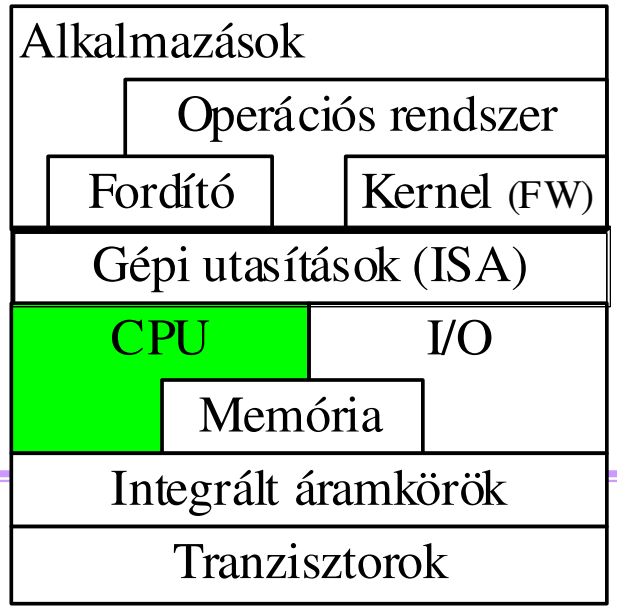
INFORMATIKA I.

BMEVIIIAB08

CPU (PIPE-LINE)



Számítógép teljesítmény növelése

- ***Mérő algoritmussal*** */Benchmark/* ***mérve*** */Már ismerjük/*

$$t_{MA} = I_N(A) \cdot C_N(I) \cdot T(C)$$

Utasításonként számolva!

$$t_{MA} = \left[\sum_{i=1}^{n} I_{iN} \cdot C_{iN} \right] \cdot T(C)$$

$$\mathbf{P} = \frac{1}{\mathbf{t}_{MA}} = \frac{IPC \cdot f}{I_N(A)}$$

C_N (CPI) $\frac{1}{CPI} = \frac{utasítás}{ciklus}$ / IPC / $\frac{1}{T} = f$

- **Egy rendszeren belül jó mérőszám**
  - **Sebesség növekedés** *(Speed up)* S_U
    valaminek a hatására, mennyivel gyorsabb

$$S_U = \frac{t_{MA}}{t_{MAúj}}$$

  - **Utasítás kibocsájtás/*áteresztő képesség*** *(throughput)*
    TP [1ut./idő], *[MIPS]* $TP = \frac{1}{t_{IKi}}$

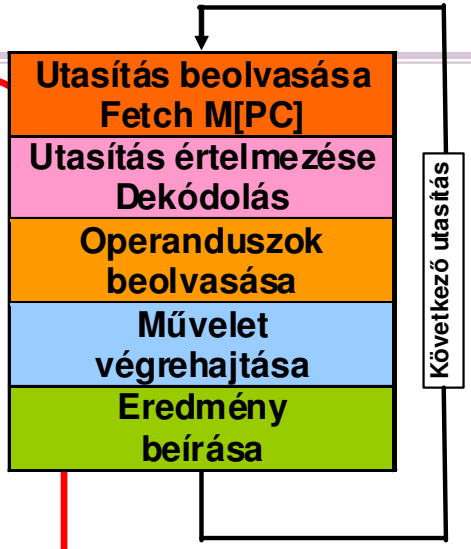
ha $t_{IKi} = t_I = C_N(I) \cdot T(C)$

$$TP = \frac{f}{C_N} = IPC \cdot f$$

$$P = \frac{IPC \cdot f}{I_N(A)} = \frac{TP}{I_N(A)}$$

Utasítás végrehajtás fázisai

Művelet	Gépi ciklus */fázis/*
■ Olvasás	**Külső** *Memória*
■ Értelmezés	**Belső** művelet
■ Operandus felhozatal	**Külső** olvasás *Memória, I/O*
■ Művelet végrehajtás	**Belső** művelet
■ Eredménybeírás	**Belső** írás *Regiszter* /**Külső** *Memória, I/O* /



- ➢ **A teljesítményt meghatározó tényezők**
  - ❑ **Technológia** /áramkörök sebessége/ ***f*↑ *határ?*** 👍 ✓
  - ❑ **Utasítás készlet** /*bonyolult-egyszerű?* / I_N↑☞C_N↑ ✓
  - ❑ **CPU belső felépítésétől,szervezésétől** ✓
  - ❑ **CPU – Mem- I/O kapcsolat szervezésétől**
  - ❑ **A számítógép szervezésétől** /***IPC***/

Számítógép teljesítmény növelése

$$I_N(A) \cdot C_N(I) \cdot T(C)$$

- **Feldolgozási sebesség fokozása**
  - **Utasítások számának csökkentése** I_N
  - **Utasítás végrehajtás gyorsítása** $C_N(I) \cdot T(C) = t_I$
  - **Több utasítás végrehajtásának párhuzamosítása** IPC
- **Utasítások számának csökkentése /*CISC*/** I_N
  - **Utasítás készlet** /*bonyolult-egyszerű?* / $I_N\downarrow$ $C_N\uparrow$
- **Utasítás végrehajtás gyorsítása** $TP = \frac{1}{t_{IKi}}$
  - **Technológia** /áramkörök sebessége ✓ / $f\uparrow$ 👍
  - **Utasítás gépi ciklusai számának csökkentése** $C_N(I)$
    - **Egyszerűbb utasítás** /*RISC*/ CPI↓ $I_N\uparrow$ $f\uparrow$/
    - **Regiszterek alkalmazása**✓CPI↓
    - **Adat/adatút bit-szám növelés** *8/16/32/64/128* **?** CPI↓ 👍
  - **Külső gépi ciklushossz rövidítése** 👍$f\uparrow$
    - **Memóriához fordulás idejének csökkentése**
    - **Hierarchikus memória kialakítása**

Feldolgozási sebesség fokozása - Külső gépi ciklushossz rövidítése

- Gyorsító /gyorstár/ *CACHE*
  - Hierarchikus gyorstár
    - » Gyorstár /CPU-n belül/ *szétválasztás önálló adatúttal* ✓
- Műveletvégzés gyorsítása
  - Speciális ALU ***(kombinációs hálózat)*** CPI↓ (→*külön egész- és-külön lebegőpontos műveletvégző)*
  - Memóriához fordulás-belső művelet párhuzamosítása
  - Belső átmeneti tároló puffer → QUEUE t_i↓
- Utasítás részműveleteinek párhuzamosítása *TP*↑
  - felhozatal
  - értelmezés
  - Végrehajtás
  - *PIPE-LINE /feldolgozó szalag/*

Amdahl törvénye *(Amdahl law)* *Gene Amdahl IBM 1967*

Adott /rész/feladat elvégzése sorosan (ts)

Párhuzamosan működő ndb. műveletvégzővel (tp)

$$S_U = \frac{t_S}{t_P} = \frac{t_S}{\frac{t_S}{n}} = n$$

Fizikailag inkább

$$S_U \le \frac{t_S}{t_P} = \frac{t_S}{\frac{t_S}{n}} = n$$

Ha csak x-ed részt párhuzamosítunk

$$S_U = \frac{t_S}{\frac{x \cdot t_S}{n} + (1-x) \cdot t_S} = \frac{n}{x + n \cdot (1-x)} = \frac{1}{\frac{x}{n} + (1-x)}$$

n növelésével

$$S_U \le \frac{1}{(1-x)}$$

90%-ot párhuzamosítunk x=0.9, n=10, Su=5.26

10%-ot párhuzamosítunk *x=0.1, n=90 Su=1.05!*

❑ *Hatékonyság* ***(párhuzamosítás)***

$$H_{Su} = \frac{Su}{n} \le 1$$

Amdahl törvénye /Amdahl law/ Gene Amdahl IBM 1967/

- *Következtetés utasításra*
  - **A sokat használt utasításokat érdemes gyorsítani**
    pl.:az utasítások x%-át n-szer gyorsabban
  - **Párhuzamos részműveletvégzés**

❑ Egyszerűbb utasításokat gyakrabban használják

– csak ezeket valósítják meg, a bonyolult utasítást több (**m**) utasítás helyettesíti → **RISC**

$$S_u = \frac{t_{MA}}{t_{MAúj}} = \frac{t_{MA}}{\frac{x \cdot t_{MA}}{n} + \frac{(1-x) \cdot m \cdot t_{MA}}{n}}$$

$$S_u = \frac{n}{x + m \cdot (1-x)} = \frac{1}{\frac{x}{n} + \frac{m}{n}(1-x)}$$

pl.: n=2, x=0.8, m=3 ***(bonyolult utasításoknál)***

$$\mathbf{Su} = \frac{2}{0.8+3 \cdot 0.2} = \frac{2}{1.4} = 1.42$$

RISC tulajdonságok összefoglalása ✓

/CDC 6600 1964 Seymour Cray, IBM 801 1980,

1981 Stanford University: MIPS John L. Hennessy MIPS: Microprocessor without Interlocked Pipeline Stages/

Berkely RISC I. ADAM D. Petterson

- Egyforma hosszú, *egyetlen ciklus /gépi/ alatt végrehajtható utasítások* CPI↓ →t_l↓ , f↑ t_l↓ ✓
- Tárolóhoz csak írás/olvasás utasítás (Load/Store ISA) CPI↓
- Nagy regisztertömb, regiszter referens utasítások ✓CPI↓ t_l↓
- Egyszerű címzés /gyors címszámítás/ CPI↓ t_l↓
- Huzalozott vezérlő egység CPI↓
  - Gyors f↑ t_l↓
- Megvalósításához kevesebb áramkör kell
- *A részműveletek egyszerűen párhuzamosíthatók*
  - Előnyös a pipe-line-hoz /C_N(I)=állandó/ *TP*↑

- szoftver végzi a bonyolult utasításokat
  - Kódoptimalizálást elősegítő fordító kell /egyszerű utasítások/

Szalagrendszerű műveletvégzés (*Pipe-line*)

- PALACSINTA SÜTÉS (Pipe-line projekt sok gyerekkel)

