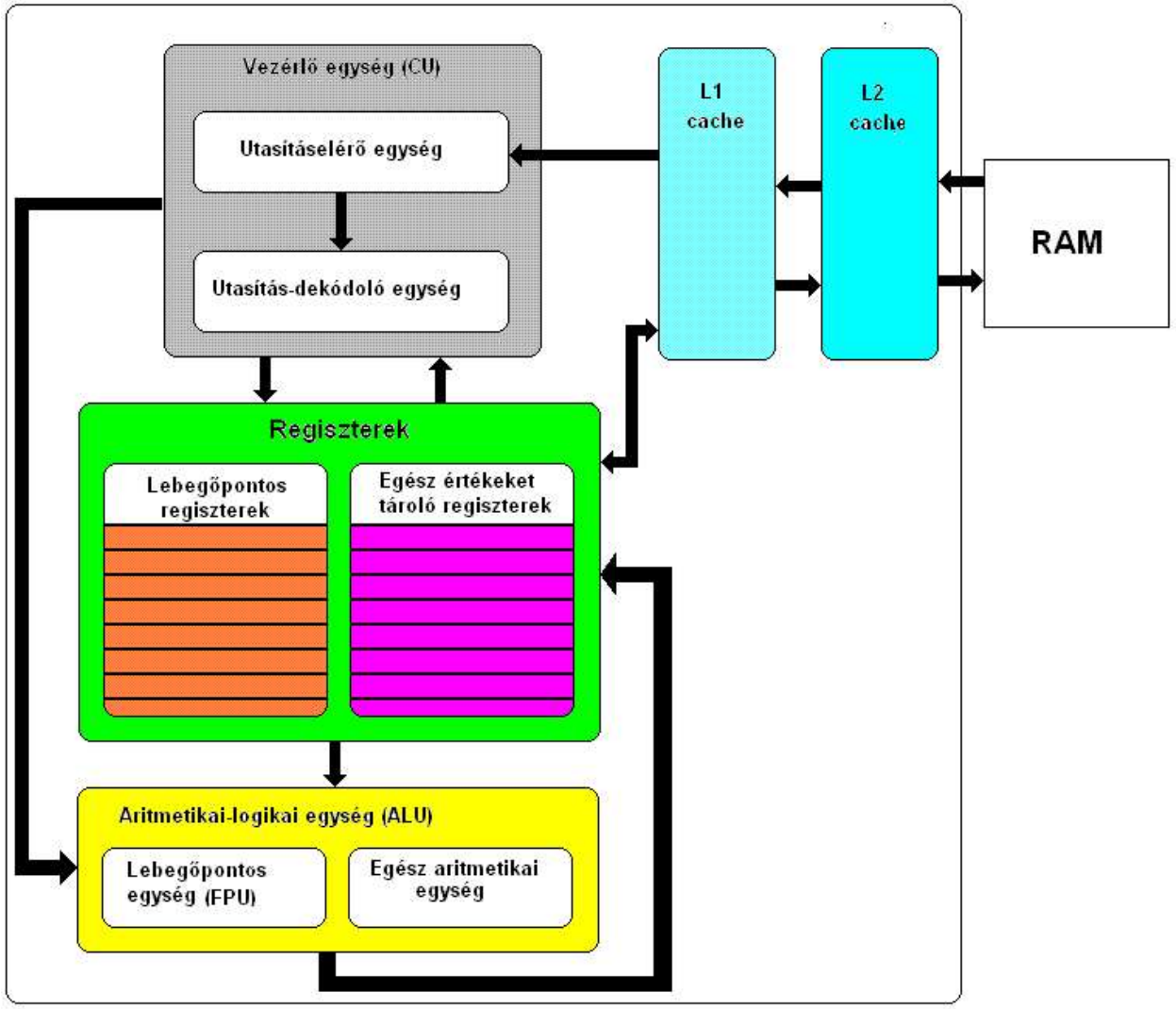
A **CPU (=Central Processing Unit – Központi Vezérlő Egység),**

más néven **processzor**, a számítógép „agya”, azon egysége, mely az utasítások értelmezését és végrehajtását végzi, félvezetős kivitelezésű, összetett elektronikus áramkör. A PC-be helyezett processzort az **Intel** fejlesztette ki.

