékonyabb az utolsó megoldás.

### 9.7.3. DynamIQ magklaszterek

Az ARM 2013-ban kezdte el fejleszteni a technológiát, 2018-ban kezdték használni. Ez jelenti a big.LITTLE architektúrák utáni lépcsőfokát a fejlődésnek.

A dinamikus magklaszter két különböző magot képes tartalmazni: kis mag és nagy mag. Különbség a big.LITTLE klaszterekhez képest, hogy a magokat közel tetszőlegesen lehet párosítani.

#### Energy Aware Scheduling

A fogyasztást egy intelligens fogyasztáscsökkentő eljárás bevezetésével javította, ez az EAS - Energy Aware Scheduling. Ez előtt a Linux kernel a CFS-t, azaz a Completely Fair Scheduler használta, ami elsőről teljesítményre optimalizált. Hárnya, hogy heterogén rendszerekben nem működik elég hatékonyan. Az EAS ezzel szemben figyelembe veszi a big.LITTLE felépítést, így elérve a lehető legalacsonyabb fogyasztást.

#### DynamIQ klaszterek jellemzői

- ARMv8.2-t támogatnak
- két típusú magot tartalmazhatnak a klaszterek
- magonkénti L2 cache-el rendelkezik - ez lényegesen jobb megoldás, mint a közös L2 cache
- megjelenik a harmadik szintű gyorsítótár, a magok közös L3 cache-el rendelkeznek
- a klaszter egy új egységet is tartalmaz, ez a DSU (Dynamic Shared Unit): ebben van az L3 cache és egy snoop filter, ami a cache koherencia fenntartását teszi hatékonyabbá. Következmény, hogy alacsonyabb késleltetéssel érhetők el a gyorsítótárak.
- a magok partícionálhatók: négy partíció hozható létre, amik magokat, külső gyorsítókat és az L3 cache bizonyos részeit tartalmazzák. A partíciókat feladatok szerint lehet kialakítani.
- finomabb feszültség-frekvencia szabályozás, a különböző partíciók eltérő frekvencián és feszültségen működhetnek

## 9.8. Magok szélességének fejlődése

A processzormagok két részre bonthatók: frontend és backend. A frontend több utasítást dolgoz fel szekvenciálisan és küldi ki a backend számára. A backendben egyedi utasítások kerülnek feldolgozásra a megfelelő futószalagokon. A mag szélessége a mag legszűkebb keresztmetszetét jelöli, ami legtöbbször a dekódoló szélességének felel meg. Teljesítmény szempontjából minél szélesebb egy mag, annál gyorsabb a processzor, a fogyasztás viszont fordítva működik. Fontos szempont még, hogy a mag tud-e többszálas működést. A szálak gyorsító hatása az adott felhasználástól függ.

Az első mobil rendszerekben (2000-es évek eleje) 2-3 széles processzorok dolgoztak. Ez nem sokat változott a 64 bites átállással, mivel a kisebb teljesítményű rendszerek maradtak 2 szélesek, a nagyobb teljesítményűek pedig 3 szélesek. Ezután az Intel és az AMD megjelentek a saját mobil processzoraikkal, amik 2 szélesek voltak, viszont nem voltak versenyképesek.

Az ARMv8 és v8.2 ISA-ra épülő processzoroknál a magasabb teljesítmény érdekében áttértek 4-es és 5-ös szélességre. Ennek eredménye, hogy a kibocsátási és a kiküldési ráta is jelentősen megnőtt. A szélesség növeléséhez növelni kellett az erőforrásokat, így a ROB méretét is. Ezzel együtt megjelent a MOP cache, ami a már dekódolt utasításokat tudja tárolni és újrahasználni, pl. elágazásbecslés visszatörlésnél vagy ciklusnál.

Az Apple nagyon hamar, az A7 processzorukban (2013) áttért a 6 széles magokra. Ezzel teljesítményben minden más processzort leelőzött, amit később tartott is. Később az A11 és A12X processzorokban 7-es, majd az A13, A14 és M1 processzorokban 8-as szélességű magokat alkalmaztak.

A Samsung szintén nagy lépésekkel fejlődött a Mongoose processzorával, 6-os szélességet értek el, a 8 széles processzoruk fejlesztését viszont leállították.

Az AMD hagyományos desktop architektúrái ezzel szemben csak 4 szélességet fejlődtek (Bulldozer család), az Intelek viszont 6-ig (Alder Lake). Az AMD ezzel sokat veszített a piacból. Tehát a mobil processzorok szélesebbek a hagyományos rendszerknél. Ennek oka, hogy a hagyományos rendszerek CISC architektúrák, ahol változó hosszúak az utasítások. A RISC-ek ezzel szemben fix utasításhosszal rendelkeznek, ezeket pedig sokkal könnyebb dekódolni.

### 9.8.1. A Samsung Mongoose családjának fejlődése

A Samsung 2010-ben alapított egy processzor fejlesztő központot Austinban, ahol a Mongoose családot fejlesztették. Első megjelent processzoruk volt az ARMv8 alapú M1, 2016-ban. Később áttértek az ARMv8.2-re, majd az M6-al leálltak a fejlesztéssel. Az elkészült processzorokat a Samsung Galaxy telefonokban használták. A több száz alkalmaszt mérnök ellenére a Samsung 2019-ben bezárta a központot, utána pedig publikálták a processzorok fejlődését.

Az első Mongoose processzor a Galaxy S7-ben jelent meg, az utolsó pedig a Galaxy S20-ban (2020). Kiemelt ezek közül az M3-as mag, amivel a Samsung áttért az ARMv8-ra. A 10 éves fejlesztés során a teljesítmény növelését három területen keresték:

- kisebb igényű feladatok végrehajtásának gyorsítása: ezen a területen arra jutottak, hogy az előlehívás és a cache koherencia protokollok optimalizálásával lehet gyorsulást elérni
- közepes igényű feladatok: itt az elágazás kezelés és a cache fejlesztésével tudtak eredményeket elérni
- nagy igényű feladatok: ezeknél főleg a szélesség növelésével és a megfelelő erőforrások biztosításával lehetett gyorsítani

Ezeknek a szempontoknak a figyelembevételével az M6 az M1-hez képest 2,71-szer nagyobb IPC-t (Instruction per Cycle) ért el. Ez évente 20%-os növekedésnek felel meg.

A teljesítményt a programfutások trace-elésével vizsgálták és ezek alapján állapították meg a fejlesztési lehetőségeket.

### **A mikrooperációs cache**

A magok szélességének növelésével szükségessé vált az utasítások gyorsítótárazása, ezért az M5 bevezetett a mikrooperációs cache-t. Ez 384 utasítást tudott tárolni, a használatával a cache-elt utasításoknál megsűrölhető a lehívás és a dekódolás. A többi gyártóhoz képest későn vezették be.

### **Végrehajtó egységek száma**

A szélesség duplájára növelésével a végrehajtó egységek is lépést tartottak, itt is kb. kétszeres növekedés ment végbe.

### **Összegzés**

A fejlesztések ellenére az ARM processzorok gyorsabban fejlődtek, ezért a Samsung is visszatért ezek használatára és felhagyott a saját fejlesztéseivel.