A konferencia terem foglaltsága miatt a november 11-i előadást november 12-én 10h-tól tudom megtartani a konferencia teremben.

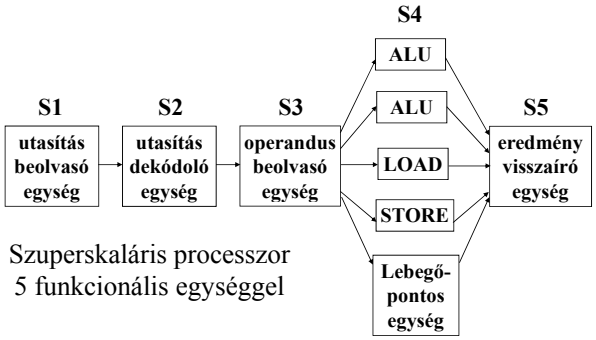


**Elágazás jövendölés**

- Figyeljük, hogy az utolsó *k* feltételes elágazást végre kellett-e hajtani. Ez egy *k* bites számot eredményez, ezt az elágazási előzmények blokkos regiszterében tároljuk. Ha a *k* bites szám megegyezik a táblázat valamely bejegyzésének a kulcsával (találat), akkor az ott talált jövendölést használja.



**Szuperskaláris architektúrák (2. 6. ábra)**




Szuperskaláris processzor 5 funkcionális egységgel



Szuperskaláris architektúra esetén a dekódoló egység az utasításokat mikroutasításokra darabolhatja. Legegyszerűbb, ha a mikroutasítások végrehajtási sorrendje megegyezik a betöltés sorrendjével, de ez nem mindig optimális.

**Függőségek**

Ha egy utasítás írni/olvasni akar egy regisztert, akkor meg kell várja azon korábbi utasítások befejezését, amelyek ezt a regisztert írni/olvasni akarták!



**Függőségek**

Egy utasítás nem hajtható végre az alábbi esetekben:

- **RAW** (valódi) függőség (Read After Write):
  Onnan akarunk olvasni operandust, ahova még nem fejeződött be egy korábbi írás.
- **WAR** függőség (Write After Read):
  Olyan regiszterbe szeretnénk írni az eredményt, ahonnan még nem fejeződött be egy korábbi olvasás,
- **WAW** függőség (Write After Write):
  Olyan regiszterbe szeretnénk írni az eredményt, ahova még nem fejeződött be egy korábbi írás. Ne boruljon föl az írások sorrendje!



**Függőségek:** nem olvashatjuk, aminek az írása még nem fejeződött be (**RAW**), és nem írhatjuk felül, amit korábbi utasítás olvasni (**WAR**) vagy írni akar (**WAW**).

Annak nincs akadálya, hogy onnan olvassunk, ahonnan egy korábbi olvasás még nem fejeződött be, tehát az olvasás utáni olvasás nem okoz függőséget. Nincs **RAR** függőség.

**Függőségek nyilvántartása:** Regiszterenként egy-egy számláló, hogy hányszor használják a végrehajtás alatt lévő (mikro)utasítások a regisztert olvasásra illetve írásra.

Pl. Tegyük fel, hogy a dekódoló ciklusonként két utasítást tud kiosztani végrehajtásra (a valóságban 4-6 utasítást). Az n. ciklusban dekódolt utasítás végrehajtása legkorábban az (n+1). ciklusban kezdődhet, és az (n+2)-ben fejeződik be, de a szorzás csak az (n+3)-ban.

Az utasítások indítása és befejezése az eredeti sorrendben történjék!

**C**= ciklus, **K**=kiosztás, **B**=befejezés (**~4.43. ábra**).



| | | | | | Olvasott regiszterek | | | | | | | | Írt regiszterek | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| C | # | Dekódolt | K | B | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 |
| 1 | 1 | R3=R0*R1 | 1 | | 1 | 1 | | | | | | | | | | 1 | | | | |
| | 2 | R4=R0+R2 | 2 | | 2 | 1 | 1 | | | | | | | | | 1 | 1 | | | |
| 2 | 3 | R5=R0+R1 | 3 | | 3 | 2 | 1 | | | | | | | | | 1 | 1 | 1 | | |
| | 4 | R6=R1+R4 | - | | 3 | 2 | 1 | | RAW R4 miatt | | | | | | | 1 | 1 | 1 | | |
| 3 | | | | | 3 | 2 | 1 | | I2 csak I1 után fejeződhet be | | | | 1 | 1 | 1 | | | | | |
| 4 | | | | 1 | 2 | 1 | 1 | | | | | | | | | | 1 | 1 | | |
| | | | | 2 | 1 | 1 | | | Csak a ciklus végére történik | | | | | | | | 1 | | | |
| | | | | 3 | | | | | meg a visszaírás R4-be | | | | | | | | | | | |
| 5 | | | 4 | | | 1 | | | 1 | | | | | | | | | | 1 | |
| | 5 | R7=R1*R2 | 5 | | | 2 | 1 | | 1 | | | | | | | | | | 1 | 1 |
| 6 | 6 | R1=R0-R2 | - | | | 2 | 1 | | 1 | WAR R1 miatt | | | | | | | | | 1 | 1 |
| 7 | | | | 4 | | 1 | 1 | | | | | | | | | | | | | 1 |
| 8 | | | | 5 | | | | | | | | | | | | | | | | |
| 9 | | | 6 | | 1 | | 1 | | | | | | | 1 | 0 | | | | | |
| | 7 | R3=R3*R1 | - | | 1 | | 1 | | RAW R1 miatt | | | | | 1 | 0 | | | | | |
| 10 | | | | | 1 | | 1 | | | | | | | 1 | 0 | | | | | |
| 11 | | | | 6 | | | | | | | | | | | | | | | | |
| 12 | | | 7 | | | 1 | | 1 | | | | | | | | 1 | | | | |
| | 8 | R1=R4+R4 | - | | | 1 | | 1 | WAR R1 miatt | | | | | | | 1 | | | | |

*megjegyzés* *hiba*



| | | | | | Olvasott regiszterek | | | | | | | | Írt regiszterek | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| C | # | Dekódolt | K | B | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 |
| 9 | | | 6 | | 1 | | 1 | | | | | | | 1 | 0 | | | | | |
| | 7 | R3=R3*R1 | - | | 1 | | 1 | | RAW R1 miatt | | | | | 1 | 0 | | | | | |
| 10 | | | | | 1 | | 1 | | | | | | | 1 | 0 | | | | | |
| 11 | | | | 6 | | | | | | | | | | | | | | | | |
| 12 | | | 7 | | | 1 | | 1 | | | | | | | | 1 | | | | |
| | 8 | R1=R4+R4 | - | | | 1 | | 1 | WAR R1 miatt | | | | | | | 1 | | | | |
| 13 | | | | | | 1 | | 1 | | | | | | | | 1 | | | | |
| 14 | | | | | | 1 | | 1 | | | | | | | | 1 | | | | |
| 15 | | | | 7 | | | | | | | | | | | | | | | | |
| 16 | | | 8 | | | | | | 2 | | | | | 1 | | | | | | |
| 17 | | | | | | | | | 2 | | | | | 1 | | | | | | |
| 18 | | | | 8 | | | | | | | | | | | | | | | | |



Néhány gép bizonyos utasításokat átugorva függőben hagy, előbb későbbi utasításokat hajt végre, és később tér vissza a függőben hagyott utasítások végrehajtására (**~4.44. ábra**).