❑ *Erőforrás igény: 1db serpenyő, 1db főzőlap, 1db kezelő szervező (én)*

❑ *Műveletek*


tésztaöntés 40sec

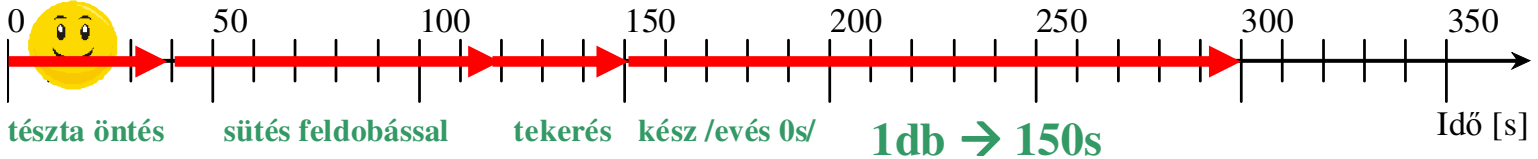

sütés 80s

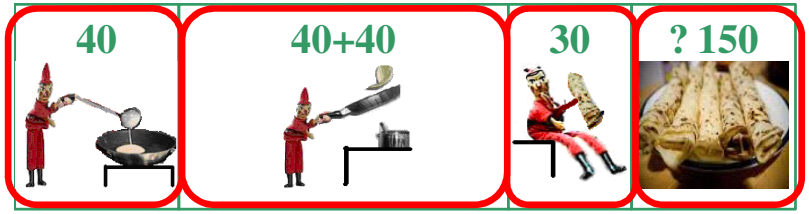

feltekerés 30s


elfogyasztás 0s

T=150s darabonként +150s 3db.=450s

❑ *Időzítés*





1db → 150s
2db → 300s

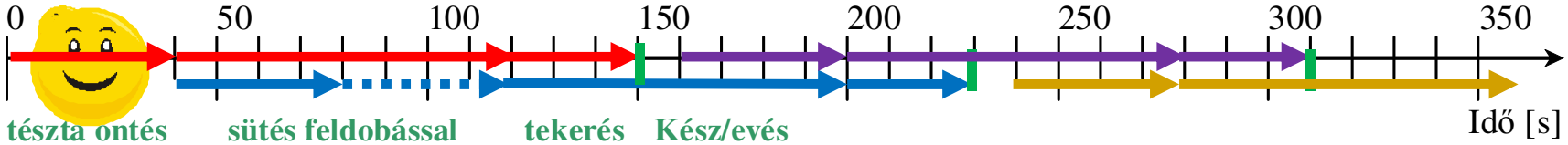
n db. palacsinta n x 150s
150s-onként lesz kész a következő
(Ez a palacsinta kibocsájtás)

Jönnek a gyerekek, jó étvágyuk van → fokozni kell a tempót

Szalagrendszerű műveletvégzés /Pipe-line/

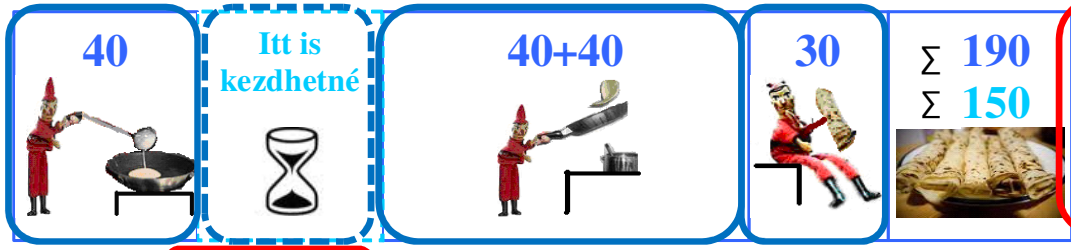
❖Párhuzamosítsuk a munkafázisokat, *amit lehet !* (művelet-feldolgozó szalag)

❑ *Erőforrás igény: 2db serpenyő /1tároló/, 1db főzőlap,*
feldolgozó szalag (öntő-sütő) feleség ***/Ő tud kétfelé figyelni/!***



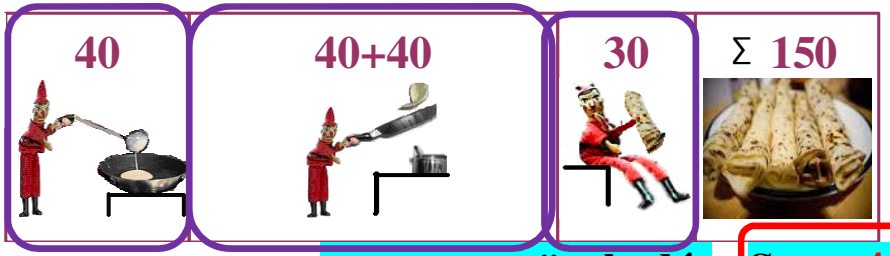
1db → 150s
2db → 230s
3db → 310s

80s-onként készül egy újabb



- **EREDMÉNY**
  100db hagyományosan 100x150s
  A jó szervezés (én)
  100db szervezéssel (150+99x80)s
- **Kibocsátás 1/80s**

Ez a mobilitás

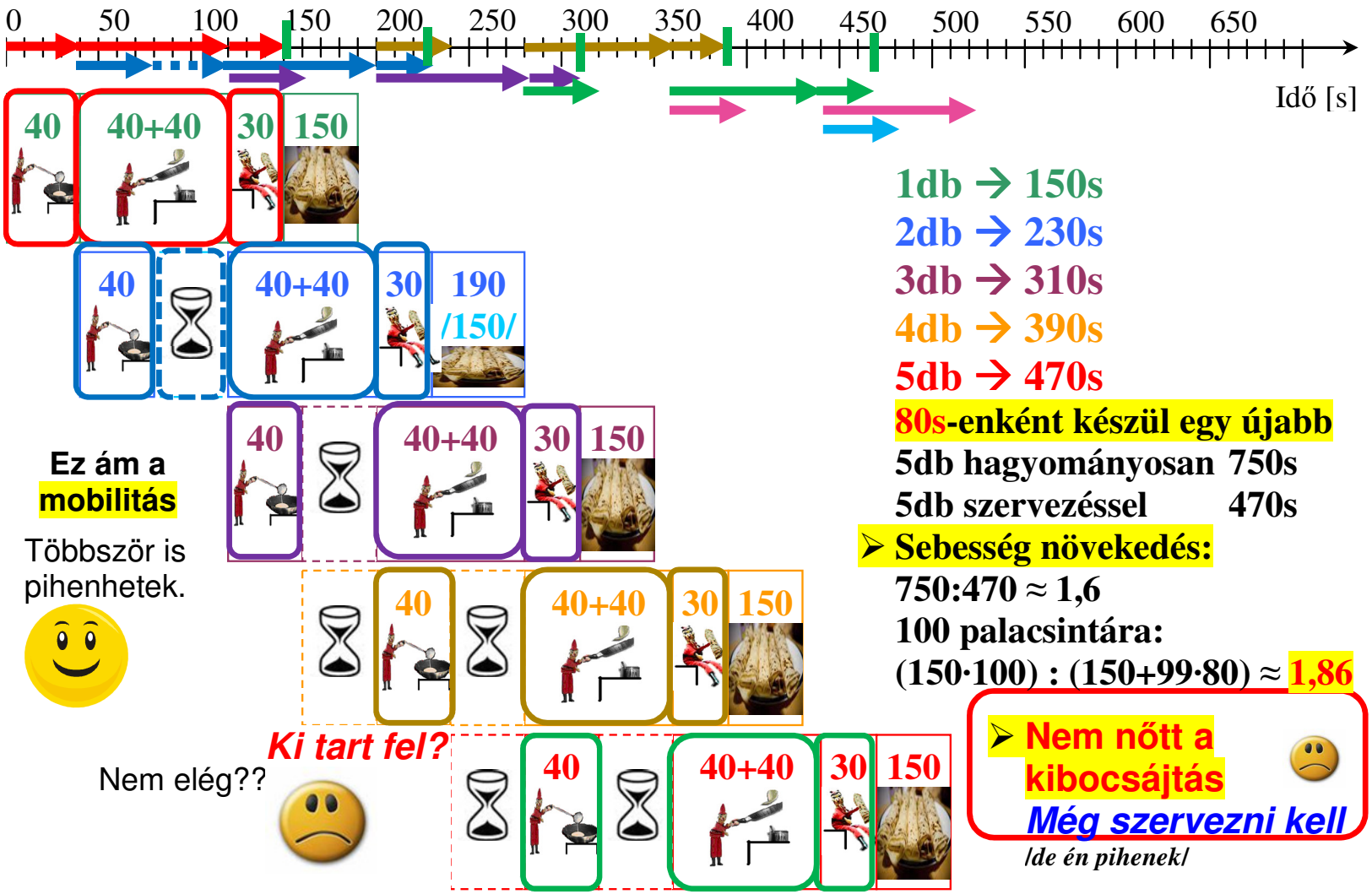


❖ $t_{100}:t_{100új}$=(150x100) : (150+99x80) → **Sebesség növekedés= Su: = 1,86**

Szalagrendszerű műveletvégzés /Pipe-line/

Jön a többi gyerek is, ***kevés a palacsinta*** ***besegítünk / a nagy gyerek meg én***

❑ *Erőforrás igény: 2db serpenyő, 1db főzőlap, öntő, sütő, tekerő/ feleség, gyerek, én/*



1db → 150s
2db → 230s
3db → 310s
4db → 390s
5db → 470s

80s-enként készül egy újabb

5db hagyományosan 750s
5db szervezéssel 470s

➢ **Sebesség növekedés:**
750:470 ≈ 1,6
100 palacsintára:
(150·100) : (150+99·80) ≈ 1,86

➢ **Nem nőtt a kibocsájtás**

Még szervezni kell
/de én pihenek/

!!!!MEGVAN!!