A processzor alatt általában **mikroprocesszort** értünk, régebben a processzor sok különálló áramkör volt, ám mikroprocesszorral sikerült a legfontosabb dolgokat egyetlen szilíciumlapkára integrálni

Egy mai processzor felépítése



Az ábrán a nyilak a lehetséges adatáramlás irányát jelölik.

A processzor részei:

- **A vezérlő egység (CU=Control Unit)**
- **Az Aritmetikai-logikai egység (ALU=Arithmetical-Logical Unit)**
- **Regiszterek**
- **L1 (=Level 1) cache memória, más néven elsőszintű gyorsítótár**
- **L2 (=Level 2) cache memória, más néven másodszintű gyorsítótár**

A CU

Ez az egység vezérli a processzor működését. Néhány processzorban az utasítás-elérő, dekódoló egység a CU-n kívüli, külön áramköregységet alkotnak. A CU egy órajel-generátor segítségével vezérli a többi áramkör működését. Az órajel-generátor négyszögjeleket ad, a CU-nak. Minden elemi művelet a processzorban legalább egy órajelnyi időt vesz igénybe. Ezért a régebbi processzortípusoknál a processzor sebessége egyedül az órajel-frekvenciától függött.

Az órajel-frekvencia= az órajel-generátor által másodpercenként leadott négyszögjelek száma. Mértékegysége a Hertz (Hz), illetve a kHz (kiloHertz), MHz (MegaHertz), GHz (GigaHertz).

- 1 kHz=1000 Hz
- 1 MHz=1 000 000 Hz
- 1GHz=1 000 000 000 Hz

Egy mai processzor órajele 3 - 4 GHz körüli.

Az ALU

Az Aritmetikai-logikai egység végzi el a CU által megrendelt aritmetikai műveleteket (összeadás, kivonás, szorzás, osztás) illetve logikai műveleteket (bitenkénti ÉS, VAGY, stb.)

Két, jól elkülöníthető részből áll:

Az első a fixpontos egység, amely csak egész számokkal végez műveleteket. (+, -, *, DIV, MOD, SHL, SHR, RoR, RoL, bitenkénti logikai műveletek)

A második a lebegőpontos egység. Feladata a lebegőpontos valós számokkal végzendő műveletek végrehajtása (+, -, *, /,gyökvonás, hatványozás, logaritmus, stb.)

A regiszterek

A regiszterek olyan gyorstárolók, amelyekben a processzor az éppen feldolgozás alatt lévő adatokat, részeredményeket, utasításkódokat vagy memóriacímeket tárol.

A regiszter legfontosabb jellemzője a mérete, amely megadja, hogy hány bites számot tud tárolni. Ez az érték rendszerint kettő valamely hatványa. A legelső mikroprocesszorban (Intel 4004)a regiszterek 4 bitesek voltak. A következő generáció processzoraiban már 8 bites regisztereket találunk.

A mai processzorokban kétféle regisztertípus van, éspedig:

1. a fixpontos regiszterek, amelyekben egész értékeket tárolunk. Ezek 64 bitesek.
2. A lebegőpontos regiszterek, ezekben normál alakú valós értékeket tárolunk. Ezek általában 128 bitesek.

Hány regiszter van egy mikroprocesszorban?

Ez processzortípustól függ.

Pl az Intel Pentium 4-esben 8 db 64 bites fixpontos, és 8 db 128 bites lebegőpontos regiszter van.

Az Itanium 2 szerverprocesszorban 128 db fixpontos, 128 db lebegőpontos regiszter van

A többi processzorban a regiszterszám e 2 érték között van.

A regiszterek egy része általános célú, azaz bármilyen típusú érték tárolható benne.

Vannak azonban speciális célú regiszterek, amelyekben csak bizonyos típusú adatok tárolhatók.

Ilyen a PC regiszter vagy programszámláló regiszter (PC= Program Counter) amelyben a következő utasítás memóriabeli címét tároljuk.

Az állapotregiszter vagy státuszregiszter a műveletek során bekövetkező állapotváltozásokat tárolja, jelzi. Ennek a regiszternek a bitjeit jelzőbiteknek, vagy flageknek nevezzük. Az állapotregisztert ezért a szaknyelv FLAGS regiszternek is nevezi. Rendeltetése például a különböző hibák jelzése is, (pl túlcsordulás, ha az eredmény nem fér bele a regiszterbe, vagy ha nem létező címről akarunk olvasni/odaírni, stb.)

