OBUDA UNIVERSITY YEAR OF JUBILEES
ÓE 15 145 630
TRADITION AND INNOVATION

Óbudai Egyetem
Alba Regia Kar
Mérnöki Intézet

Bevezetés a számítógép architektúrákba

Dr. Seebauer Márta
egyetemi docens

seebauer.marta@uni-obuda.hu

A CPU számítási teljesítménye

A processzorok működési sebessége

- technikai oldalról
  - az elektronikus áramkörök működési sebessége
  - a processzor architektúrája
- felhasználói oldalról
  - a feldolgozás időtartama

Egy meghatározott feladat végrehajtására fordított idő

$$T_{feladat} = I\ C_{eff} \cdot T \cdot$$

C_{eff} – gépi utasításonként az órajel ciklusok (átlagos) száma

T - az órajel ciklusok időtartama (helyette a reciproka f=1/T, az órajel frekvencia használatos)

I – a feladatonként végrehajtott utasítások száma

$$T_{feladat} = I\ C_{eff} \cdot /f$$

A CPU számítási teljesítménye az egységnyi idő alatt végrehajtott utasítások száma

$$R=I/T_{feladat} = f/C_{eff}$$

A CISC processzoroknál az I, míg a RISC processzoroknál C és T csökkentésére, illetve f növelésére törekszenek.

A teljesítményértékelés fejlődése

- Kezdetben egy **egyedi művelet**, mint, például, egy összeadás végrehajtási idejének mérése. Mivel a legtöbb utasítás ugyanazon végrehajtási ideig tartott, így egyetlen utasítás végrehajtási idejének ismerete megfelelő értékelést biztosított.
- A számítógépek utasítás-végrehajtási ideje egyre különfélébb lett, így egyetlen utasítás végrehajtási ideje nem volt többé hasznos információ. Annak érdekében, hogy ezeket a különbségeket számításba vegyük, egy **utasítás-mixet** számítottak, megállapítva az utasítások relatív gyakoriságát sok programból általánosítva. A Gibson mix (1970) volt egy korai népszerű utasítás-mix. Minden utasítás végrehajtási idejét megszorozva a hozzá tartozó súllyal megadta a felhasználónak az átlagos utasítás végrehajtási időt. Mivel az utasítás-készletek hasonlóak voltak, így ez pontosabb összehasonlítást eredményezett, mint az idők összeadása.
- Az átlagos végrehajtási időtől csak egy kis lépést jelentett a **MIPS** mint mértékegység megjelenése, mivel az egyik a másiknak az inverze.
- A CPU-k egyre bonyolultabbak lettek, megjelentek a memória-hierarchiák és a futószalag-technikák, nem volt már értelmezve az átlagos utasítás-végrehajtási idő. A következő lépés volt a **benchmark-ok, kernel**-ek és **szintetikus benchmark programok** használata.
- A 70-es és 80-as éveket a szuperszámítógép ipar fellendülése jellemzi, melyek lebegőpontos műveleteket tartalmazó programokat igen gyorsan hajtották végre. Az átlagos műveleti idő és a MIPS nyilvánvalóan alkalmatlan mértékegység volt ezen ipar számára, ezért megalkotta a **MFLOPS** mértékegységet.

A CPU teljesítmény mértékegységei

MIPS - Million instructions per second - egyszerű, jól érthető fogalom (a gyorsabb gépek nagyobb MIPS-szel rendelkeznek) Alapvetően a fixpontos CPU-k értékelésére használható.

A problémák:

- függ az utasításkészlettől, ezért nehezen vethetők össze a különböző utasításkészletű gépek
- ugyanazon a gépen is más programnál más értékű
- a teljesítménnyel fordított arányban is változhat. Mivel a lebegőpontos műveletek lebegőpontos utasításonként több óraciklust igényelnek, magasabb MIPS értéket eredményez, mivel több utasítást kell végrehajtania, és a teljes végrehajtási idő hosszabb lesz.

Csúcs (peak) MIPS - a processzor hány milliót tud végrehajtani a leggyorsabb műveletéből (ilyen például a NOP), tehát nem csinál semmit.