- Sokat várunk a sütésre/sütőlapra/ -*foglalt erőforrás*
- *A Sütés ideje túl hosszú a többihez képest*
  - *Bontsuk két részre*
  - *Használjunk 2db sütőlapot*
    /A két oldal sütéséhez** külön erőforrást **használunk/
  - *Párhuzamos műveletvégző egység*
    ☞2 utas feldolgozó szalag

❑Erőforrás igény: 3db serpenyő /1 tároló/, 2db főzőlap,

vezérlés /feleség, nagy gyerek, én/

1db → 150s
2db → 190s
3db → 230s
4db → 270s
5db → 310s

40s-enként készül egy újabb

5db hagyományosan 750s
5db szervezéssel 310s

➢ **Sebesség növekedés:**
750 : 310 ≈ 2,42
100 palacsintára:
(150 ·100) : (150+99·40) ≈ 3,65

➢ **Nőtt a kibocsájtás=1/40s**
/én nem pihenek/

➢ **Jó a műveleti idők kiegyensúlyozása**

➢ **Jó a koordináció** /Feleség 2 kézzel dolgozik/

➢ ***Nem 4x-es*** **a sebesség növekedés**
H_{Su} ?? /mert a gyerek pihen közben/

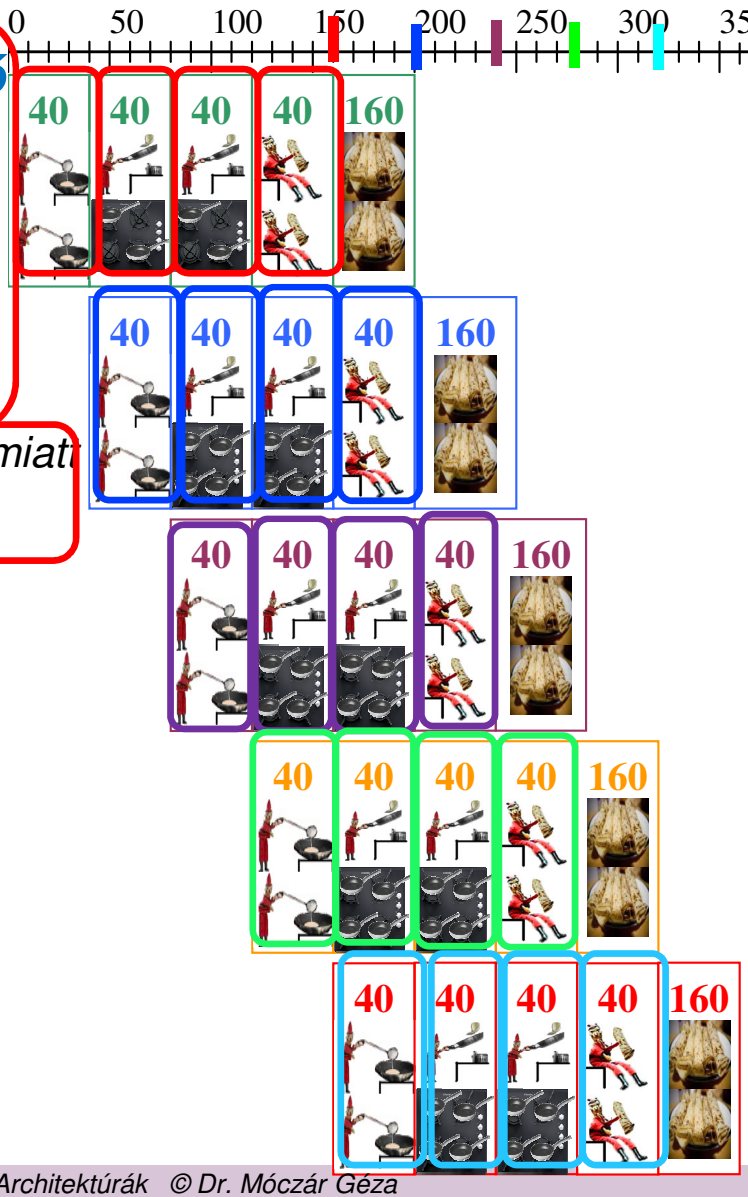
Műveletvégzés feldolgozó szalagon

- **Erőforrás: 4db főzőlap, 6db serp. 2 töltő, 2 sütő kezelő, 2 tekerő**

Palacsintagyártó üzem szervezése /már nagyon értünk hozzá/

6 serpenyő
2 tárol
4 süt
2 töltő
2 kezelő
2 tekerő

Csomagolás miatt ez is **40 sec**



2db → 160s
4db → 200s
6db → 240s
8db → 280s
10db → 320s

40s-enként készül két újabb

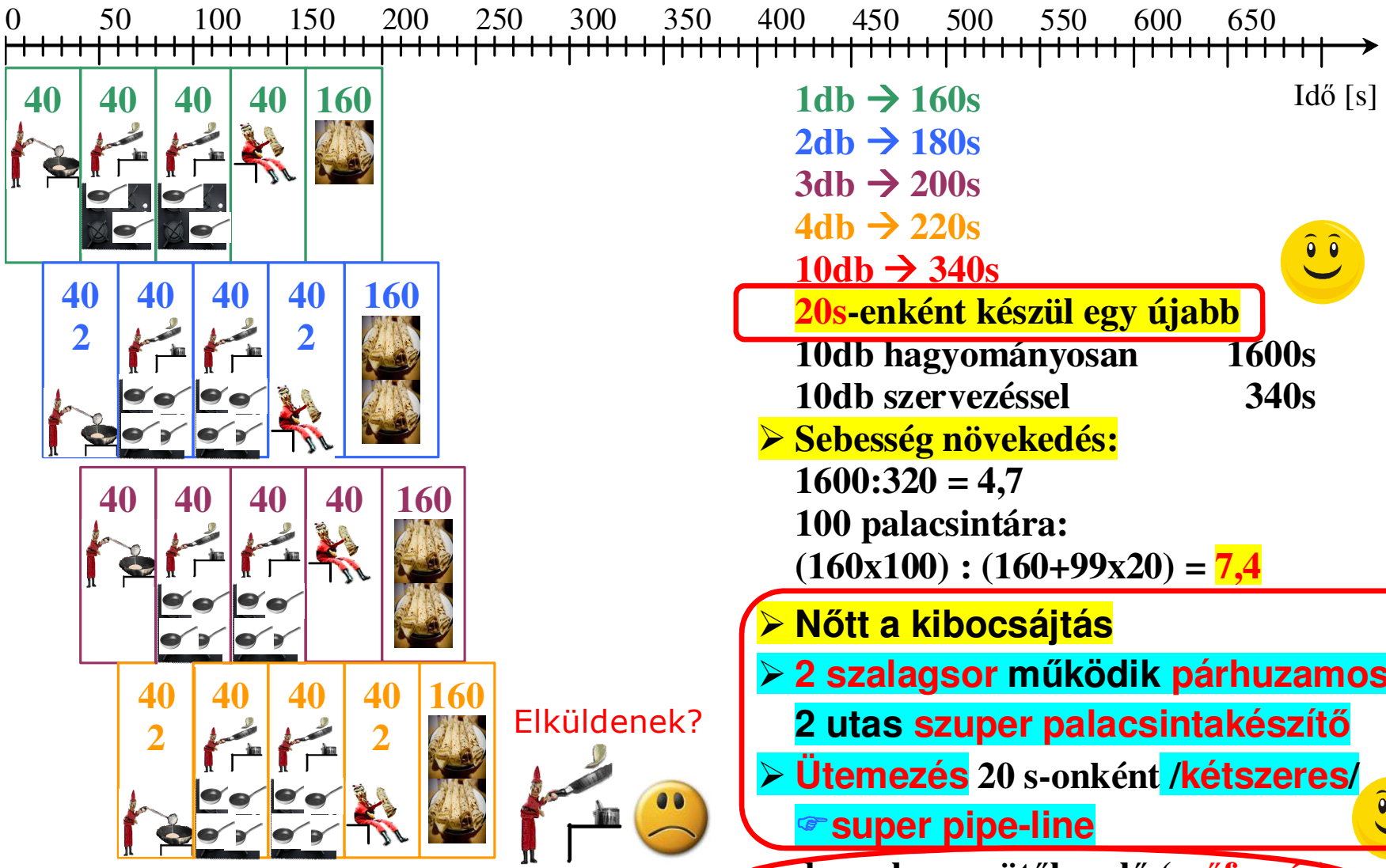
10db hagyományosan 1600s
10db szervezéssel 320s

- **Sebesség növekedés:**
  1600:320 = 5
  100 palacsintára:
  (160·100) : (160+48·40) ≈ 7,7

- **Nőtt a kibocsájtás=2/40s**
- **2 szalagsor működik párhuzamosan**
  /2 utas palacsinta pipe-line/

Műveletvégzés feldolgozó szalagon

- **Erőforrás: 4db főzőlap, 6db. serpenyő, 2 töltő, 1 sütő kezelő, 2 tekerő csomagoló /eltolt fázisú kétszeres ütemezés/**



1db → 160s
2db → 180s
3db → 200s
4db → 220s
10db → 340s

20s-enként készül egy újabb

10db hagyományosan 1600s
10db szervezéssel 340s

- **Sebesség növekedés:**

1600:320 = 4,7
100 palacsintára:
(160x100) : (160+99x20) = 7,4

- **Nőtt a kibocsájtás**
- **2 szalagsor működik párhuzamosan**
  2 utas szuper palacsintakészítő
- **Ütemezés 20 s-onként /kétszeres/**
  ☞ **super pipe-line**

de csak egy sütőkezelő (erőforrás)

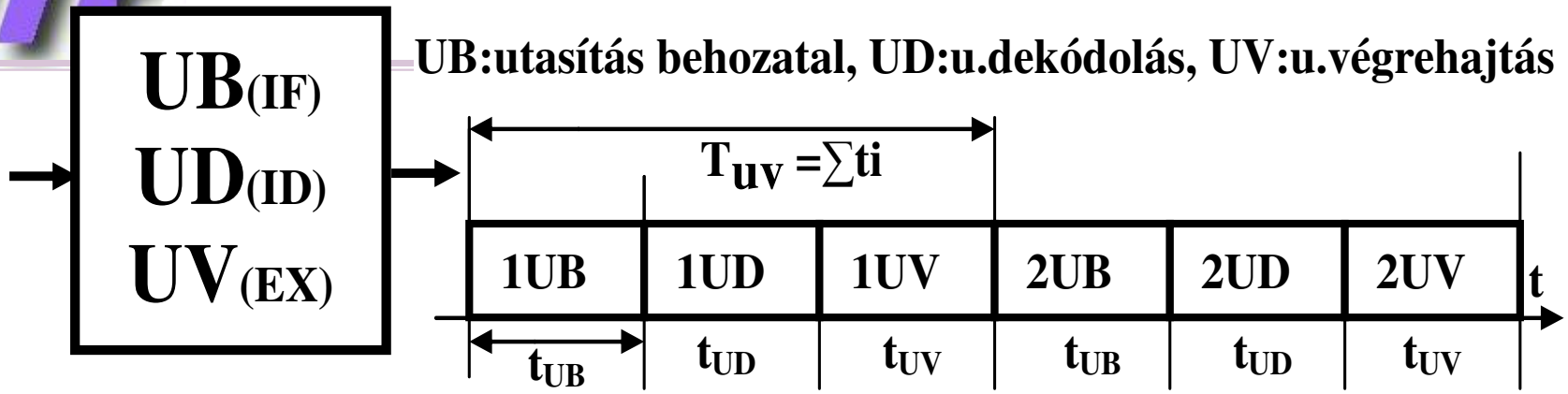
Elégedetten levonhatjuk a palacsinta gyártás tanulságait