**Sorrendtől eltérő végrehajtás**
(kezdés és befejezés) esetén (**4.44. ábra**)

| | | | | | Olvasott regiszterek | | | | | | | | | | Írt regiszterek | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| C | # | Dekódolt | K | B | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | *1* | *2* | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | *1* | *2* |
| 1 | 1 | R3=R0*R1 | 1 | | 1 | 1 | | | | | | | | | | | | 1 | | | | | | |
| | 2 | R4=R0+R2 | 2 | | 2 | 1 | 1 | | | | | | | | | | | 1 | 1 | | | | | |
| 2 | 3 | R5=R0+R1 | 3 | | 3 | 2 | 1 | | | | | | | | | | | 1 | 1 | 1 | | | | |
| | 4 | R6=R1+R4 | - | | 3 | 2 | 1 | | 0 | RAW | | | | | | | | 1 | 1 | 1 | 0 | | | |
| 3 | 5 | R7=R1*R2 | 5 | | 3 | 3 | 2 | | 0 | *I5 megelőzi I4-et* | | | | | | | | 1 | 1 | 1 | 0 | 1 | | |
| | 6 | R1*(S1)*=R0-R2 | 6 | | 4 | 3 | 3 | | 0 | *WAR: R1 helyett S1* | | | | | | | | 1 | 1 | 1 | 0 | 1 | *1* | |
| | | | | 2 | 3 | 3 | 2 | | 0 | | | | | | | | | 1 | | 1 | 0 | 1 | *1* | |

**I4** nem indulhat **RAW** függőség (**R4**) **I2** miatt, de **adminisztrációt igényel**, hogy melyik regisztereket használja (függőséget okozhat az átugrott utasítás is!).

**I5** megelőzheti **I4** –et.

**I6** **R1=R0-R2** helyett ***S1*=R0-R2.** Az *S1* segéd regisztert használja **R1** helyett (**regiszter átnevezés**). Az eredményt később átmásolhatja **R1** -be, ha **R1** fölszabadult.



| | | | | | Olvasott regiszterek | | | | | | | | | | Írt regiszterek | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| C | # | Dekódolt | K | B | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | *1* | *2* | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | *1* | *2* |
| 1 | 1 | R3=R0*R1 | 1 | | 1 | 1 | | | | | | | | | | | | 1 | | | | | | |
| | 2 | R4=R0+R2 | 2 | | 2 | 1 | 1 | | | | | | | | | | | 1 | 1 | | | | | |
| 2 | 3 | R5=R0+R1 | 3 | | 3 | 2 | 1 | | | | | | | | | | | 1 | 1 | 1 | | | | |
| | 4 | R6=R1+R4 | - | | 3 | 2 | 1 | | 0 | RAW | | | | | | | | 1 | 1 | 1 | 0 | | | |
| 3 | 5 | R7=R1*R2 | 5 | | 3 | 3 | 2 | | 0 | *I5 megelőzi I4-et* | | | | | | | | 1 | 1 | 1 | 0 | 1 | | |
| | 6 | R1*(S1)*=R0-R2 | 6 | | 4 | 3 | 3 | | 0 | *WAR: R1 helyett S1* | | | | | | | | 1 | 1 | 1 | 0 | 1 | *1* | |
| | | | | 2 | 3 | 3 | 2 | | 0 | | | | | | | | | 1 | | 1 | 0 | 1 | *1* | |
| 4 | | | 4 | | 3 | 4 | 2 | | 1 | | | | | | | | | 1 | | 1 | 1 | 1 | *1* | |
| | 7 | R3=R3*R1*(S1)* | - | | 3 | 4 | 2 | 0 | 1 | | | | *0* | | | | | 1 | | 1 | 1 | 1 | *1* | |
| | 8 | R1*(S2)*=R4+R4 | 8 | | 3 | 4 | 2 | 0 | 3 | | | | *0* | | | | | 1 | | 1 | 1 | 1 | *1* | *1* |
| | | | | 1 | 2 | 3 | 2 | 0 | 3 | | | | *0* | | | | | 0 | | 1 | 1 | 1 | *1* | *1* |
| | | | | 3 | 1 | 2 | 2 | 0 | 3 | | | | *0* | | | | | 0 | | | 1 | 1 | *1* | *1* |

**I6** eredménye **R1** helyett ***S1*-ben** képződik (regiszter átnevezés)! A későbbi utasításokban **R1** helyett ***S1*-et** kell használni!

**I7** **RAW** és **WAW** függőség **R3** miatt (**I1**), **RAW** függőség **R1** (***S1***) miatt (**I6**), regiszter átnevezés miatt: **R3=R3*R1** helyett **R3=R3**S1**

**I8** **WAR** függőség: **R1**-et **I1**, **I3** olvassa, ***S1*-be I7** ír (**WAW**) ezért **R1=R4+R4** helyett ***S2*=R4+R4** (mostantól **R1** helyett ***S2*** kell).

| | | | | Olvasott regiszterek | | | | | | | | | | Irt regiszterek | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| C | # | Dekódolt | K | B | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | *1* | *2* | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | *1* | *2* |
| 1 | 1 | R3=R0*R1 | 1 | | 1 | 1 | | | | | | | | | | | | 1 | | | | | | |
| | 2 | R4=R0+R2 | 2 | | 2 | 1 | 1 | | | | | | | | | | | 1 | 1 | | | | | |
| 2 | 3 | R5=R0+R1 | 3 | | 3 | 2 | 1 | | | | | | | | | | | 1 | 1 | 1 | | | | |
| | 4 | R6=R1+R4 | - | | 3 | 2 | 1 | | 0 | | | | *RAW* | | | | | 1 | 1 | 1 | 0 | | | |
| 3 | 5 | R7=R1*R2 | 5 | | 3 | 3 | 2 | | 0 | | | | *I5 megelőzi I4-et* | | | | | 1 | 1 | 1 | 0 | 1 | | |
| | 6 | R1*(S1)*=R0-R2 | 6 | | 4 | 3 | 3 | | 0 | | | | *WAR: R1 helyett S1* | | | | | 1 | 1 | 1 | 0 | 1 | *1* | |
| | | | | 2 | 3 | 3 | 2 | | 0 | | | | | | | | | 1 | | 1 | 0 | 1 | *1* | |
| 4 | | | 4 | | 3 | 4 | 2 | | 1 | | | | | | | | | 1 | | 1 | 1 | 1 | *1* | |
| | 7 | R3=R3*R1*(S1)* | - | | 3 | 4 | 2 | 0 | 1 | | | | *RAW* | *1* | | | | 1 | | 1 | 1 | 1 | *1* | |
| | 8 | R1*(S2)*=R4+R4 | 8 | | 3 | 4 | 2 | 0 | 3 | | | | *WAR* | *1* | | | | 1 | | 1 | 1 | 1 | *1* | *1* |
| | | | | 1 | 2 | 3 | 2 | 0 | 3 | | | | | *1* | | | | 0 | | 1 | 1 | 1 | *1* | *1* |
| | | | | 3 | 1 | 2 | 2 | 0 | 3 | | | | | *1* | | | | 0 | | | 1 | 1 | *1* | *1* |
| 5 | | | | 6 | | 2 | 1 | 0 | 3 | | | | | *1* | | *0* | | 0 | | | 1 | 1 | *0* | *1* |
| 6 | | | 7 | | | 2 | 1 | 1 | 3 | | | | | *1* | | *0* | | 1 | | | 1 | 1 | | *1* |
| | | | | 4 | | 1 | 1 | 1 | 2 | | | | | *1* | | *0* | | 1 | | | | 1 | | *1* |
| | | | | 5 | | | | 1 | 2 | | | | | *1* | | *0* | | 1 | | | | | | *1* |
| | | | | 8 | | | | 1 | | | | | | *1* | | | | 1 | | | | | | |
| 7 | | (R1=*S2*) | | | | | | 1 | | | | | | *1* | | | | 1 | | | | | | |
| 8 | | | | | | | | 1 | | | | | | *1* | | | | 1 | | | | | | |
| 9 | | | | 7 | | | | | | | | | | | | | | | | | | | | |