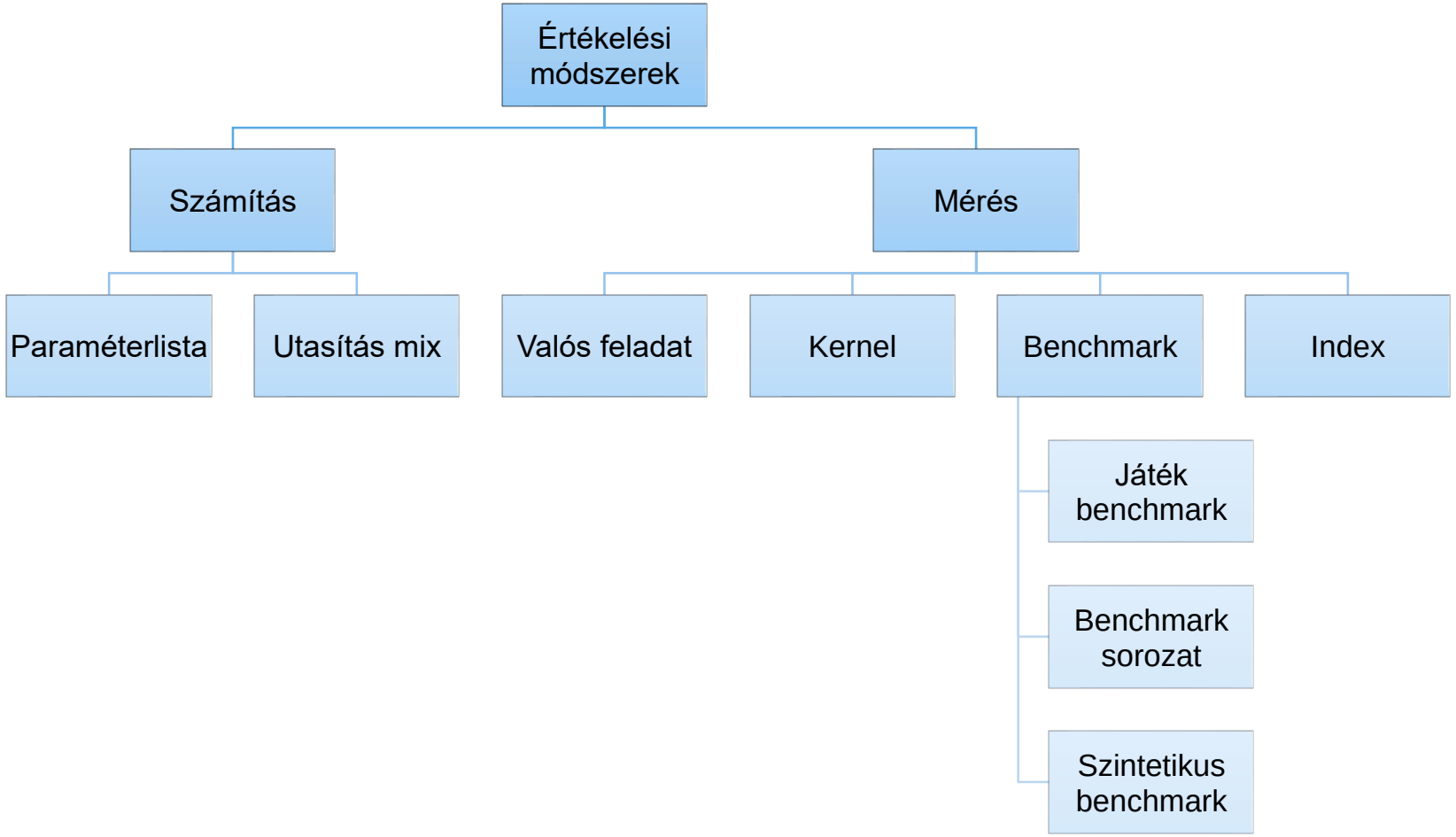
A CPU teljesítmény mértékegységei

MFLOPS - Million floating-point operations per second - Elve, hogy ugyanaz a program különböző gépeken futva különféle számú utasítást hajt végre ugyan, de azért ugyanolyan számú lebegőpontos műveletet végez el. A MFLOPS értéke függ mind a géptől, mind pedig a programtól. Mivel ezen mértékegység szándéka a lebegőpontos műveletek mérése, ezért nem alkalmazható ezen a körön kívül. Műveleteken alapul, nem az utasításokon. Célja a különféle gépek objektív összehasonlítása.

- A MFLOPS az utasítás-készlettől mégsem függetleníthető, mivel a lebegőpontos utasításkészlet a különféle gépek között nem konzisztens. Például, míg a Cray-2-nek nincs lebegőpontos osztása, addig a Motorola 68882 rendelkezik osztással, négyzetgyökvonással, sinus és cosinus művelettel.
- A MFLOPS sebesség nem csupán az integer és lebegőpontos műveletek keverési arányával változik, hanem a gyors és lassú lebegőpontos műveletek arányával is. Például, egy 100%-ig lebegőpontos összeadást tartalmazó program gyorsabb lesz, mint egy 100%-ig csak lebegőpontos osztást tartalmazó program.

Mindkét probléma megoldása, hogy a forrásprogram tartalmazzon egy stabil számú lebegőpontos műveletet és azt osszuk el a végrehajtási idővel.

Értékelési módszerek



Paraméterlista

Egy általános és széles körben elfogadott módszer a hardver-teljesítmények összehasonlítására a releváns paramétereik összevetésével.

A paramétereket egy megfelelő táblázat formájában összegzik, és a számítógépeket olyan érték szerint rendezik, amely a specifikus feladatuk számára fontos. Ez a módszer kevésbé átlátható eredményt ad, ha sok paramétert veszünk figyelembe, viszont fontos különbségek mellett elmehetünk, ha túl kevés paramétert veszünk figyelembe.

A paraméter-lista bővíthető az egyes igényeknek megfelelően. A perifériák is paraméterezhetőek hasonló módon. Gyakori, hogy az egyedi adatokat aránypárba állítják, például, az egység-árral. Ez például lehetővé teszi a tároló bitenkénti árát (bit/USD), vagy a gép átlagos teljesítményét.