- **Megfelelő szervezéssel** /amíg sül, addig tölthetek és tekerhetek/ **nő a kibocsájtott palacsinták száma, miközben egy ugyanannyi idő alatt készül el**
- **Többen csináltuk, párhuzamosan**
- **A sütést két lépésre bontottuk /párhuzamosan két sütőt használtunk/**
- **Egy palacsinta elkészítési ideje nem csökken, sőt!? De rövidebb időnként jön le a szalagról egy kész palacsinta /nő a kibocsájtás/**

- **Több darabot kevesebb idő alatt készítettünk /rövidebb idejű program/**
- **Megfelelő ütemezéssel jól kihasználhatóak az erőforrások**

– **Próbáljuk meg processzorra!**

Hagyományos utasítás végrehajtás

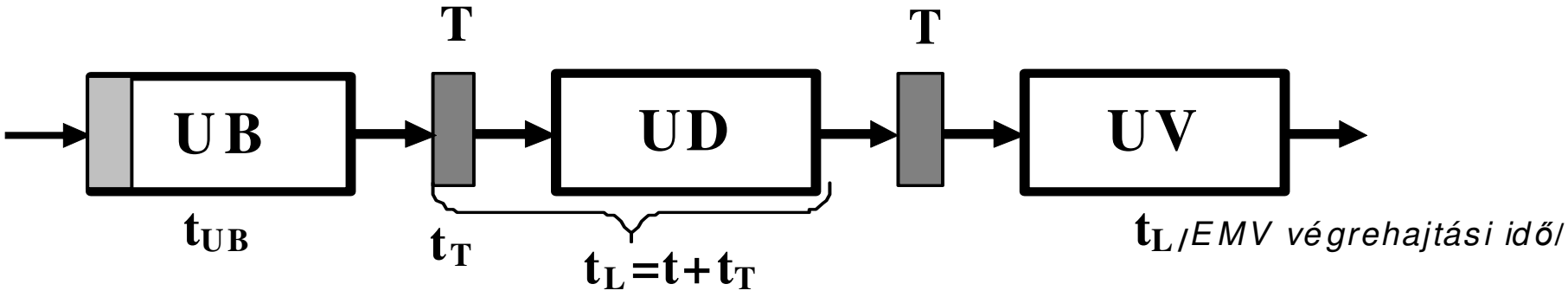


Legegyszerűbb eset: tUB= tUD=tUV=t részműveletekhez egyenlő idő szükséges
Tuv = 3·t 1ut = 3·t, n·ut = n·Tuv =n·3·t

PIPE-LINE /IBM801 1975-80/

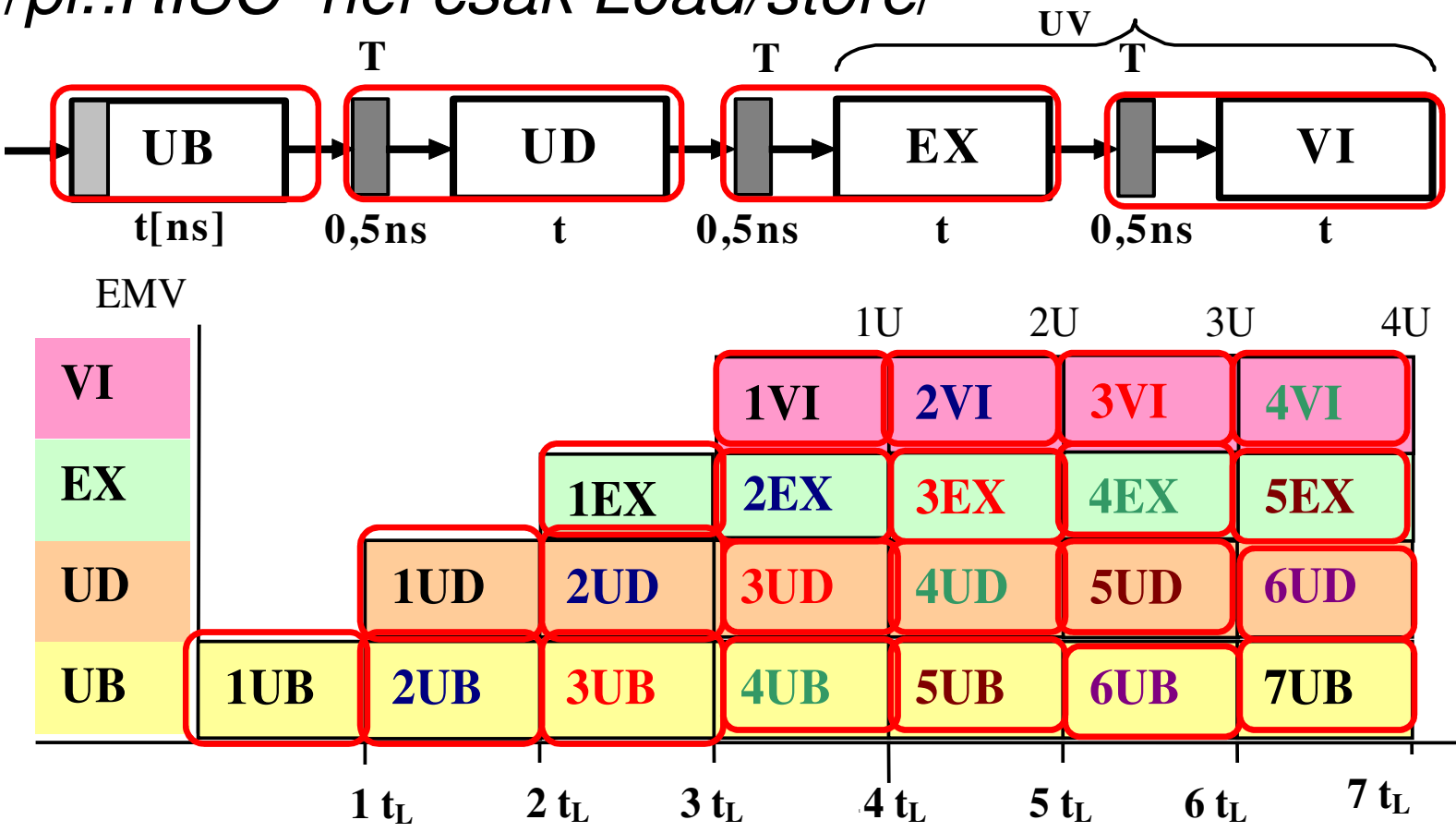
- **Részfeladatokat megoldó, párhuzamosan működő elemi műveletvégzők, tárolóval**

/UB funkcionálisan tartalmaz tárolót, egyszerűség kedvéért az elejére rajzoltuk/



Utasítás feldolgozó szalag /PIPE-LINE/

- Finomítás :pl.: négy elemi műveletvégző
  /pl.:RISC–nél csak Load/store/



- Működés jellemző adatai, hatékonyság jellemzése?
  - ❑ Utasítás lappangási idő (*Instruction latency*)

$$T_L = \sum(t_{UX} + t_T) = \sum t_L$$

itt $T_L = EMVsz \cdot t_L$

$$T_L = 4 \cdot t_L$$

Itt megegyezik egy utasítás végrehajtás idejével

Pl.: t_L =10ns → T_L= Tuv=40ns

  - ❑ Időegység alatt befejezett utasítások száma

TP=1ut/ t_{IKi} ??! */t_{ui} a szalag tovább indítása (újraindítási idő)/*

Optimális esetben: $t_L = t_{ui} = t_{IKi}$ *(feltöltődés után)*

TP=1ut./t_L

Pl.: t_L=10ns

$t_L \neq T_L$!!!

TP=1ut/10ns

Utasítás feldolgozó szalag /PIPE-LINE/

- Sebesség növekedés (S_U) a legjobb mérőszám?

☞ **pl.:N utasításra** ($t_T << t_{UX}$, $N >> 1$ **esetén)**

$$Su_{PL} = \frac{T_{exec}}{T_{exePL}} = \frac{N \cdot 4 \cdot t}{N \cdot t_L + (EMVsz - 1) \cdot t_L} \approx \frac{4 \cdot t}{t_L} \leq 4$$