| | | | | Olvasott regiszterek | | | | | | | | | | Irt regiszterek | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| C | # | Dekódolt | K | B | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | *1* | *2* | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | *1* | *2* |
| 6 | | | 7 | | | 2 | 1 | 1 | 3 | | | | *1* | | | *0* | | 1 | | | 1 | 1 | | *1* |
| | | | | 4 | | 1 | 1 | 1 | 2 | | | | *1* | | | *0* | | 1 | | | | 1 | | *1* |
| | | | | 5 | | | | 1 | 2 | | | | *1* | | | *0* | | 1 | | | | | | *1* |
| | | | | 8 | | | | 1 | | | | | *1* | | | | | 1 | | | | | | |
| 7 | | (R1=*S2*) | | | | | | 1 | | | | | *1* | | | | | 1 | | | | | | |
| 8 | | | | | | | | 1 | | | | | *1* | | | | | 1 | | | | | | |
| 9 | | | | 7 | | | | | | | | | | | | | | | | | | | | |

**I8** eredménye a 7. ciklusban átkerülhet ***S2*** -ből **R1**-be, de jobb, ha a hardver nyilvántartja, hogy hol van.

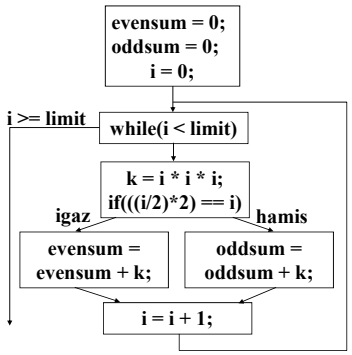
A modern **CPU**-k gyakran titkos regiszterek tucatjait használják regiszter átnevezésre, hogy ezáltal kiküszöbölék a **WAR** és **WAW** függőségeket.



**Feltételezett végrehajtás (4.45. ábra)**

**Páros és páratlan számok köbének összege:**

```
evensum = 0;
oddsum = 0;
i = 0;
while(i < limit) {
  k = i * i * i;
  if(((i/2)*2) == i)
    evensum = evensum + k;
  else
    oddsum = oddsum + k;
  i = i + 1;
}
```






**Feltételezett végrehajtás (4.45. ábra)**
**Speculative Execution**

**Alap blokk** (**basic block**): lineáris kód sorozat. Sokszor rövid, nincs elegendő párhuzamosság, hogy hatékonyan kihasználjuk.

**Emelés**: egy utasítás előre hozatala egy elágazáson keresztül (lassú műveletek esetén nyerhetünk vele). Pl. evensum és oddsum regiszterbe tölthető az elágazás előtt. Az egyik **LOAD** – természetesen – fölösleges.

Ha valamit nem biztos, hogy meg kell csinálni, de nincs más dolga a gépnek, akkor megteheti, de csak „firkáló" regiszterekbe írhat. Ha később kiderül, hogy kell, akkor átírja az eredményeket a valódi regiszterekbe, ha nem kell, elfelejti.



**Feltételezett végrehajtás (Speculative Execution)**

**Mellékhatások:**

- fölösleges gyorsító sor csere, **`SPECULATIVE_LOAD`**: megpróbálja a betöltést a gyorsító tárból. Ha nincs ott, akkor föladja.
- ***`if(x>0) z=y/x;`*** Ha az osztást a feltétel ellenőrzése előtt végzi el a processzor (emelés), akkor ***`x=0`*** esetén csapda következik be 0-val való osztás miatt.
  Megoldás: **mérgezés bit**. Emelés esetén a csapda csak akkor következik be, ha az emelt műveletet tényleg végre kell hajtani.



**Pentium 4** (2000. november)

Felülről kompatibilis az **I8088**, **…**, **Pentium III**-mal.

29.000, …, 42 → 55 M tranzisztor, 1,5 → 3,2 GHz, 63-82W, 478 láb (**3. 44. ábra**), 32 bites gép, 64 bites adat sín.

**NetBurst** architektúra.

2 fixpontos **ALU**. Mindkét **ALU** kétszeres órajel sebességgel fut.

Többszálúság (hyperthreding): 5% többlet a lapkán ~ két **CPU**.

## Többszálúság (hyperthreding, 8.7. ábra)

| | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|
| (a) | A1 | A2 | | | A3 | A4 | A5 | | | A6 | A7 | A8 |
| (b) | B1 | | | B2 | | | B3 | B4 | B5 | B6 | B7 | B8 |
| (c) | C1 | C2 | C3 | C4 | | | C5 | C6 | | | C7 | C8 |

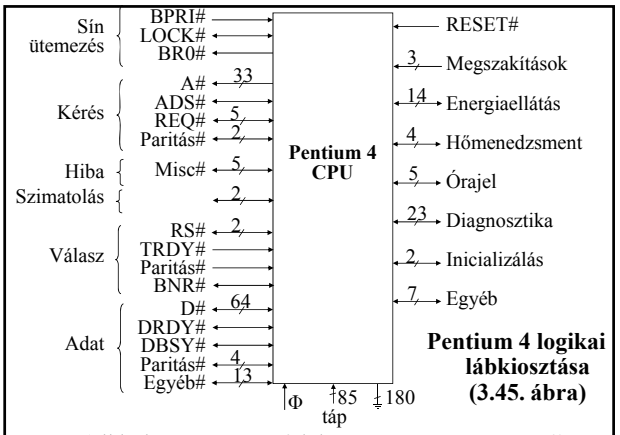
Óraciklus →

Az (a), (b) és (c) processzus külön futtatva az üres téglalapoknál várakozni kényszerül a memóriához fordulások miatt.