Ez az értékelési módszer nem veszi figyelembe elsődlegesen a hardver működési sebességét, hanem más mutatókat, mint a memória kapacitás, az adattípusok száma, az utasítás-típusok, a szóhossz, stb. alkalmaz.

Jellemző	A	B	C
CPU szóhossza [bit]	16	32	32
Órajel frekvencia [MHz]	40	40	60
Gépi utasítások száma	193	250	430
Memóriaméret [MB]	16	16	32
Meddig bővíthető	-	128	128
...			
Ár			
Gyártó jó híre			

Utasítás mix

A proceszor teljesítménye alapján kiválasztanak néhány jellemzõ utasítást.

Az utasítás-mix módszer teljesítmény-számításon alapul, melynek bázisa a központi egység átlagos utasítás-végrehajtási ideje. Az I/O mûveleteket nem számítják be.

$$t = \sum_{i=1}^{n} k_i t_i$$

t - az utasítások végrehajtási ideje
n - a figyelembe vett utasítások száma
k_i - az i-edik utasítás súlyfaktora
t_i - az i-edik figyelembe vett utasítás végrehajtási ideje [s]

Az alkalmazott súlyfaktort

- vagy tapasztalat
- vagy pedig a tipikus feladatra vonatkozó statisztikák

alapján határozzák meg. Különféle mixeket lehet definiálni a gazdasági és az üzleti, valamint a műszaki-tudományos feladatokhoz. A mixek előnye, hogy viszonylag egyszerűen számíthatók és összehasonlítható számértékeket adnak. Ezen kívül olyan utasítás-csoportot választhatunk és olyan súlyokat rendelhetünk ezekhez, melyek legjobban megfelelnek annak a feladatnak, melynek számára a gépet teszteljük.

GAMM-mix

$$t = \frac{1}{300}\left(t_1 + 2t_2 + 3t_3 + 4t_4 + \frac{t_5}{5}\right)$$

t - a GAMM mix értéke

t_1 -két harminc komponensű vektor skaláris szorzata számításához szükséges idő

t_2- két harminc komponensű vektor összegének számításához szükséges idő

t_3 -a Horner módszerével számított tízedik hatvány

t_4 -a Newton módszerével történő négyzetgyök meghatározásához szükséges öt iteráció ideje

t_5 -100 elemű vektor maximális elemének megtalálása

Ez a módszer tisztán a CPU teljesítményét értékeli, és figyelmen kívül hagyja a címzési módokat, valamint az utasítás típusokat.

Valós feladat

Egy teljesítmény-teszt (benchmark) végrehajtása a felhasználó számára konkrétabb eredményt ad, mint az utasítás mixek. Ez a módszer **tipikus feladatok** definiálását jelenti, amely tartalmazza a szükséges I/O műveleteket, és ezen feladatok végrehajtási idejét **méri**. Ezzel a módszerrel a különböző típusú és gyártójú gépeket sorrendbe állíthatjuk.

A tipikus feladat meghatározása rendkívül problematikus lehet. A túlságosan komplex feladat nem gazdaságos, mivel hosszú fejlesztési és futtatási fázist eredményez. A műszaki és tudományos tesztek egyszerű matematikai használó választhat a program futása közben.

A műszaki és tudományos tesztek egyszerű matematikai problémák

- polinom számítás
- mátrix-számítás
- interpoláció
- képletszámítás
- más területekre elégséges az I/O és matematikai orientáltságú műveletek kombinációja.