$$H_{Su} = \frac{Su}{EMVsz} \leq 1$$

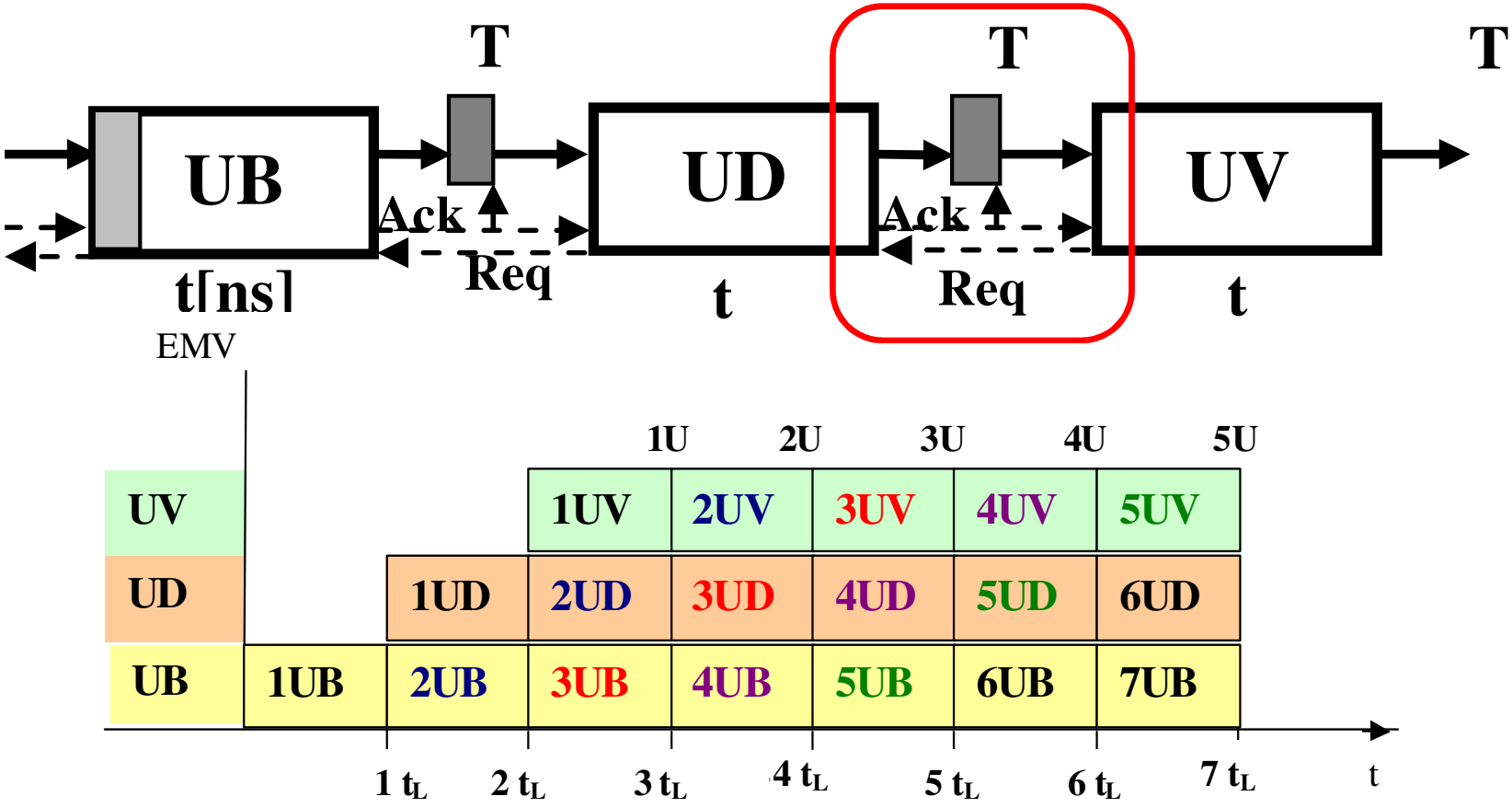
- Optimális esetben: t_L *– ek egyenlők,* t_T *elhanyagolható*
  - $Su_{PL} \approx EMVsz$ *Pl.:* $t_T = 0{,}1ns$ $t_L = 10ns + 0{,}1ns = 10{,}1ns$
  - 10^3 ***utasításra:***

$$Su_{PL} = \frac{T_{exec}}{T_{exePL}} = \frac{10^3 \cdot 4 \cdot t}{10^3 \cdot t_L + (4-1) t_L} = 3.94 \cong 4$$

$$H_{Su} = 0.98$$

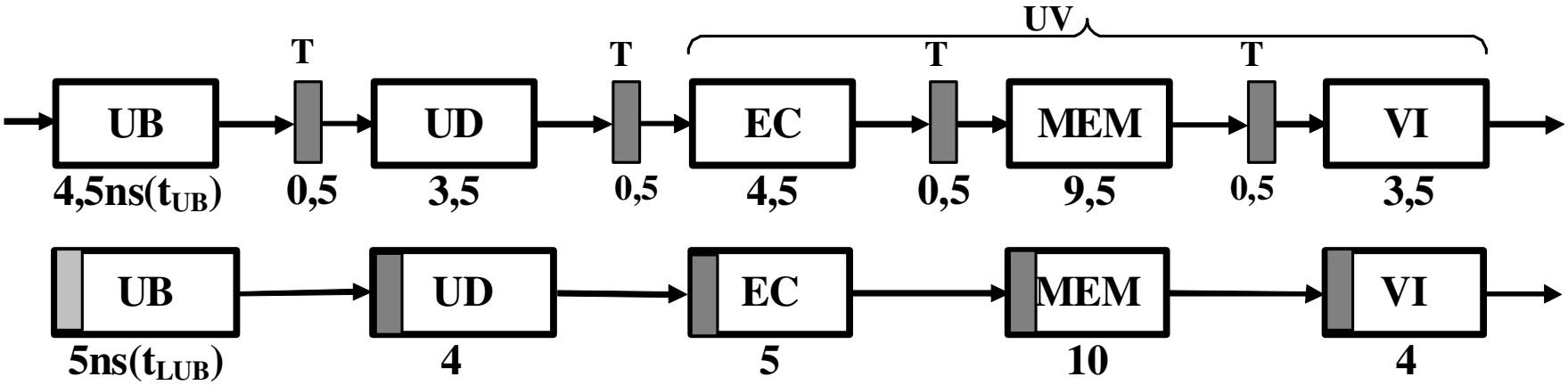
Megoldandó feladatok

- Ütemezés
  - Aszinkron / *ASAP* ***(as soon as posible)*** ütemezés Req/Ack jelekkel



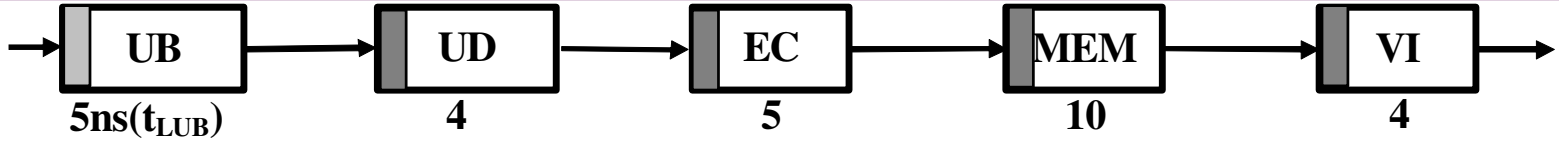
❑Célszerű *további* finomítás a párhuzamosítás miatt

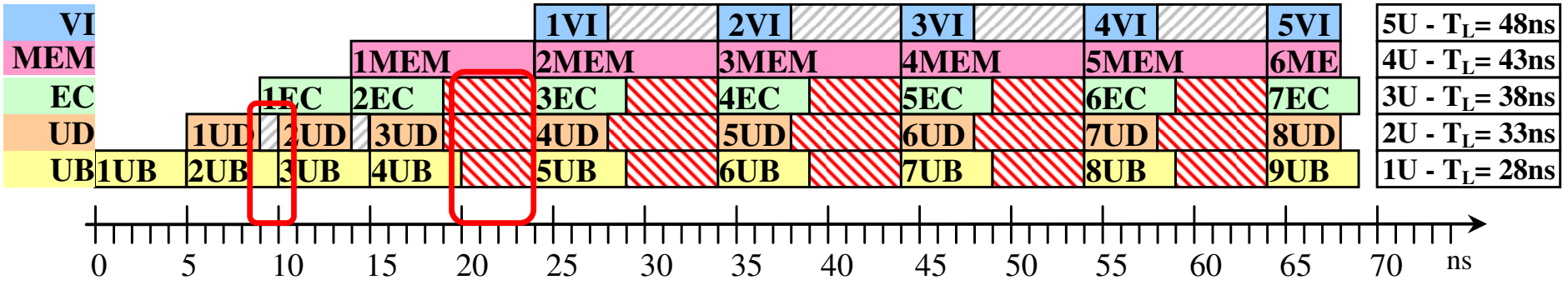
- UV felbontása /*pl.*:***MIPS-nél*** /
  - EC: ***aritmetika, effektív címszámítás***
  - MEM: ***adat memória olvasás/írás***
  - VI: ***eredmény visszaírása***



Utasítás feldolgozó szalag - ASAP ütemezéssel

- ASAP ütemezéssel





Jelölés: ***várakozás az előző műveletvégzőre***

várakozás a következő műveletvégzőre

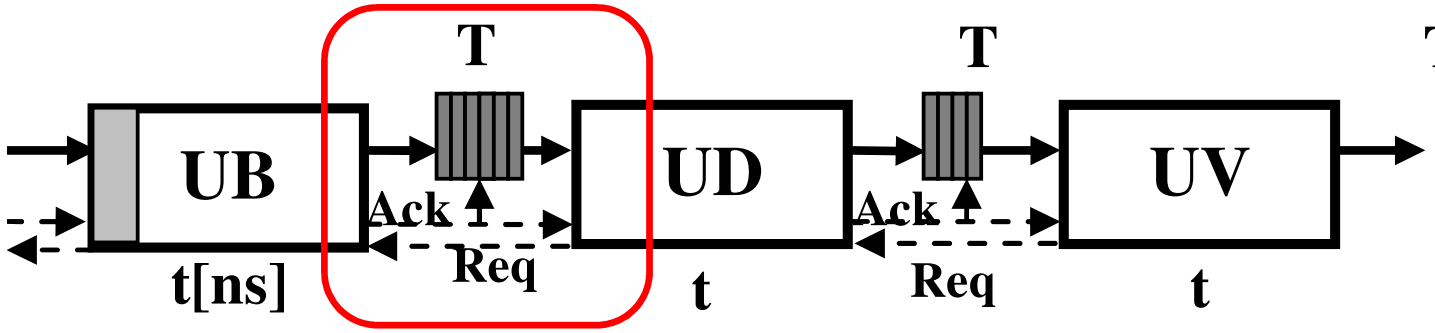
$T_L=Tu=\sum(t_{EMVi}+t_{Ti}+t_{var})$ T_L=? 1.ut T_L =28ns, 2.ut T_L =33ns

➢**Probléma:** ***ugyanazon utasítás végrehajtási ideje eltérő!*** ?