A többszálúság többszörözött regiszter készlet és némi szervező hardver hozzáadásával valósítható meg:

| EGYÜTT | A1 | B1 | C1 | A2 | B2 | C2 | A3 | B3 | C3 | A4 | B4 | C4 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|






**Pentium 4 logikai lábkiosztása (3.45. ábra)**



## Pentium 4 logikai lábkiosztása (3.45. ábra)

**RESET#**: a CPU alapállapotba hozatala,

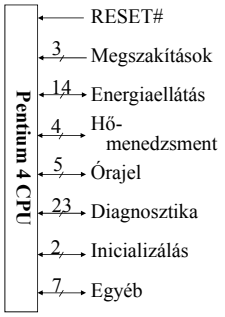
**Megszakítások**: régi vezérlő, és Advanced Programmable Interrupt Controller (**APIC**)

Különböző tápfeszültségek, alvási állapotok,

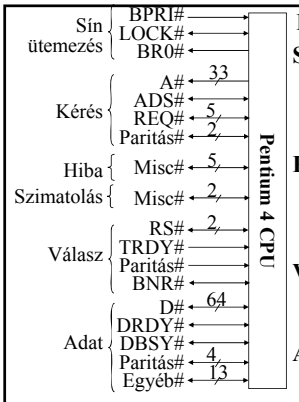
Jelzés $130^0$ fölött, …

Rendszersín frekvenciája,

**…**






**(3.45. ábra)**

**Pentium 4 logikai lábkiosztása**

**Sín ütemezés: BPRI#**: magas, **BR0#**: normál prioritású igény **LOCK#**: sín foglalás több ciklusra,

**Kérés: A#:** 8 bájtos adat 36 bites címe (64 GB címezhető), **ADS#**: a cím érvényes, **REQ#**: kívánság,

**Válasz: RS#:** státus, **TRDY#**: a szolga tud adatot fogadni, **BNR#:** WAIT

**Adat: D#:** 8 bájtos adat, **DRDY#**: az adat a sínen van, **DBSY#**: a sín foglalt.



## Pentium 4

Gépi utasítások → **RISC** szerű mikroutasítások, több mikroutasítás futhat egyszerre: szuperskaláris gép, megengedi a sorrenden kívüli végrehajtást is.



## Pentium 4

2-3 szintű belső gyorsító tár.

**L1**: **8 KB** adat, 4 utas halmaz kezelésű, írás áteresztő, 64 bájtos gyorsító sor.
**8 KB** utasítás + nyomkövető tár akár 12000 dekódolt mikroutasítás tárolására.

**L2**: **256 KB – 1 MB**, egyesített, 8 utas halmaz kezelésű, késleltetve visszaíró, 128 bájtos gyorsító sor. Előre betöltő egység.

Az Extrem Edition-ban **2 MB** (közös) **L3** is van.

Multiprocesszoros rendszerekhez szimatolás - snoop.

## Szimatolás – snoop

Minden processzor figyeli a sínen a többi processzor memóriához fordulásait (szimatol). Ha valamelyik processzor olyan adatot kér, amely bent van a gyorsító tárában, akkor a gyorsító tárából megadja a kért adatot, és letiltja a memóriához fordulást.



## Szimatolás – snoop

Ha minden processzornak **saját** írás áteresztő gyorsító tára van (**8.25. ábra**)

| Esemény | Saját gyorsító tár | Többi gyorsító tár |
|---|---|---|
| Olvasás hiány | Olvasás a memóriából | szimatolás |
| Olvasás találat | Olvasás a gyorsító tárból | |
| Írás hiány | Írás a memóriába | Ha az írandó szó a gyorsító tárban van, akkor érvényteleníti a gyorsító tár bejegyzést |
| Írás találat | Írás a gyorsító tárba és a memóriába | |

A **Pentium 4** esetén **L1** az írás áteresztő gyorsító tár, a késleltetve visszaíró **L2** a „memória”.



## Pentium 4 memória sín

A memóriaigények, tranzakciók 6 állapota: 6 fázisú csővezeték (**3.45. ábra** bal oldal) fázisonként külön vezérlő vonalakkal (amint a mester megkap valamit, elengedi a vonalakat):

0. **Sín ütemezés (kiosztás, bus arbitration)**: eldől, hogy melyik sínmester következik,
1. **Kérés**: cím a sínre, kérés indítása,
2. **Hibajelzés**: a szolga hibát jelez(het),
3. **Szimatolás**,
4. **Válasz**: kész lesz-e az adat a következő ciklusban,
5. **Adat**: megvan az adat.



## Pentium 4 memória sín csővezetéke (3.46. ábra)

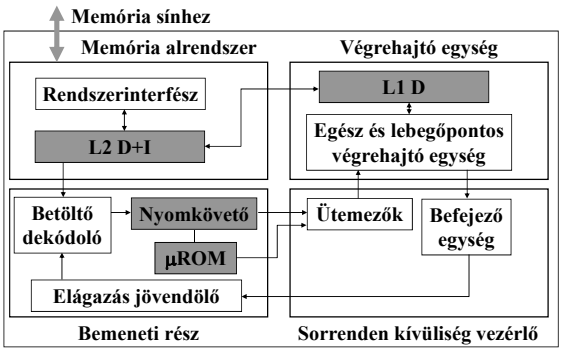
| Φ: tranzakció | $T_1$ | $T_2$ | $T_3$ | $T_4$ | $T_5$ | $T_6$ | $T_7$ | $T_8$ | $T_9$ | $T_{10}$ | $T_{11}$ | $T_{12}$ |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 1 | K | H | S | V | A | | | | | | | |
| 2 | | K | H | S | V | A | | | | | | |
| 3 | | | K | H | S | V | A | | | | | |
| 4 | | | | K | H | S | V | | A | | | |
| 5 | | | | | K | H | S | V | | A | | |
| 6 | | | | | | K | H | S | | V | A | |
| 7 | | | | | | | K | H | S | | V | A |

Ütemezés (nem ábrázoltuk), csak akkor kell, ha másé a sín.

**K**: kérés, **H**: hiba, **S**: szimatolás, **V**: válasz, **A**: adat



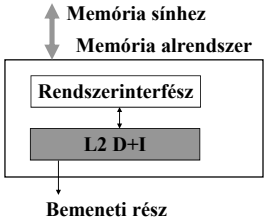
## A Pentium 4 mikroarchitektúrája




**4.46. ábra.** A Pentium 4 blokkdiagramja



**4.46. ábra**. A Pentium 4 memória alrendszere



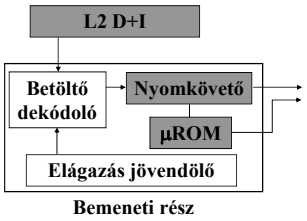

**L2** 256 KB az első, 512 KB a második, 1 MB a harmadik generációs Pentium 4-ben.

**L2** 8 utas halmaz kezelésű, késleltetve visszaíró, 128 bájtos gyorsító sor, minden második ciklusban kezdődhet egy 64 bájtos feltöltés a memóriából.

Előre betöltő: megpróbálja **L2**-be tölteni azt a gyorsító sort, amelyre majd szükség lesz (nincs az ábrán).