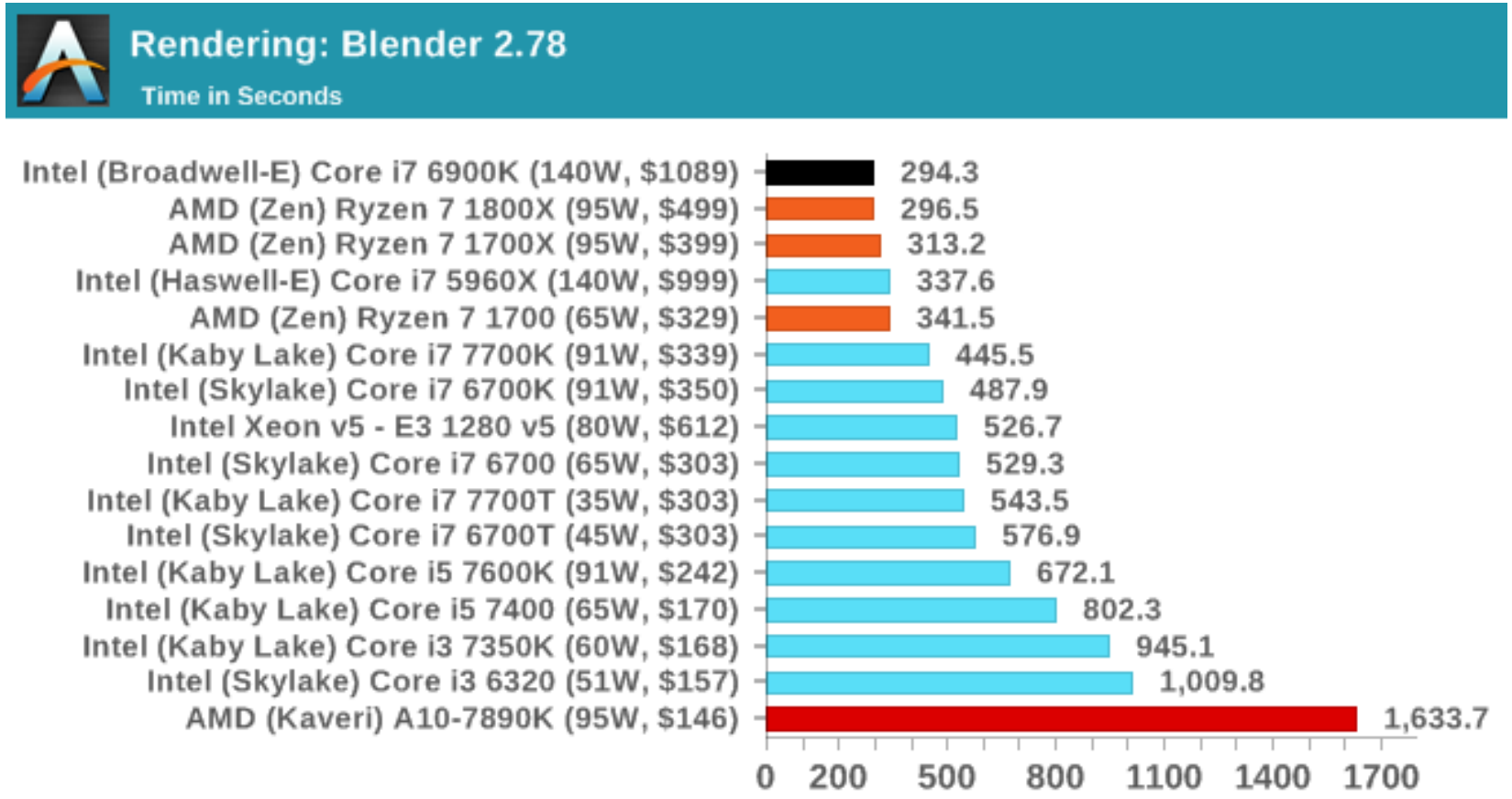
Valós feladatra példaként szolgálhatnak

- a fordítóprogramok
- a szövegszerkesztők, vagy
- a CAD eszközök.