- **A műveletvégzőknek eltérő időkre van szükségük**
  - ***Várakozás hatása?*** 👎 ✓

Megoldás?! → 👍 ✓

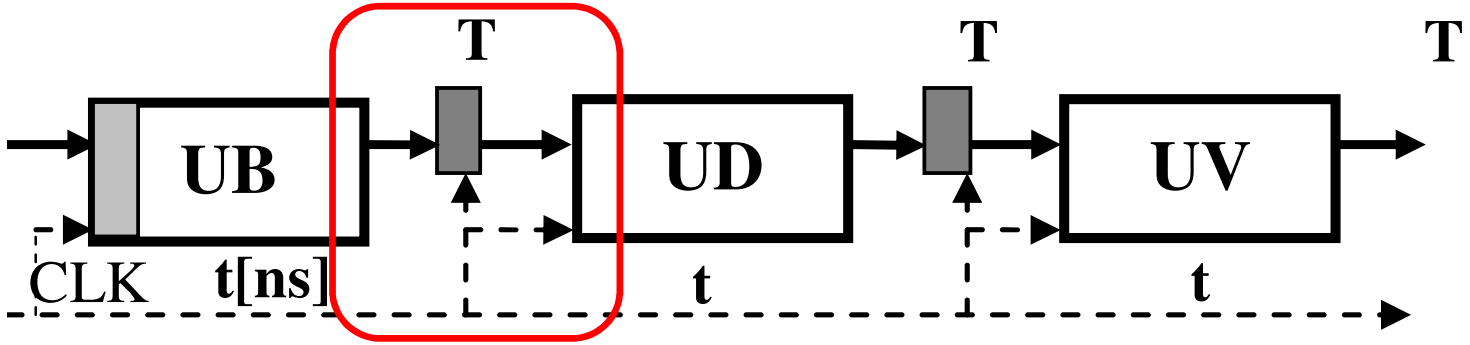
- Puffertároló alkalmazása /szervezés, méret?/
  - » Eltérő idejű, ciklusszámú utasítások esetén
    /csak átmeneti túlterhelés kiegyenlítésére alkalmas/ 👍 👎 ✓



♦ Szinkron ütemezés

- **CLK időzítés /*léptetés*/**

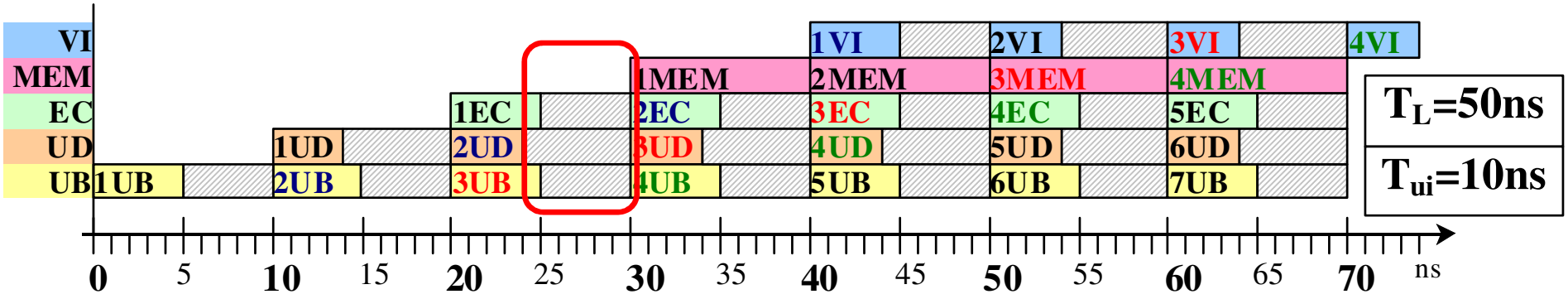
  a leglassúbb műveletvégző idejével



❑Szinkron ütemezés

/UB, UD, EC, MEM, VI műveletvégzővel/

– Újraindítási idő *Tui = $t_L$max = tCLK =10ns*



T_L= Tui =állandó ***TP=1ut./ t_{CLK} = 1ut/10ns***

Pl.:N=10^5 utasítás esetén

$$Su_{PL} = \frac{T_{exec}}{T_{exePL}} = \frac{N \cdot 25{,}5}{(EMVsz-1)\cdot T_{ui} + N \cdot T_{ui}} = \frac{10^5 \cdot 25{,}5}{(5-1)\cdot 10 + 10^5 \cdot 10} \cong 2.55$$

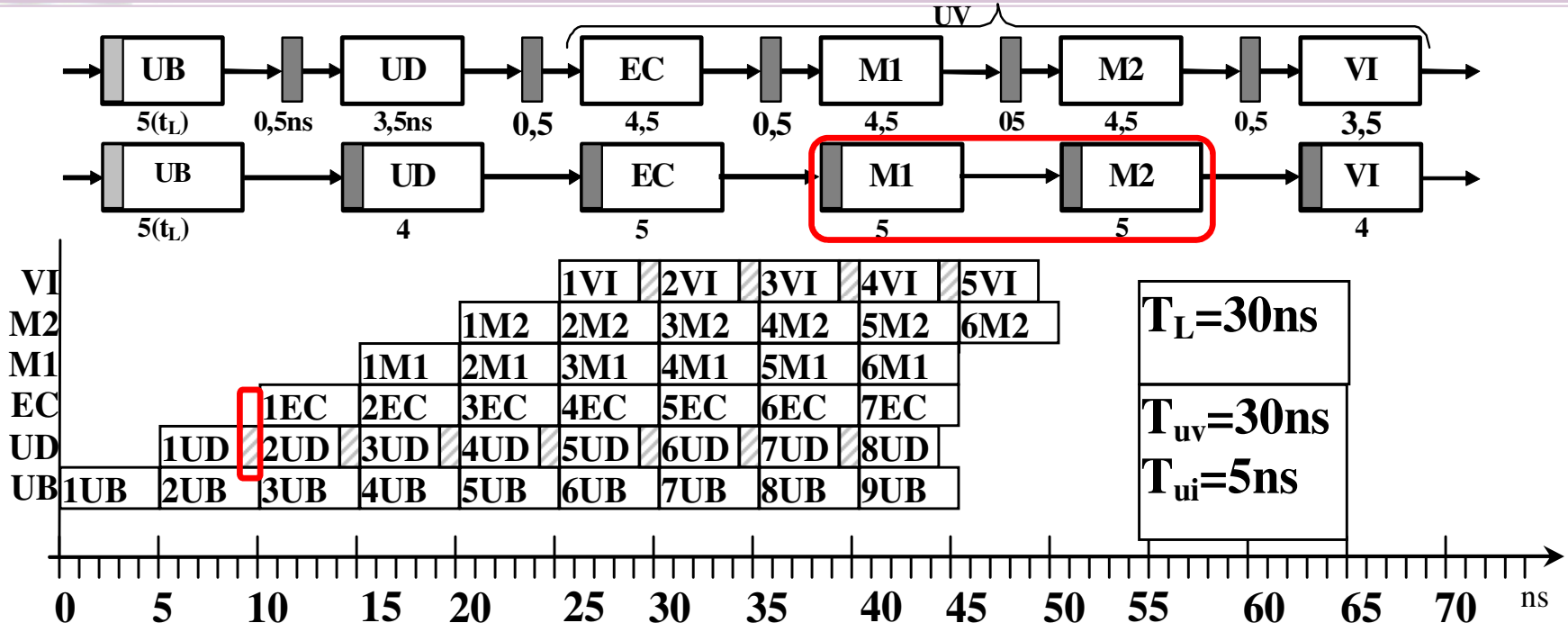
❑ ***Su_{PL}≈0,5·EMVsz*** ⇳ ***H_{SU} =0,5***

A sok üres idő miatt!

➢ ***MEM ideje nagyobb →kiegyensúlyozatlanság (unbalanced)!***

Utasítás feldolgozó szalag - Szinkron ütemezéssel

Megoldás:a memória kezelésre két egység: M1, M2



- **TP=1ut./5ns**

  /10^5 utasítás esetén/

  SuPL≤EMVsz=6 */Azonos t_L és elhanyagolható töltési idő esetén/*

$$Su_{PL} = \frac{T_{exec}}{T_{exePL}} = \frac{10^5 \cdot 25{,}5}{(6-1)\cdot 5 + 10^5 \cdot 5} = \frac{10^5 \cdot 25{,}5}{5 \cdot 5 + 10^5 \cdot 5} \cong 5{,}1$$

H_{SU} =0,85

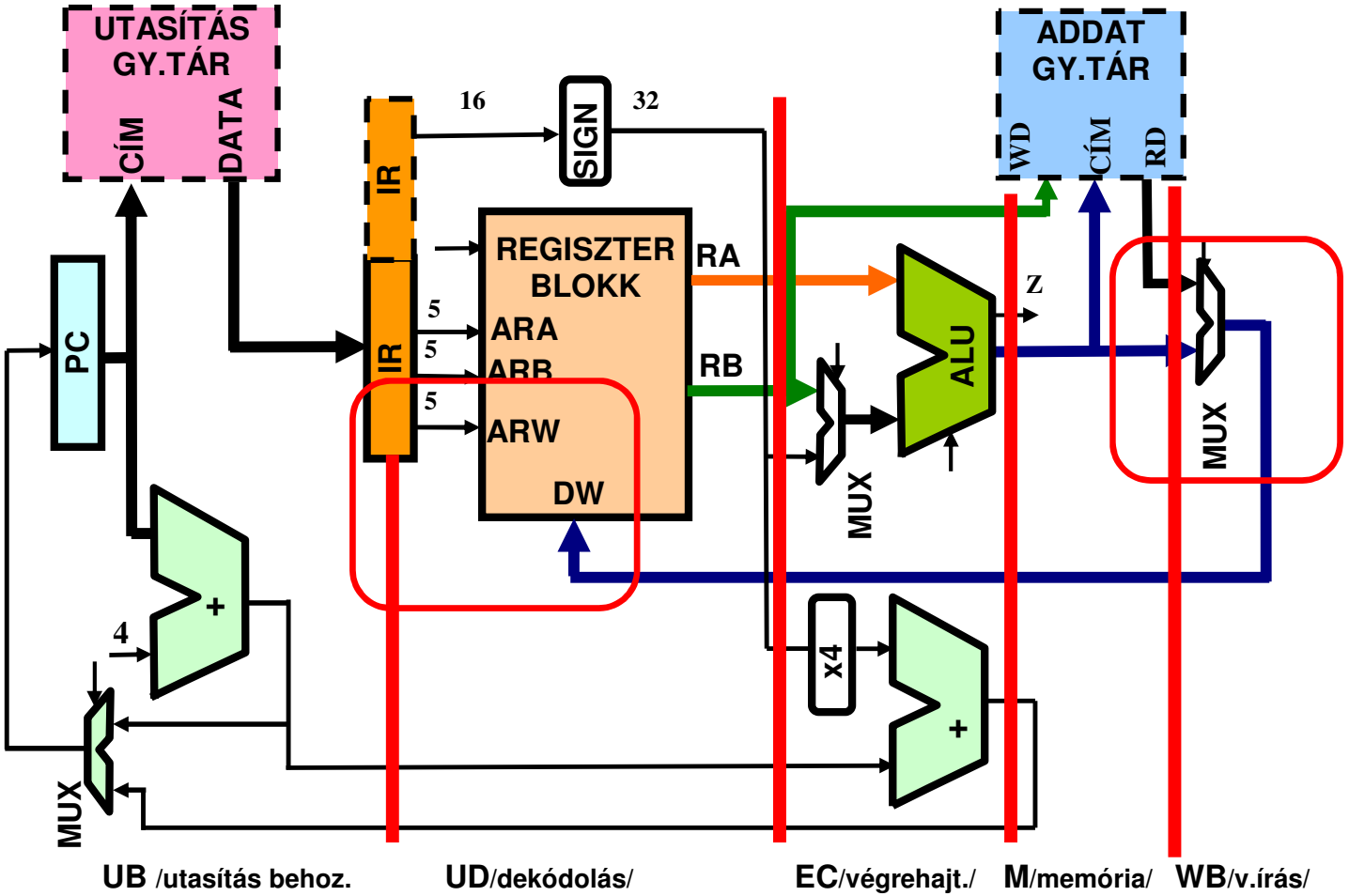
- ***Több műveletvégző, nagyobb az áttöltési idő is, ez csökkenti a hatásfokot***
- ***Pentium 4 20 elemi műveletvégzőt tartalmaz***