Intel x86 FLAGS Register			
Bit #	**Jelölés**	**Leírás**	**Kategória**
		FLAGS	
0	**CF**	**Carry flag=1, ha összeadáskor van átvitel**	**S**
1	**1**	**Reserved**	
2	**PF**	**Paritásbit, az eredmény páros vagy nem**	**S**
3	**0**	**Reserved**	
4	**AF**	**Adjust flag**	**S**
5	**0**	**Reserved**	
6	**ZF**	**Zero flag, értéke azt jelzi, hogy az eredmény 0 vagy sem**	**S**
7	**SF**	**Sign flag=az eredmény előjelét jelzi (+ vagy -)**	**S**
8	**TP**	**Trap flag (single step)**	**X**
9	**IF**	**Interrupt enable flag (megszakítás engedélyezése)**	**X**
10	**DF**	**Direction flag**	**C**

11	OF	Overflow flag Ez jelzi, hogy az eredmény belefér-e a regiszterbe vagy sem	S
12, 13	IOPL	I/O privilege level (286+ only)	X
14	NT	Nested task flag (286+ only)	X
15	0	Reserved	
		EFLAGS	
16	RF	Resume flag (386+ only)	X
17	VM	Virtual-8086 mode flag (386+ only)	X
18	AC	Alignment check (486SX+ only)	X
19	VIF	Virtual interrupt flag (Pentium+)	X
20	VIP	Virtual interrupt pending (Pentium+)	X
21	ID	Identification (Pentium+)	X
22	0	Reserved	
23	0	Reserved	
24	0	Reserved	
25	0	Reserved	
26	0	Reserved	
27	0	Reserved	
28	0	Reserved	
29	0	Reserved	
30	0	Reserved	
31	0	Reserved	
		RFLAGS	
32-63	0	Reserved	

S: Status flag= állapot flag
C: Control flag= vezérlő flag
X: System flag= rendszer flag

A "Reserved" bitek olyan bitek, amelyek rendeltetését a processzor belső műveletek végzésénél használja.

Az L1 cache, vagyis az elsőszintű gyorsítótár

A működéshez szükséges adatokat, utasításokat a processzornak a RAM memóriából kell kiolvasni, ami igen lassú művelet, és a processzor ezáltal várakozásra kényszerülne. Ezért egy nagyobb adatblokkot a processzor egy menetben az L1 cache memóriába bemásol, és ezt használja. Az L1 cache memória sokkal gyorsabb, mint a RAM. Az elsőszintű gyorsítótár jellemző mérete 16-32 kB.

Az L2 cache, vagyis a másodszintű gyorsítótár

A RAM-ból való kiolvasás/ kiírás gyakoriságának további csökkentésére vezették be a másodszintű gyorsítótárat. Ez jóval nagyobb méretű, a mai processzorokban az L2 cache memória mérete 1-2 MB. Egy speciális algoritmus előrejelzi, mely memóriaszeletekre lesz a processzornak szüksége a közeljövőben, majd ennek ismeretében a cache memóriába betöltődik az előrejelzett tartomány. Így a processzornak az esetek többségében a cache-ből kell olvasnia, nem a RAM-ból. Ezáltal a processzor működése jelentősen felgyorsul.

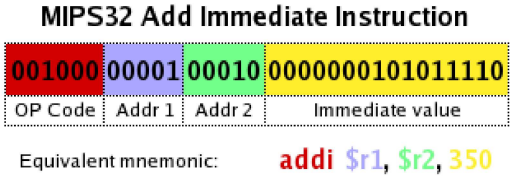
A processzor működése

A processzor egy utasítást több elemi lépésben hajt végre. Egy hagyományos, Neumann-elvű (RISC) processzornál ezek az elemi lépések a következők:

1. Utasítás elérése (fetch)
2. Utasítás dekódolása (decode)
3. Operandusok beolvasása (Mem)
4. Utasítás végrehajtása (execute)
5. Eredmény visszaírása (writeback)

Az utasítás elérés: az a művelet, melynek során a következő utasítás memóriabeli címét meghatározzuk, és az utasítást beolvassuk a processzor egyik regiszterébe

Az utasítás dekódolása: Az utasítás egy vagy több 32 bites (vagy 64 bites) szám, amelyet értelmezni kell, vagyis meg kell határozni a jelentését.