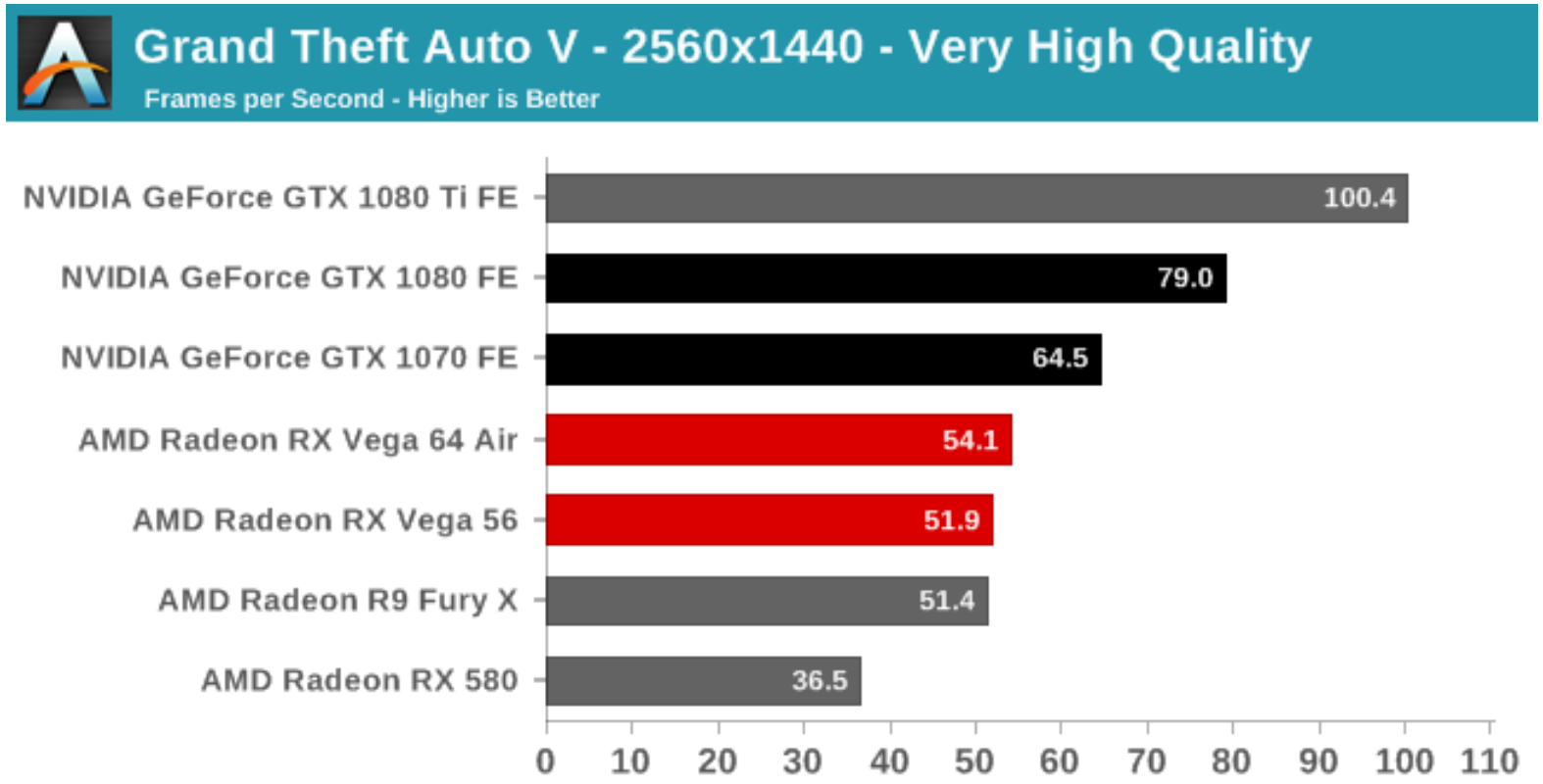
A valós programoknak van

- input-juk
- output-juk
- egy opciójuk, amit a felhasználó választhat a program futása közben.

Intel és AMD CPU-k összehasonlítása



AMD és NVIDIA CPU-k összehasonlítása



Kernel

Kiemelik a valós programok **kulcs elemeit**, és ezeket használják teljesítmény értékelésre.

A valós programokkal ellentétben egyetlen felhasználó sem fogja futtatni a kernel programokat, azok kizárólag csak a teljesítmény értékelésére szolgálnak. A kernelek leginkább arra használatosak, hogy a gép egyes sajátossága teljesítményét elhatároljuk, izoláljuk, és így magyarázzuk meg a valós programok teljesítmény-különbségének az okait, azaz kísérletet tesznek arra, hogy egy széles programkör vonatkozásában megfeleljenek a műveletek és operandusok gyakoriságának.

LinPack kernel

1976-ban publikálták egy, a Fortranban gyakran használt lineáris algebra szubrutin gyűteményként (innen ered a neve is). A program nagy mátrix-szal végez műveletet. A belsõ szubrutinok a mátrixot egydimenziós tömbként kezelik. A szabványos változatban a mátrix mérete 100x100 (egy kétszáz eleműnek deklarált kétdimenziós tömbön belül), de nagyobb változatai is léteznek.

Az eredményt általában MFLOPS-ban közlik. A program által végrehajtott lebegõpontos mûveletek számát a tömb méretébõl határozhatjuk meg. Ez a terminológia azt jelenti, hogy a nem lebegõpontos mûveleteket vagy elhanyagoljuk, vagy más módon határozzuk meg, de a végrehajtási idejüket a lebegõpontos műveletek idejébe beszámítják.

Livermore FORTRAN Kernel vagy Livermore Loops

Ez a módszer vezette be a natív MFLOPS mellett a normalizált MFLOPS-ot: az összeadás-kivonást egyetlen egységnek tekintve, a szorzás-osztás például négy, az exp, sin pedig nyolc egység. Például, egy olyan utasításssorozat, amely tartalmaz egy ADD, egy DIVIDE és egy SIN műveletet, az 13 normalizált lebegőpontos műveletként jön számításba. Így a natív MFLOPS nem egyezik meg a normalizált MFLOPS értékkel. Numerikus jellegű feldolgozások értékelésére szolgál.

Benchmark

A benchmarkokat **mesterségesen** hozták létre annak érdekében, hogy megfeleljenek egy átlagos végrehajtási profilnak. A szintetikus benchmarkokban egyetlen programrészlet sincsen valós programokból, míg az összes többi benchmarkban lehet valós programrészlet. A benchmark programokra jellemző, hogy **nem rendelkeznek input** adatokkal és igen kevés output adatot állítanak elő, azaz mindig ugyanazt az eredményt szolgáltatják.

Játék benchmark

A játék benchmarkok tipikusan 10 és 100 forráskód-sor közötti terjedelmű programok. Ezek olyan eredményeket szolgáltatnak, melyek már a játék benchmark futása előtt ismertek. Ilyenek,, a Sieve of Erasthotenes, a Puzzle, a Quicksort. Ezek létező programok ugyan, de nem valósak, mert csak a teljesítmény mérésére használják őket.

Benchmark sorozat

Benchmark sorozatok általában egy-egy utasításban térnek el egymástól, így lehetőséget nyújtanak egy-egy gép gyorsabb-lassabb utasításainak kiemelésére. Itt a súlyozást a felhasználó kívánság szerint elvégezheti a számára tipikus feladat súlyával, de nem létezik feladat-kategóriánként elismert súlyozású eredmény. Nem tesztel lebegőpontos műveletet, ezért tudományos-műszaki feldolgozáshoz ez a teszt nem alkalmazható.