Esettanulmány

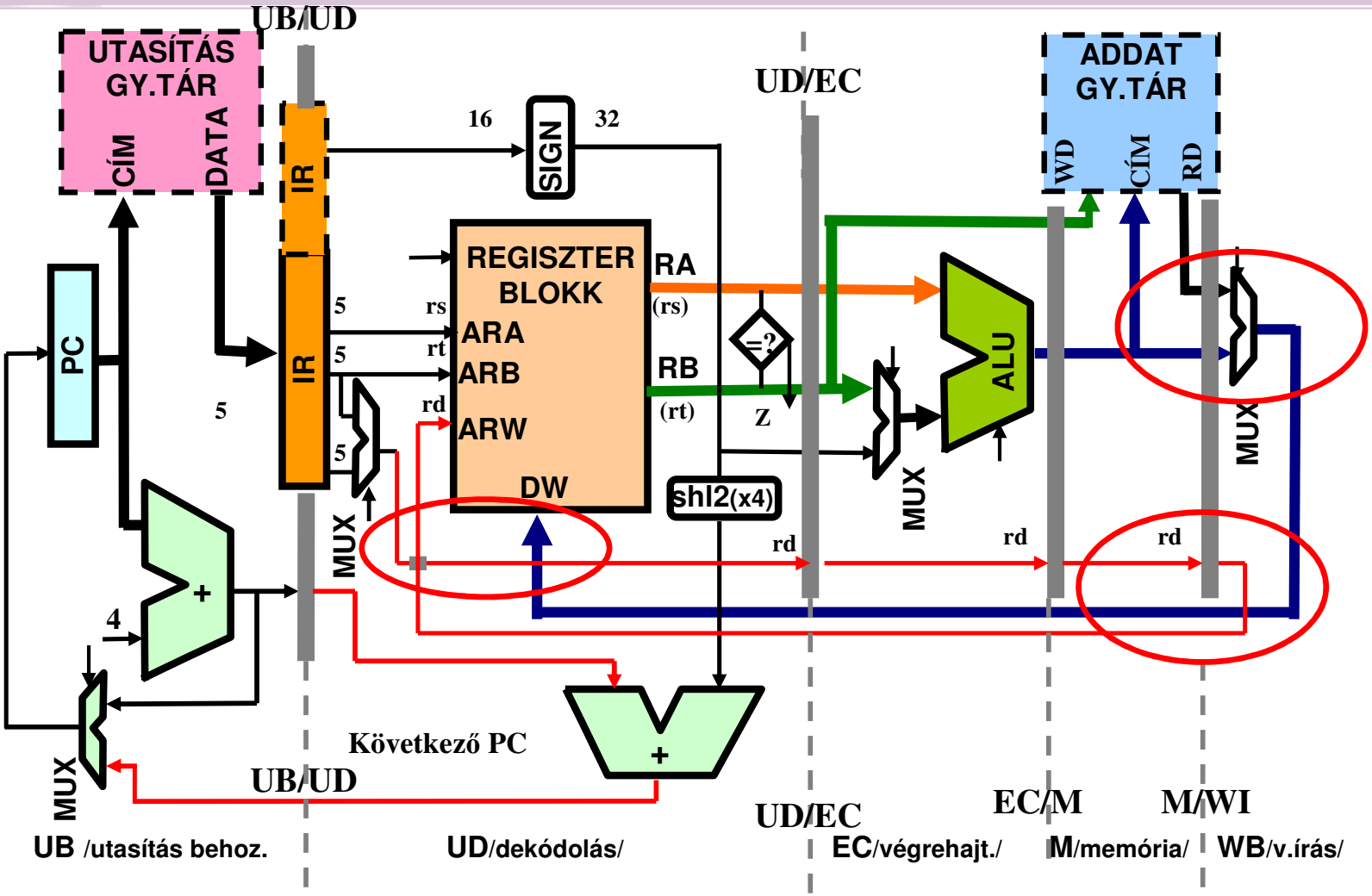
A hw megoldás kérdései néhány gondolat erejéig

Keressenek további problémákat és megoldásokat

MIPS ADATÚT /variáció1 pipe - line. Miért nem működne?!/



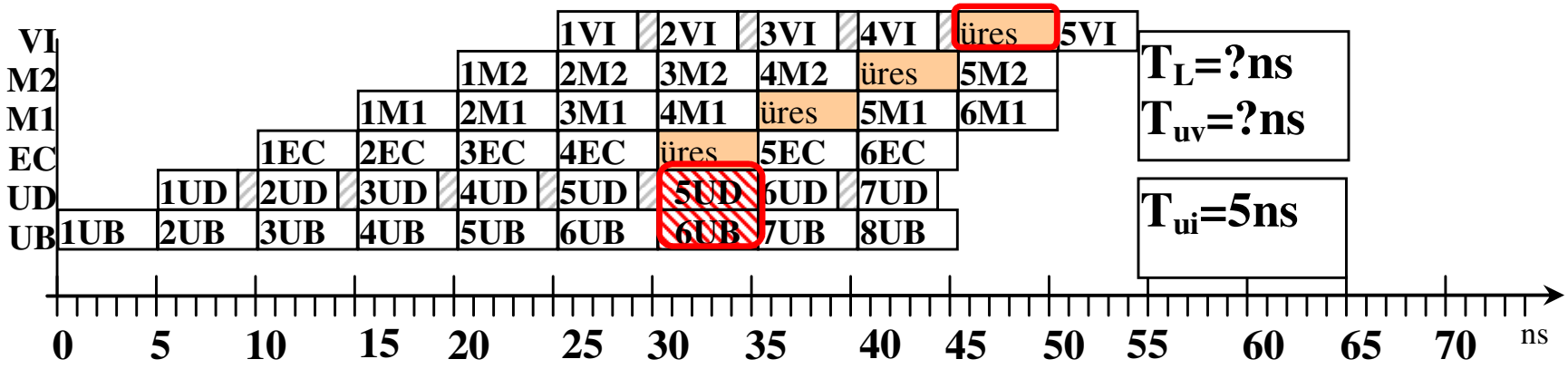
MIPS ADATÚT /Pipe-line gondolat ébresztő variáció 1/Tervezzünk komplet gépet?



– **Milyen utasítás nem működik a fenti megoldásnál?**

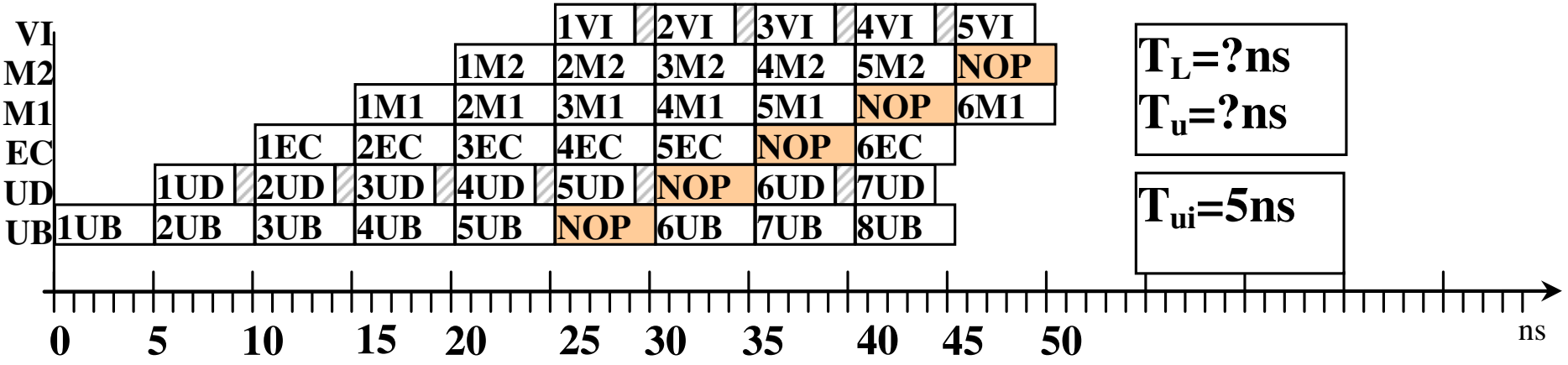
Feldolgozó szalag hatásfok csökkenés

- Problémafelvetés
  - **Nem vettük figyelembe az esetleges egyidejű használatot**
    pl.: EC-VI, vagy UD-VI, néha UB-MEM, stb.
    pl.: 2.utasítás VI-EC (címzési módtól függ)
    Megoldás
  - **Puffer tárolók** 👍
  - **Üres állapot (buborék,**stall**) közbeiktatása** 👍 → H_{SU} ↓
  - **Kizárás vezérlés**
    Adott műveletvégzőt és az előtte lévőket nem lépteti tovább

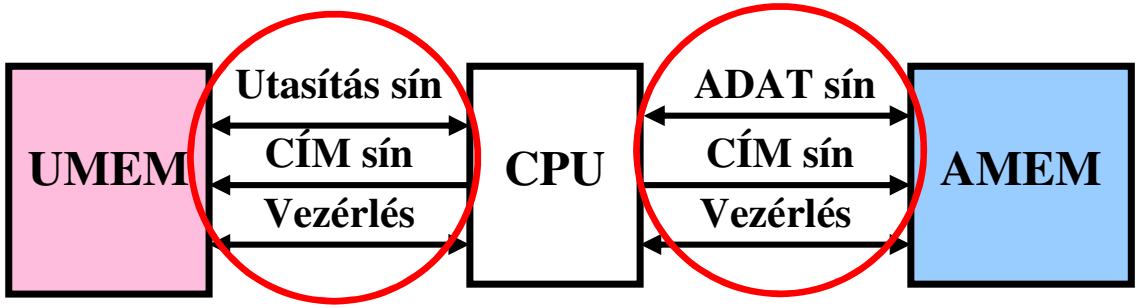


Feldolgozó szalag hatásfok csökkenés

- pl.:3.utasítás az op. memóriába ír és 6.utasítást onnan kell beolvasni
  /pl.:nincs a gyorsítótárban/