Az első 6 bit adja a műveleti kódot, vagyis az opkódot. Jelen esetben ez 8, és a 8-as ebben a gépi kódban az úgynevezett közvetlen összeadást jelenti (addi=add immediate). A második 5 bit az egyik regiszter számát, azonosítóját jelenti, a harmadik szintén. Az utolsó 16 bit jelöli azt a számot, (350) amelyet az $r2 regiszterbeli értékhez hozzá kell adni, és az eredményt az $r1 regiszterbe kell beírni. Tehát a teendő:

$r1:=($r2 értéke)+350

A harmadik lépés az **operandusok beolvasása.** Az operandusok azok az adatok, amelyekkel a műveletet végezzük. Pl a 6+5 utasításnál a **6** és az **5** az operandusok, a **+** az operátor. Az operandusok vagy az egyik regiszterben, vagy valamelyik cache memóriában, ritkábban a RAM-ban találhatók. Az utóbbi két esetben van szükség az operandusok beolvasására. A cache-beli operandusok a processzor valamely regiszterébe kerülnek.

A negyedik lépés a **végrehajtás,** amelynek során a CU a végrehajtandó műveletet elrendeli, az ALU pedig végrehajtja.

Az ötödik lépés a **visszaírás**, Az eredményeket visszaírjuk valamely regiszterbe vagy a memóriába.

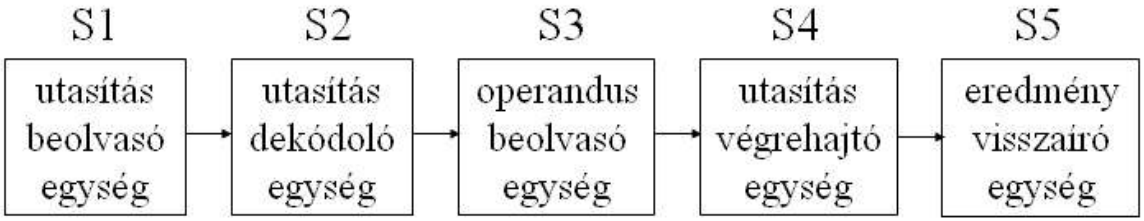
Párhuzamos feldolgozás

Alapvetően két stratégiát alkalmazunk, amellyel a párhuzamos feldolgozást megvalósítjuk:

- Az utasítás szintű párhuzamosság, **ILP** (ILP= **I**nstruction **L**evel **P**arallelism). Ezt valósítja meg a pipelining módszer.
- A programszál szintű párhuzamosság **TLP** (TLP= **T**hread **L**evel **P**arallelism).

A pipelining

A fentebb felvázolt processzor a legegyszerűbb processzor, ez az öt elemi lépést egy-egy órajel alatt hajtja végre.



Így az utasítás elérése egy ütem, dekódolás a második ütem, az operandusbeolvasás a harmadik,a végrehajtás a negyedik, visszaírás az ötödik ütem alatt történik. Így egy utasítás 5 "ütemet", időegységet venne igénybe.

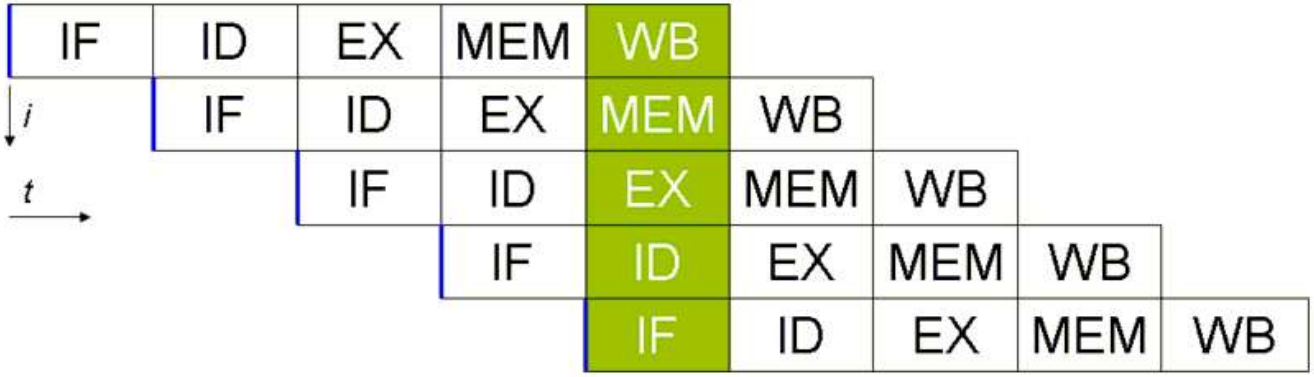
Az olyan processzort, amelynél egy elemi utasítás egynél több lépést igényel, **szubskaláris** processzornak nevezzük.