Szintetikus benchmark (értékelő feladat)

Hasonló a filozófiája a kernelhez, és pont ebben különbözik a benchmark sorozattól, hogy a szintetikus benchmarkok kísérletet tesznek arra, hogy egy széles programkör vonatkozásában megfeleljenek a műveletek és operandusok gyakoriságának. A különbség az, hogy míg a kernel valós programok részleteit tartalmazza, a szintetikus benchmark egyetlen valós programsort sem tartalmaz.

Index

Napjainkban megfigyelhető tendencia, hogy nem terhelik a felhasználót olyan fogalmakkal, mint a fixpontos benchmark, lebegőpontos benchmark..

Annak érdekében, hogy a számítástechnikában járatlan mai tipikus felhasználó gépválasztását megkönnyítsék, egy összetett mutatót, úgynevezett indexet használnak. Ilyen például az Intel által bevezetett iCOMP index.

Maga az index is fejlődik. A jelenlegi 2.0-ás változat tehát nem egy benchmark, hanem olyan **benchmarkok gyűjteménye**, amelyeket a relatív processzor teljesítmény-számításához használnak abból a célból, hogy segítsék a végfelhasználókat azon határozatuk meghozatalában, mely processzor alkalmas leginkább a számítási igényeik kielégítésére. Minél magasabb az iCOMP érték, annál magasabb a processzor relatív teljesítménye.

A 32-bites CPU teljesítmény három különböző aspektusát tartalmazza:

- integer;
- lebegőpontos és
- multimédia.

Az utóbbi négy alkomponensre van bontva:

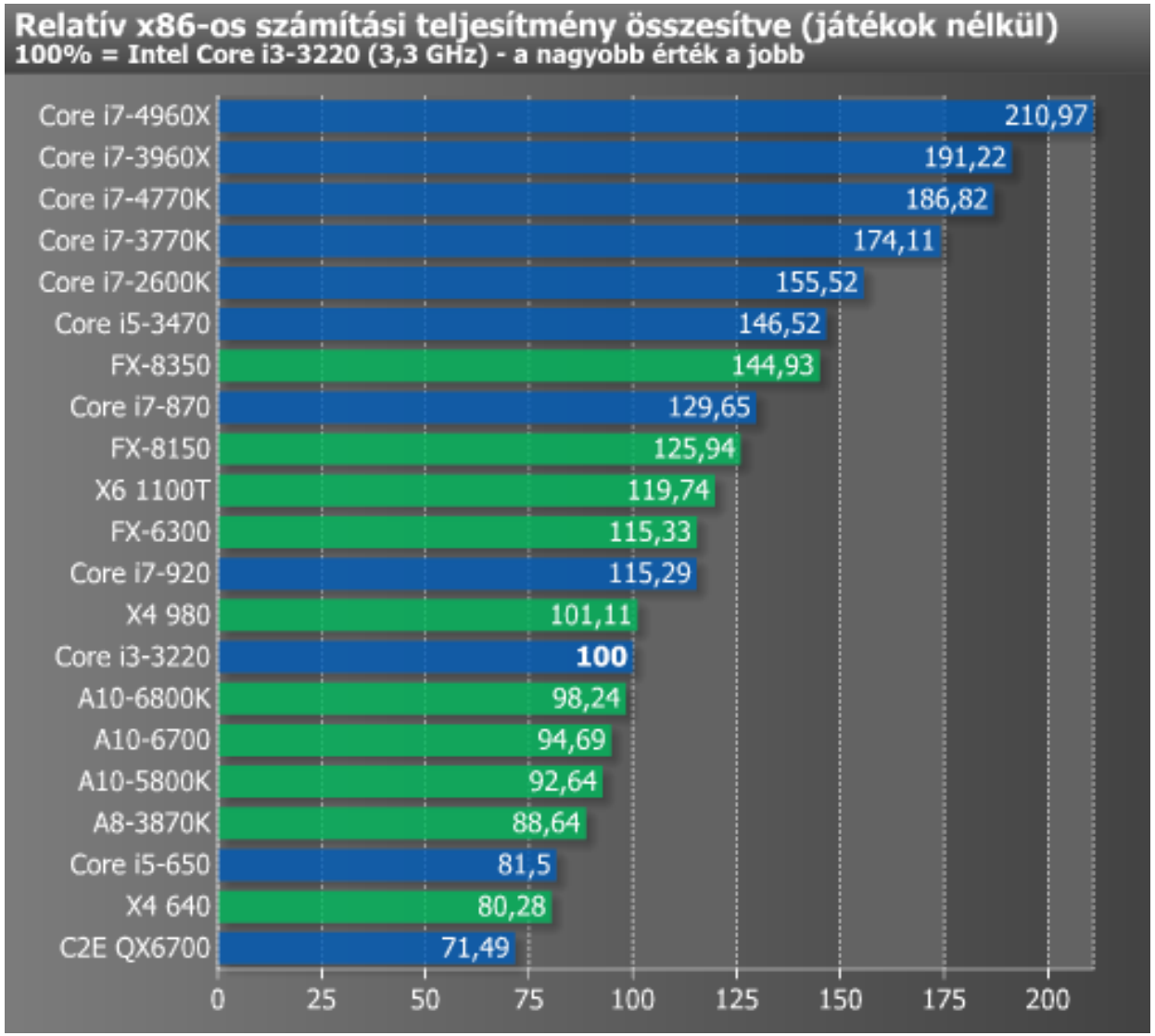
- audio;
- imaging;
- video
- 3D.