**4.46. ábra.** A Pentium 4 bemeneti rész

**L2**-ből betölti és dekódolja a programnak megfelelő sorrendben az utasításokat. Az utasításokat **RISC** szerű mikroműveletek sorozatára bontja. A dekódolt mikro-műveletek a **Nyomkövető**be kerülnek (nem kell újra dekódolni). Ha több, mint 4 mikroművelet szükséges, akkor **µROM**-ra történik utalás.




Elágazás jövendölés.



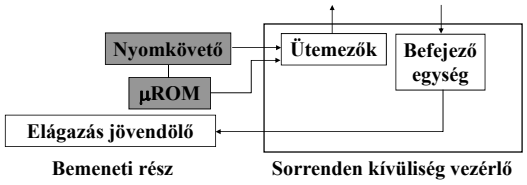
A bemeneti rész az utasításokat **L2**-ből kapja. Ezeket dekódolja, **RISC** szerű mikroműveletekre bontja, a nyomkövető gyorsító tárban tárolja (akár 12 K mikroműveletet) a programnak megfelelő sorrendben.
6 mikroműveletet csoportosít minden nyomkövető sorban.

Feltételes elágazásnál az utolsó **4 K** elágazást tartalmazó **L1 BTB**-ből (Branch Target Buffer – elágazási cél puffer) kikeresi a jövendölt címet, és onnan folytatja a dekódolást. Ha az elágazás nem szerepel **L1 BTB**-ben, akkor statikus jövendölés történik: visszafelé ugrást végre kell hajtani, előre ugrást nem.



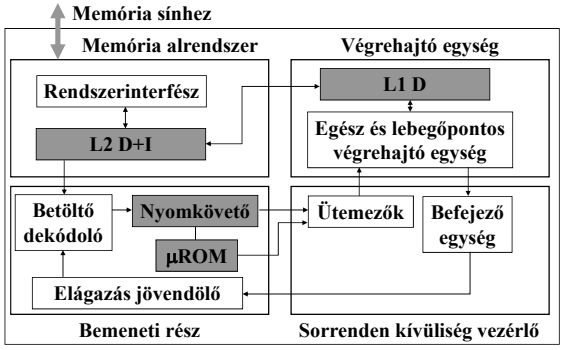
**4.46. ábra.** Sorrenden kívüliség vezérlő

Az utasítások a programnak megfelelő sorrendben kerülnek az ütemezőbe, eltérő sorrendben kezdődhet a végrehajtásuk (esetleg regiszter átnevezéssel), de a pontos megszakítás követelménye miatt az előírt sorrendben fejeződnek be.





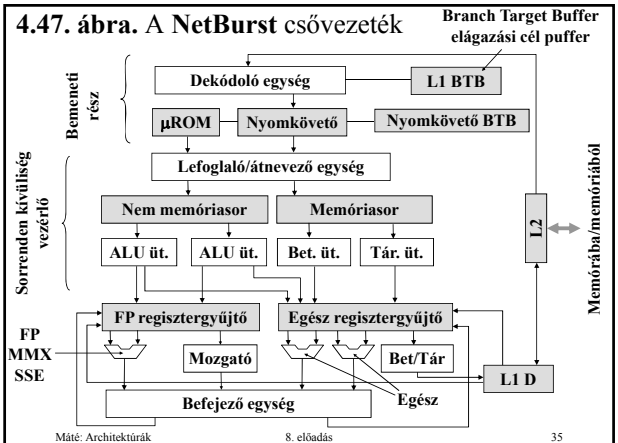

**A Pentium 4 mikroarchitektúrája**



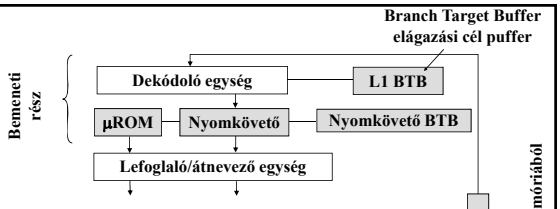

**4.46. ábra.** A Pentium 4 blokkdiagramja



**4.47. ábra.** A **NetBurst** csővezeték





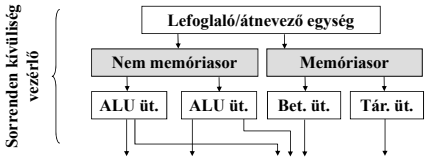




A dekódoló egység az utasításokat **L2**-ből kapja. Ezeket **RISC** szerű mikroműveletekre bontja, és a nyomkövető gyorsító tárban tárolja. Innen ciklusonként három mikroművelet kerül a lefoglaló/átnevező egység **ROB** (ReOrder Buffer, átrendező puffer) nevű táblájába. Ez a tábla 128 bejegyzést tartalmazhat.

Ha egy mikroművelet minden inputja rendelkezésre áll, akkor az esetleges **WAR** vagy **WAW** függőséget a 120 firkáló regiszter segítségével kiküszöböli. **RAW** függőség esetén a mikroműveletet várakoztatja, és a rákövetkező mikroműveleteket kezdi feldolgozni.
Egyszerre akár 126 utasítás feldolgozása is folyamatban lehet, köztük 48 betöltés és 24 tárolás.
Az utasítások a programnak megfelelő sorrendben kerülnek az ütemezőbe, eltérő sorrendben kezdődhet a végrehajtásuk, de az előírt sorrendben fejeződnek be.
**Pontos megszakítás**: a megszakítás előtti összes utasítás befejeződött, az utána következőkből egy sem kezdődött el.



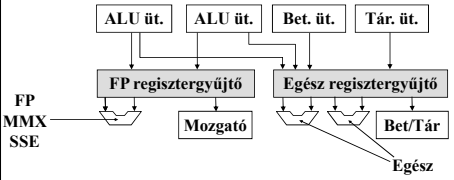
A **Lefoglaló/átnevező egység** a két sor megfelelőjébe teszi a végrehajtható mikroutasításokat. Az ALU-k az órajel kétszeres sebességével dolgoznak, nehéz folyamatosan munkát adni nekik.




Minden órajel ciklusban egy betöltés és egy tárolás is végrehajtható.

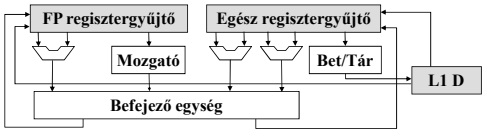


Mindkét regisztergyűjtő 128 regisztert tartalmaz, időben változik, hogy melyikben van **EAX**, …
Az egyik egész aritmetikájú **ALU** az összes logikai, aritmetikai, és elágazó, a másik csak az összeadó, kivonó, léptető és forgató utasítás végrehajtására képes.