A szubskaláris processzorok nem hatékonyak: minden órajel alatt csak az egyik alegység dolgozik bennük, a többi várakozik.

Amíg az egyik egység (az utasítás-elérő egység dolgozik, a dekódoló, az ALU, stb tétlenül várakozik.

Erre a problémára találták ki a pipelining módszert: itt az utasítás végrehajtása egy szerelőszalaghoz hasonlóan történik: mialatt az egyik utasítás eredményét visszaírjuk, a másodikat éppen végrehajtja az ALU, a dekódoló már a harmadik utasítást értelmezi, az elérő pedig a negyedik utasítást tölti be.

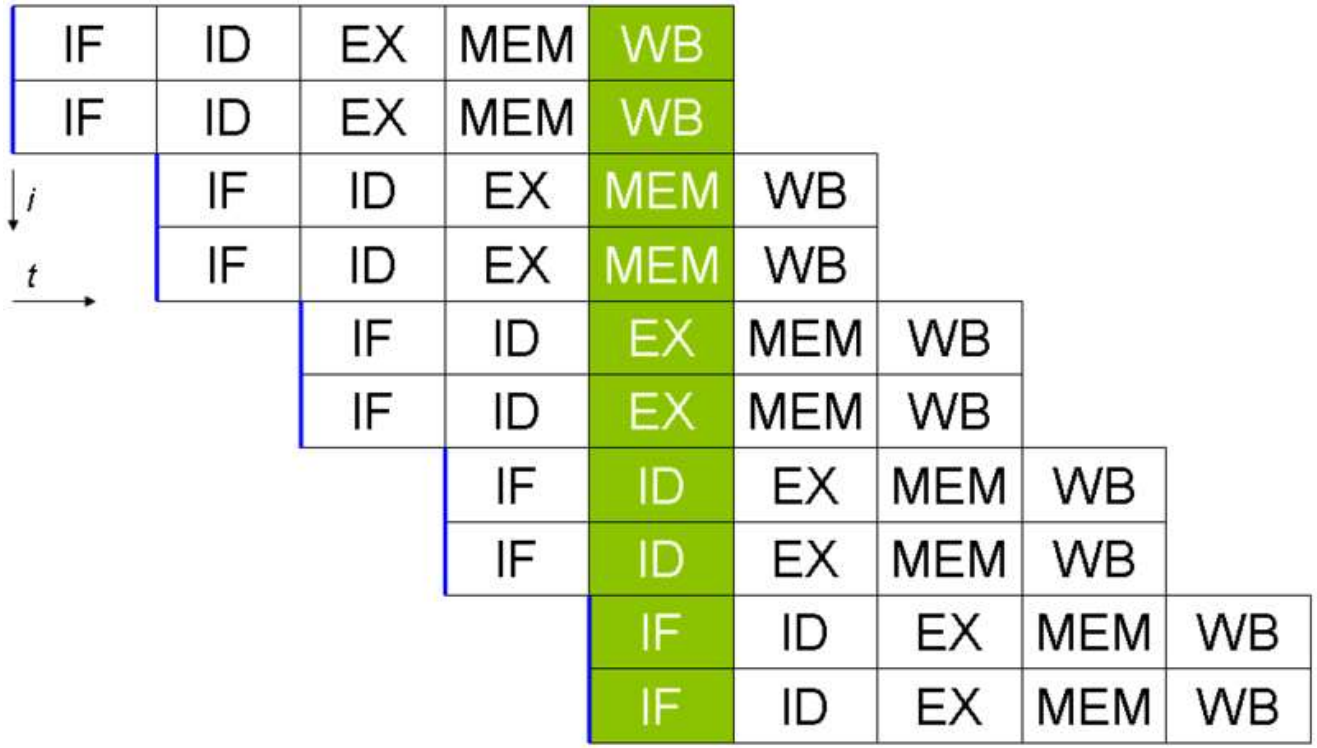
Ezzel a módszerrel egy utasítást lehet órajelenként elvégezni. Az órajelenként egy utasítást végző processzor a **skaláris** processzor, az ilyen feldolgozást **skaláris** feldolgozásnak nevezzük.