Mindezt öt benchmark kombinálásával érik el:

- CPUmark32: 32-bites CPU/memória szintetikus benchmark,
- NortonSI-32: 32-bites CPU/memória szintetikus benchmark;
- SPECint95;
- SPECfp95;
- Intel Media Benchmark.

Valamennyi processzort csak a bevezetésekor értékelik egy meghatározott, jól konfigurált, kereskedelemben elérhetõ rendszer felhasználásával.

Relatív teljesítmény X86 CPU-k



Teljesítmény/ár viszony

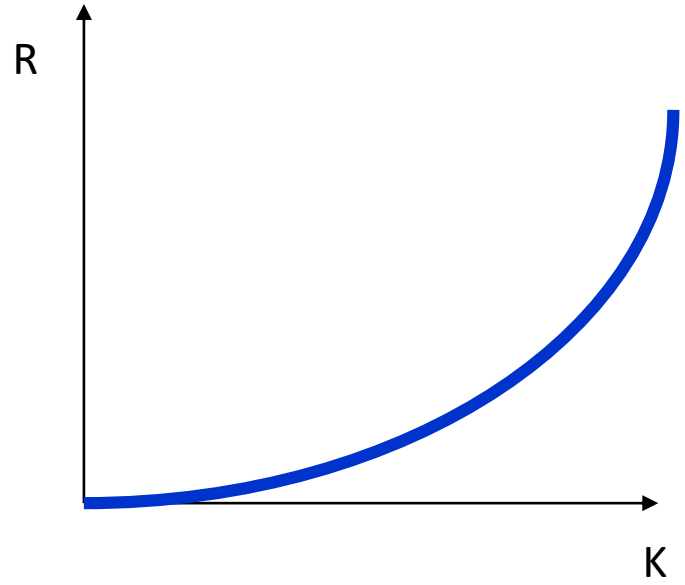
Grosch törvénye

Az értékelési módszerek fejlődése során a 70-es évek elején jelent meg a Grosch törvénye. Eszerint a számítógép ára és a teljesítménye között a **exponenciális** összefüggés áll fenn:

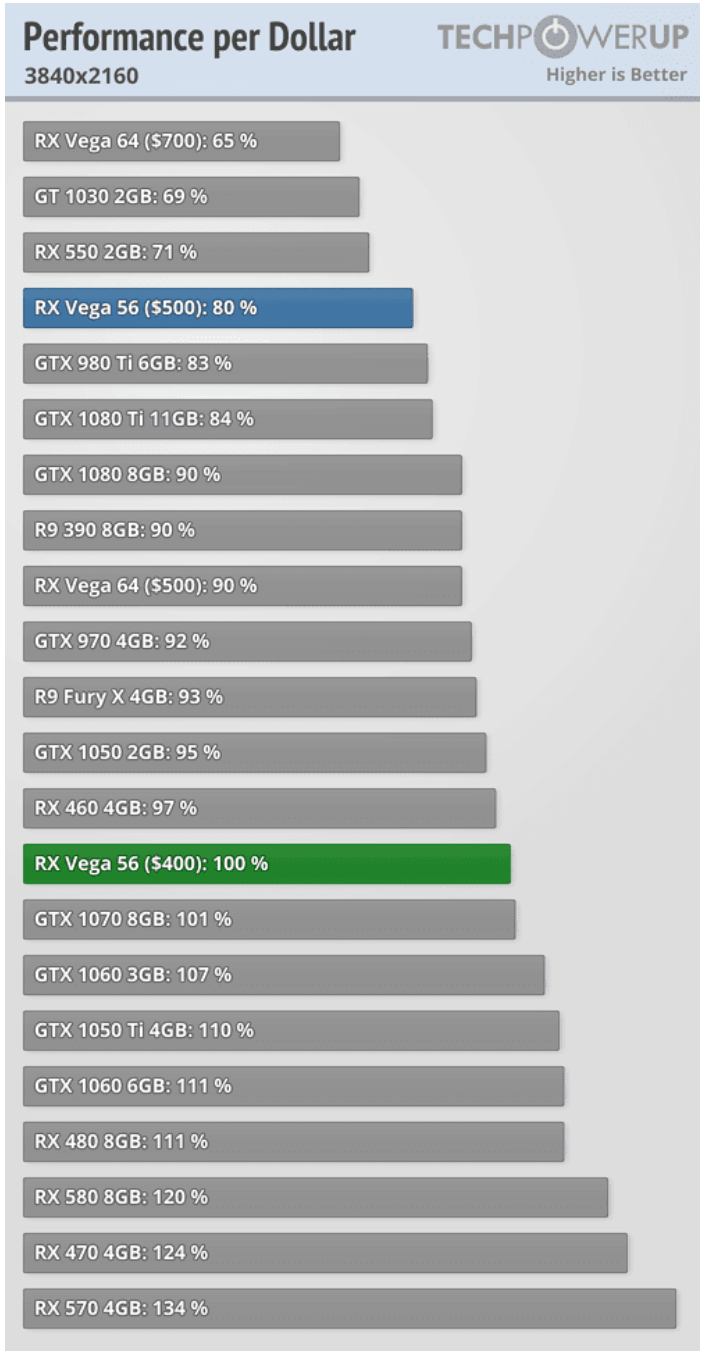
$$R=C\,K^n$$

- R - a gép teljesítménye
- C - konstans;
- K - a gép ára
- n - egy konstans, 1÷2 kis és középgépeknél, 1÷ 1,5 nagygépeknél.

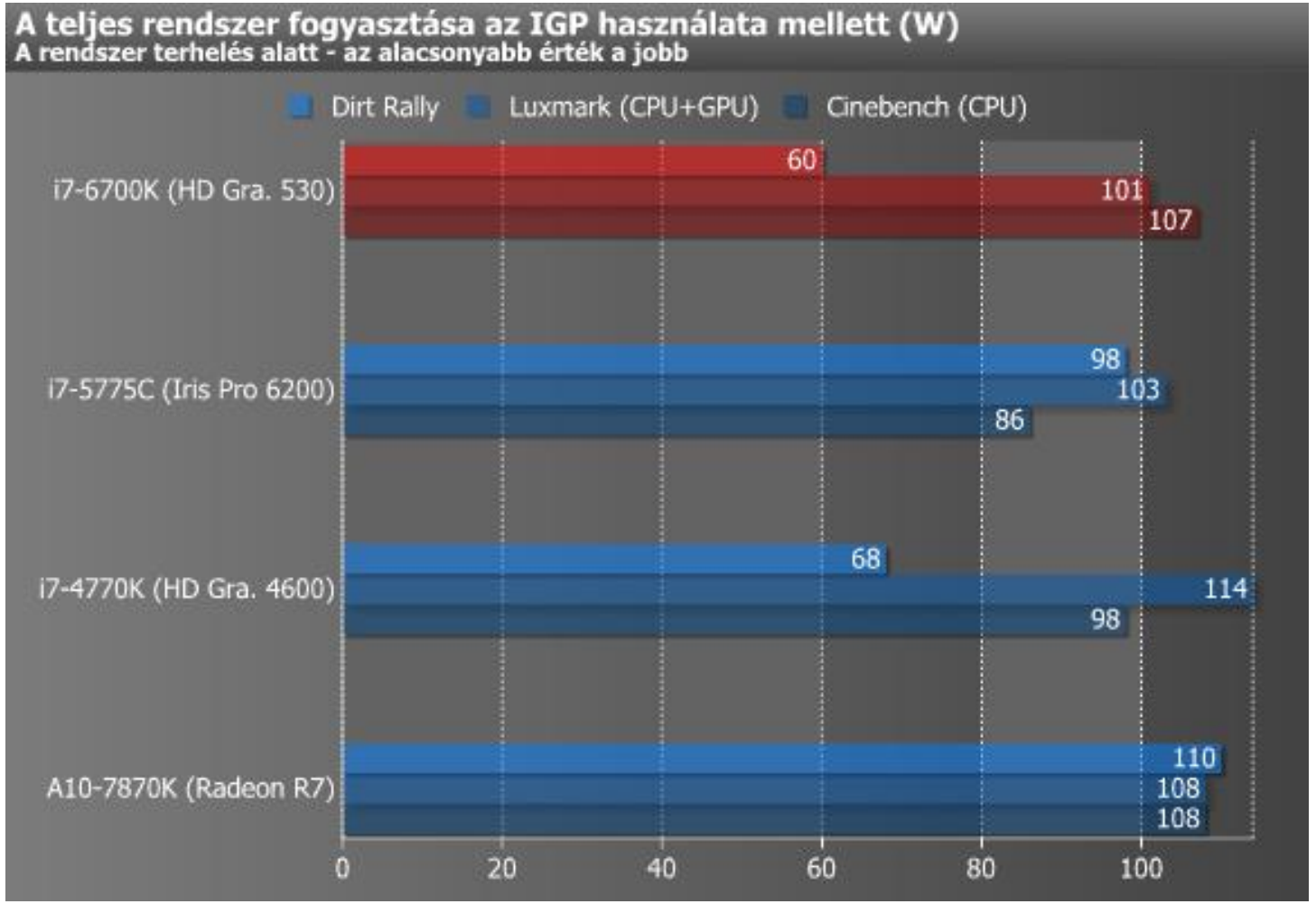
Grosch törvénye természeténél fogva empirikus és nem hordozza a rendszer ár/teljesítmény arányának pontos mérését.



Performance per Price



Disszipáció különböző számítási terhelés alatt



ÖE 15 145 630

Green Computing – Performance per Watt