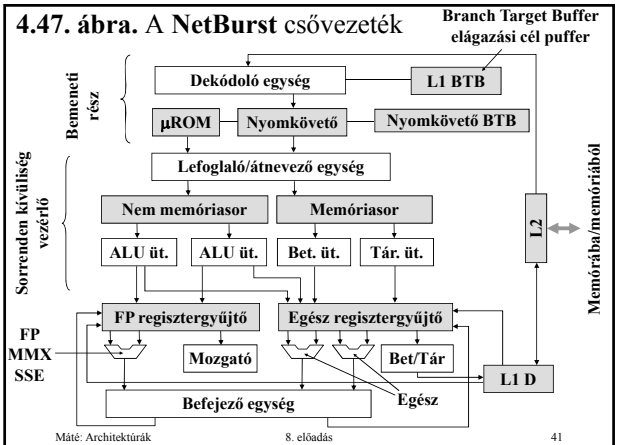



A befejező egység feladata, hogy az utasítások a programnak megfelelő sorrendben fejeződjenek be.
**L1** 4 utas halmazkezelésű, írás áteresztő gyorsítótár 64 bájtos gyorsító sorral. Nem lehet **L1**-et módosítani, amíg a tárolást megelőző műveletek be nem fejeződtek (24 bejegyzéses tároló puffer), de ha egy betöltő utasítás onnan akar olvasni, ahova egy korábbi tárolt, akkor a tárolások pufferéből megkaphatja a kért adatot (tárolás utáni betöltés).






**4.47. ábra.** A **NetBurst** csővezeték






**UltraSPARC III** (2000)

64 bites **RISC** gép, felűről kompatibilis a 32 bites **SPARC V8** architektúrával és az **UltraSPARC I, II**-vel. Új a **VIS 2.0** utasításkészlet (3D grafikus alkalmazásokhoz, tömörítéshez, hálózat kezeléshez, jelfeldolgozáshoz, stb.).

Több processzoros alkalmazásokhoz készült. Az összekapcsoláshoz szükséges elemeket is tartalmazza.

2000-ben 0.6, 2001-ben 0.9, 2002-ben 1.2 GHz, órajel ciklusonként 4 utasítás elvégzését tudja indítani.

## UltraSPARC III

**CPU** 29 millió tranzisztor, 4 **CPU** közös memóriával használható. 1368 láb (**3. 47. ábra**). 64 (jelenleg csak 43) bites cím és 128 bites adat lehetséges (több helyen ellentmondó adatok vannak a könyvben, az új kiadásban néhol bennmaradtak az **UltraSPARC II**-re vonatkozó adatok).

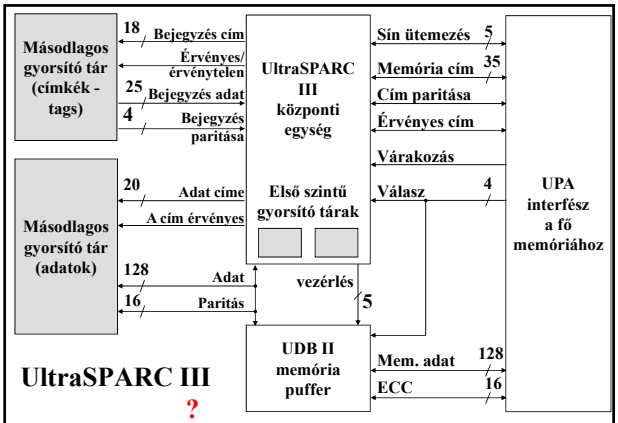
**Belső gyorsító tár** (**32 KB** utasítás + **64 KB** adat, gyorsító sor **32 B**).

**2 KB** előre betöltő és tároló gyorsító tár **L2** eléréséhez.

Az **L2** gyorsító tár osztott, külső **1**, **4** vagy **8 MB**

A gyorsító sor mérete **64**, **256** illetve **512 B**







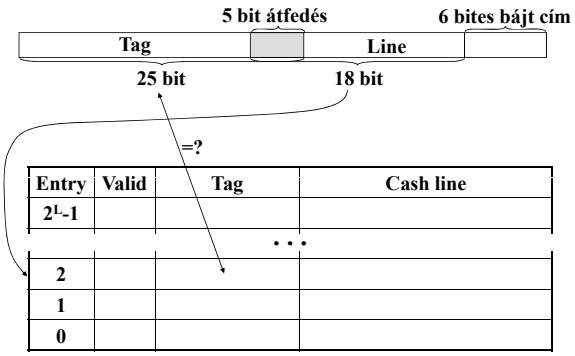

Az **UltraSPARC II**-nél **0.5-16 MB**-os az **L2** tár és **8 K - 256 K** db **64 B**-os gyorsító sor (cache line) lehet.

A gyorsító sor címzéséhez **13 – 18** bit szükséges.

A **CPU** mindig **18** bites **Line** címet (**Bejegyzés cím**) ad át. Csak maximális méret esetén van kihasználva mind a **18** bit címzésre.



A cím 64 bit-es, de egyelőre 44 bit-re korlátozva van





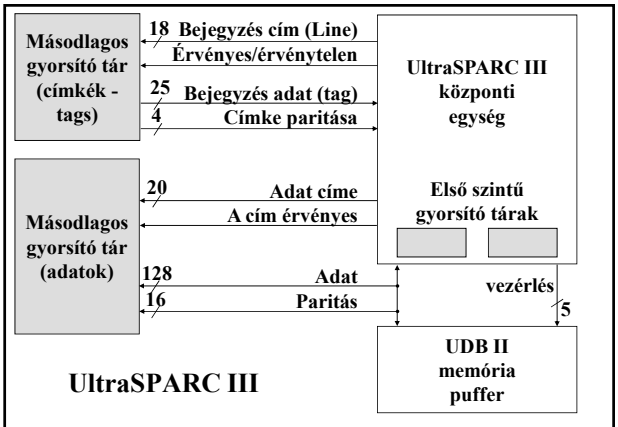

**512 KB**-os gyorsító tár esetén a **44** bites cím felosztása:

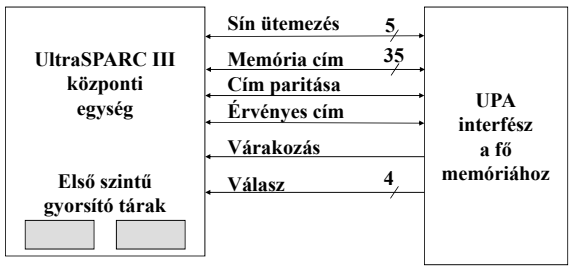
**Tag: 25** bit, **Line: 13** bit, **bájt cím: 6** bit = **44** bit.

**16 MB**-os tár esetén **18** bites **Line** kell, és **20** bites **Tag** (**Bejegyzés adat**) is elég lenne, de ilyekor – hogy a **CPU** egységesen működhessen – a gyorsító tárban tárolt **20** bites **Tag**-et a gyorsító tár kiegészíti **Line 5** legmagasabb helyértékű bitjével.

Az **Adat címe** a gyorsító sor címén (**Bejegyzés cím**, **Line**) kívül még **2** bitet tartalmaz, mert egy átvitel során a gyorsító sornak csak negyed része (**16** bájt) mozgatható.