Az ilyen processzorok felépítését bonyolítja az a helyzet, amikor egy utasítás a közvetlenül előtte levő utasítás eredményétől függ. Ilyenkor a "szerelőszalag" elakad, ezt külön kezelni kell.

A további hatékonyság növelését olyan processzorok jelentik, amelyek már a **szuperskaláris** kategóriába tartoznak.

A **szuperskalár** processzorok egy órajel alatt több, mint egy utasítást képesek végrehajtani. Ezt úgy érik el, hogy egy processzor több "szerelőszalaggal", pipelinig vonallal rendelkezik, ami természetesen növeli a tranzisztorok számát. Egy speciális, "forgalomirányító" egység (dispatcher) szervezi a különböző szerelőszalagokra az utasításokat aszerint, hogy melyik szerelőszalag elérhető az adott pillanatban.



A mai processzorok mindegyike szuperskalár szervezésű.

A TLP

A programszál: olyan, egyazon programbeli részfeladat, amely más részfeladattal egy időben fut, vagy futhat.

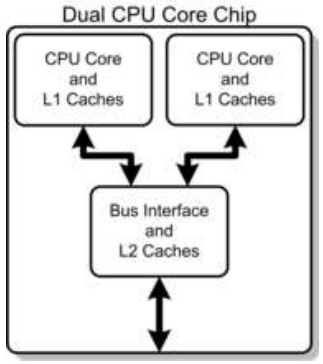
Két megoldás létezik itt is:

- A HTT, vagyis Hyper Threading
- A többmagos processzor

HTT technológia egyfajta félmegoldást jelent. (HTT=**H**yper **T**hreading **T**echnology) Itt bizonyos egységeket két példányban találunk, de a kettőzött egységek ugyanazon az ALU-n osztoznak. Az Intel P4 processzoroknál ezt a módszert alkalmazzák.

Ahhoz, hogy a párhuzamosságot ki tudjuk használni, operációs rendszerszintű és alkalmazói szoftverszintű támogatásra van szükségünk. Ha tehát nincs megfelelő operációs rendszerünk, illetve szoftverünk, amely ezt a lehetőséget ki tudja használni, akkor nem tapasztalhatunk teljesítménynövekedést.
A HTT-vel elérhető teljesítménynövekedés 15-30% közötti.

A többmagos processzor



A személyi számítógépekben használt processzorok akkor képesek ténylegesen párhuzamos programszál-feldolgozásra, ha több processzormagot alkalmazunk egy tokozáson belül. Ezt ma a kétmagos, illetve négymagos processzorok valósítják meg. Ezek akár 100%-os teljesítménynövekedést nyújthatnak. Ezek a processzorok egyazon L2-cache memórián osztoznak, minden egyébből 2, illetve 4 példány van. Nyilván a 4 magos processzorral a teljesítménynövekedés többszörös.

A vektorprocesszor

A vektorprocesszor az egy utasítás-több adat szemlélet alapján működik, ahol sok adaton kell egyszerre hasonló utasításokat végrehajtani. Ilyen alkalmazások:

- a mérnöki alkalmazások, ahol több alkatrészek mozgását kell ugyanazon mozgástörvények szerint nyomon követni.
- a kriptográfia, azaz a rejtjelezés, rejtjel-fejtés tudománya.
- meteorológia, időjárás előrejelzés,
- video- audio alkalmazások, pl egy filmtrükk renderelése, ahol több képkockát kell ugyanolyan algoritmus szerint megjeleníteni.

A vektorprocesszorok úgynevezett vektorregisztereket tartalmaznak. Ezek a hagyományos regiszterekkel szemben egyszerre több értéket is tárolni tudnak. A regisztereket az aritmetikai egységgel gyors szállítószalag kapcsolja össze. Így akár egyszerre több számmal is tudunk műveleteket végezni.

A multiprocesszorok

A multiprocesszorok