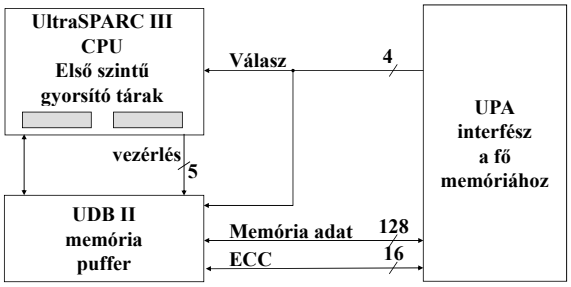




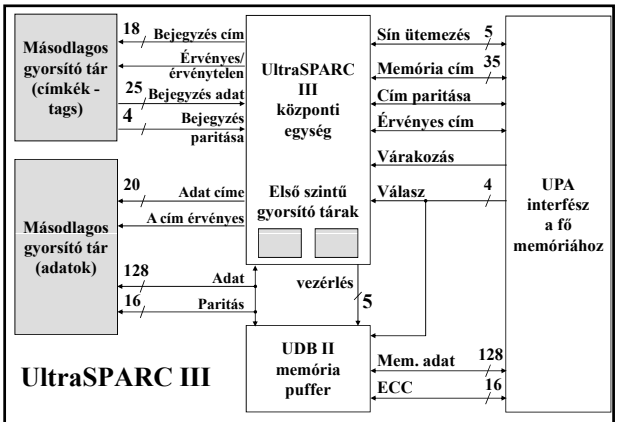

**UPA** (Ultra Port Architecture) sín, hálózati csomópont vagy a kettő kombinációja. Több **CPU** esetén egy központi vezérlőn keresztül kapcsolódnak a sínhez. Több írást és olvasást tud egyidejűleg kezelni.



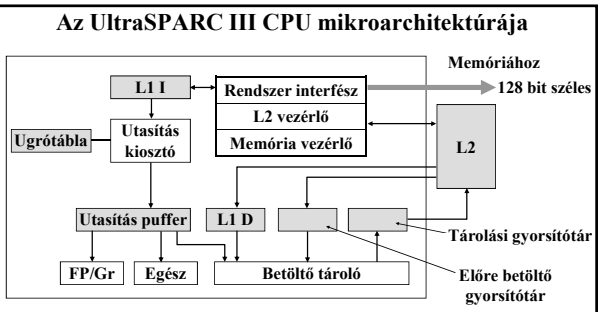
**UDB II** (UltraSPARC Data Buffer II): ezen keresztül zajlik a memória és a gyorsító tárak közötti adatforgalom. Az adatsín **150 MHz**-es 128 bit széles szinkron sín, így a sávszélesség 2.4 GB/s.





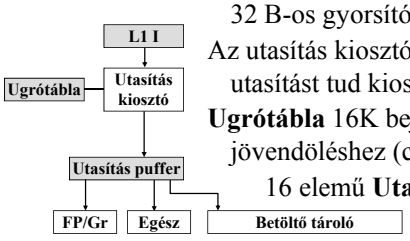






**4.48. ábra.** Az UltraSPARC III CPU blokkdiagramja






**L1 I** 32 KB 4 utas halmazkezelésű, 32 B-os gyorsító sor.

Az utasítás kiosztó ciklusonként 4 utasítást tud kiosztani

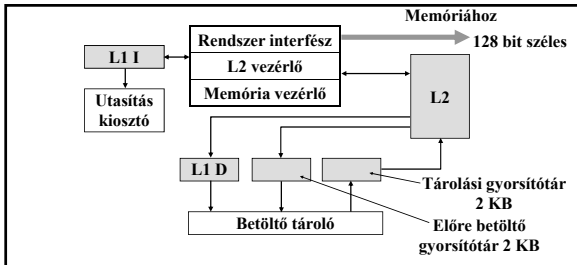
**Ugrótábla** 16K bejegyzés elágazás jövendöléshez (cél cím is)

16 elemű **Utasítás puffer**

Két egész aritmetikájú ALU + regiszterek + firkáló regiszterek

Lebegőpontos egység: 32 regiszter, összeadó/kivonó + szorzó + osztó ALU, grafikai utasítások






**L1 D** 64 KB-os 4 utas halmazkezelésű, írás átíresztő, 32 bájtos gyorsító sor. Feltételezett betöltésre 2 KB előre betöltő gyorsítótár. 2 KB tárolási gyorsítótár.

**Memória vezérlő**: virtuális → fizikai cím.

## UltraSPARC III CPU mikroarchitektúrája

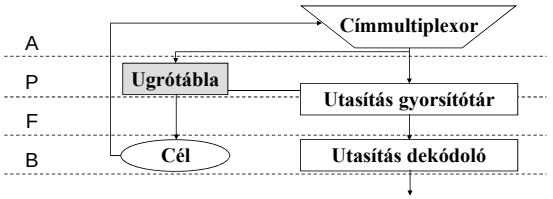
A **SPARC** sorozat **RISC** elgondoláson alapul. A legtöbb utasításnak két forrás és egy cél regisztere van.

Előre betöltés:
speciális utasításokkal,
és a visszafelé kompatibilitás miatt hardveresen is.

2 bites elágazás jövendölő + statikus elágazás jövendölés.



## UltraSPARC III csővezetéke (4.49. ábra)



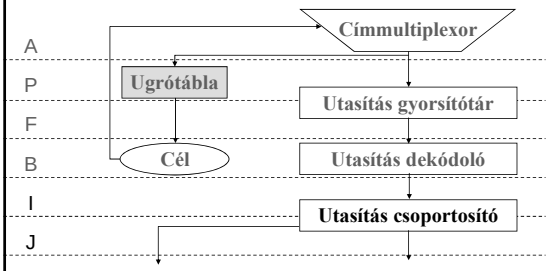

**A**ddress generation, cím generáló. Ugrás, csapda, … Az eltolás résben lévő utasítást mindig végrehajtja!

**P**reliminari **F**etch, előzetes betöltő. Legfeljebb 4 utasítást képes betölteni **L1 I**-ből, nézi, hogy van-e köztük elágazó, elágazás jövendölés.

**B**ranch target, elágazási cél. Ha kell ugrani, → **A**



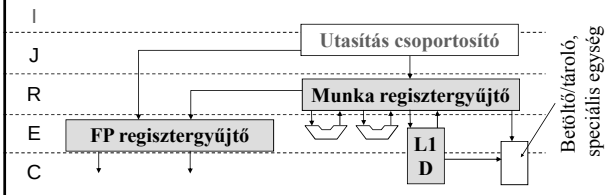
## UltraSPARC III csővezetéke




**I**nstruction group formation, utasítás csoportosító. Aszerint csoportosítja az utasításokat, hogy melyik működési egységet használják.



## UltraSPARC III csővezetéke

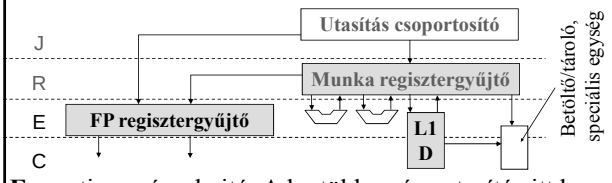



**J** instruction stage grouping, utasítás kiosztó. Az elérhető működési egységektől függően akár 4 utasítást is továbbít az **R** szakasznak.

**R**egister, függőség esetén vár, nincs sorrenden kívüli végrehajtás.



## UltraSPARC III csővezetéke

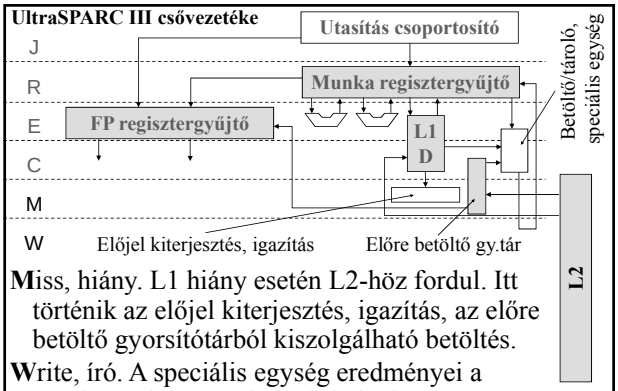



**E**xecution, végrehajtó. A legtöbb egész utasítás itt be is fejeződik. Ha egy utasítás készen van, akkor frissül a munka regisztergyűjtő.
Itt dől el, hogy az ugrás feltétele teljesül-e. Hibás jövendölés esetén jelzés az **A** szakasznak, a csővezeték érvénytelenítése.

**C**ache, gyorsítótár. Itt történik az olvasás **L1 D** –ből.



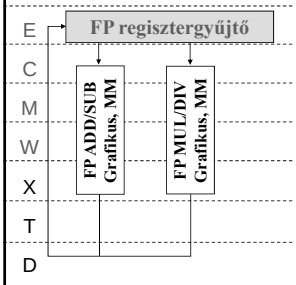
## UltraSPARC III csővezetéke




**M**iss, hiány. L1 hiány esetén L2-höz fordul. Itt történik az előjel kiterjesztés, igazítás, az előre betöltő gyorsítótárból kiszolgálható betöltés.

**W**rite, író. A speciális egység eredményei a munka regisztergyűjtőbe kerülnek.

**UltraSPARC III csővezetéke**




e**X**tended, kiterjesztett. Itt fejeződik be a legtöbb FP és grafikai utasítás.

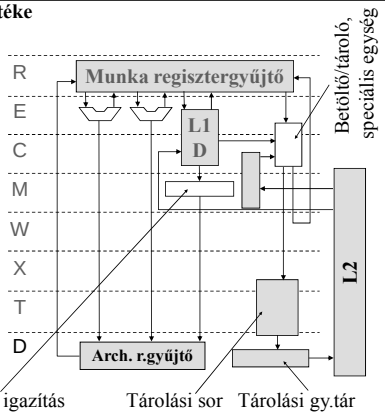
**T**rap, csapda. Ez észleli az egész és FP csapdákat.

Pontos megszakítás.



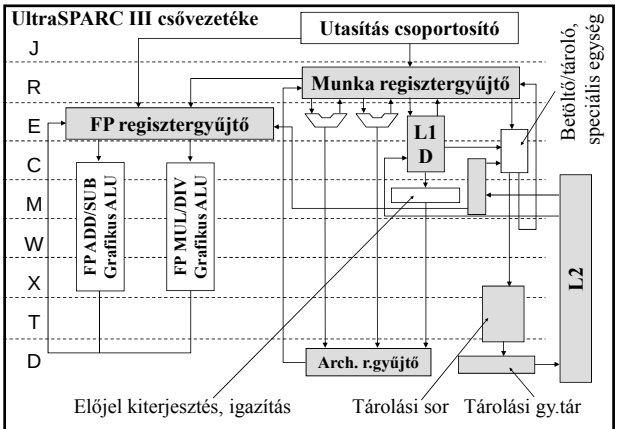
**UltraSPARC III csővezetéke**

A **D** szakasz véglegesíti a regiszterek értékét az architektúrális regiszter gyűjtőben. Megszakításkor az itteni adatok érvényesek.






**UltraSPARC III csővezetéke**






## Feladatok

Mit nevezünk függőségnek?

Milyen függőségeket ismer?

Mely függőségek oldhatók fel, és hogyan?



## Feladatok

Mi az előnye a sorrendtől eltérő végrehajtásnak?

Mire szolgál a regiszter átnevezés?

Mi a feltételezett végrehajtás?

Mit nevezünk emelésnek?

Mikor előnyös az emelés?

Milyen mellékhatásai lehetnek a feltételezett végrehajtásnak?

Mi a **`SPECULATIVE_LOAD`** lényege?

Mi a mérgezés bit?



## Feladatok

Mi a többszálúság lényege, haszna?

Mik a többszálúság megvalósításának feltételei?

Hogy érvényesül a RISC elv a Pentium 4 esetén?

Mi a szuperskaláris gép lényege?

Mit jelent a sorrenden kívüli végrehajtás?

Milyen gyorsítótárakat használ a Pentium 4?

Jellemezze a Pentium 4 L2 gyorsítótárát!

Mire szolgál az előre betöltő?

Mit jelent a szimatolás?

Milyen sorrendben dekódolja a Pentium 4 az utasításokat?

Mire szolgál a **µROM**?

**Feladatok**

Mire szolgál a nyomkövető gyorsítótár?
Milyen elágazás jövendölést használ a Pentium 4?
Mire szolgál az **L1 BTB**?
Mire szolgál a nyomkövető **BTB**?
Milyen sorrendben kezdődik az utasítások végrehajtása a Pentium 4-en?



**Feladatok**

Mire szolgál a Pentium 4 lefoglaló/átnevező egysége?
Mire szolgálnak a regiszter gyűjtők?
Milyen sorrendben fejeződik be az utasítások végrehajtása a Pentium 4-en?
Mi a különbség a Pentium 4 két egész aritmetikájú **ALU**-ja között?
Miért nem írható azonnal az eredmény L2-be?
Mit jelent a pontos megszakítás kifejezés?
Milyen problémát okozhat a tárolás utáni betöltés?



**Feladatok**

Hogy működik az UltraSPARC III másodlagos gyorsítótára?
Mire szolgál az UPA (Ultra Port Architecture)?
Mire szolgál az UDB II (UltraSPARC Data Buffer II)?
Milyen szervezésű az UltraSPARC III **L1 I** gyorsítótára?
Mire szolgál a munka regisztergyűjtő?
Mire szolgál az architektúrális regisztergyűjtő?
Mire szolgál az előre betöltő gyorsítótár?
Mire szolgál a tárolási sor?
Mire szolgál a tárolási gyorsítótár?



**Feladatok**

Mire szolgál az UltraSPARC III ugrótáblája?
Milyen elágazás jövendölést használ az UltraSPARC III?
Mit nevezünk eltolás résnek?
Hogy kezeli az UltraSPARC III az eltolás rést?
Mire szolgál az utasítás csoportosító egység?
Hány ALU van az UltraSPARC III-ban?
Hogy kezeli az UltraSPARC III a függőségeket?



**Az előadáshoz kapcsolódó**

**Fontosabb tételek**

Sorrendtől eltérő végrehajtás, szuperskaláris architektúra, függőségek, regiszter átnevezés
Feltételezett végrehajtás
A Pentium 4 processzor, a Pentium 4 mikroarchitektúrája
A NetBurst csővezeték
Az UltraSPARC III processzor és az UltraSPARC III mikroarchitektúrája, csővezetéke