- **UB-MEM ütközés oldása: Queue, (puffer tároló)**
  - **Harvard architektúra *szétválasztott adatút/***



  - ***Többszintű, szeparált cache***

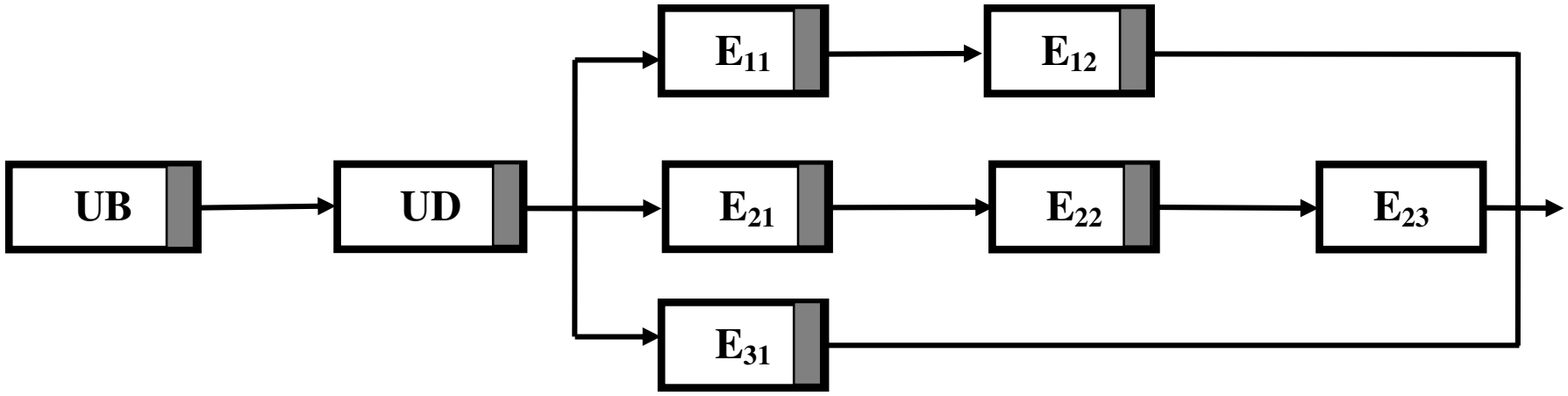
Megoldandó feladatok, problémák

- Végrehajtás lépései utasítás csoportonként különböznek
  - ❑ *Több műveletvégző sor*

  Utasításcsoportonként más feldolgozó ág

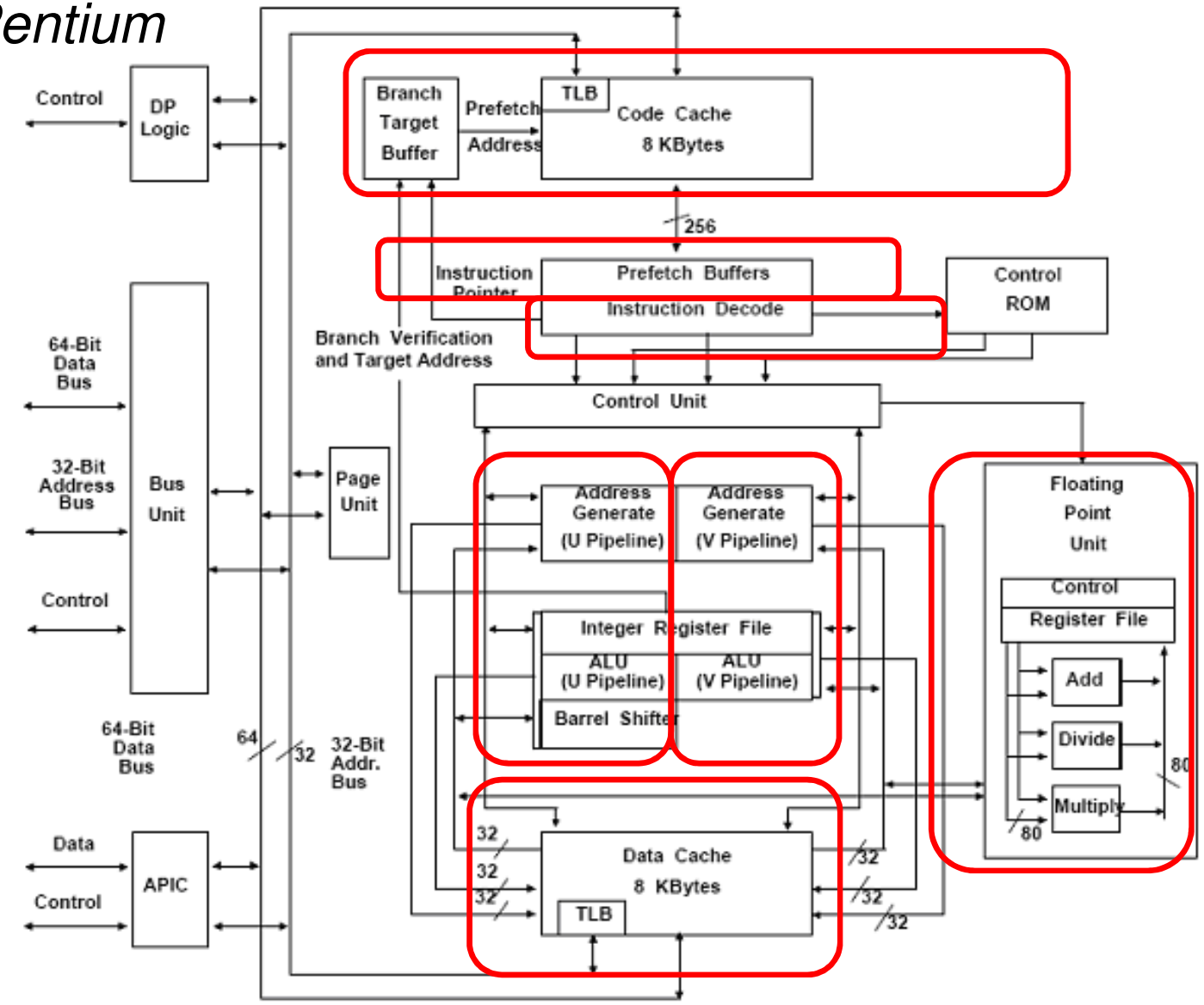
  Egyes egységek átléphetők

  n utas feldolgozó szalag



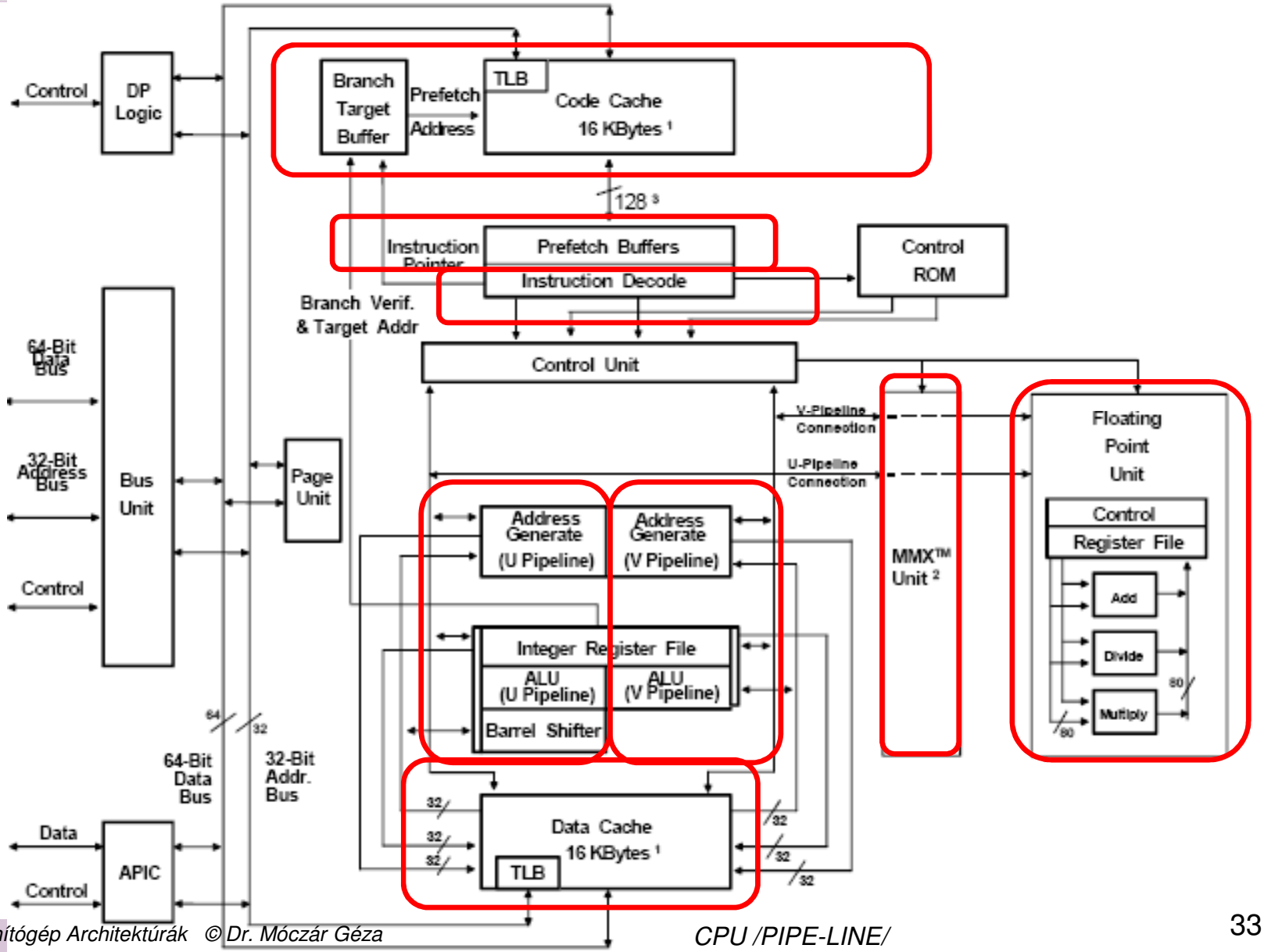
Több műveletvégző sor

✧ Esettanulmány Pentium

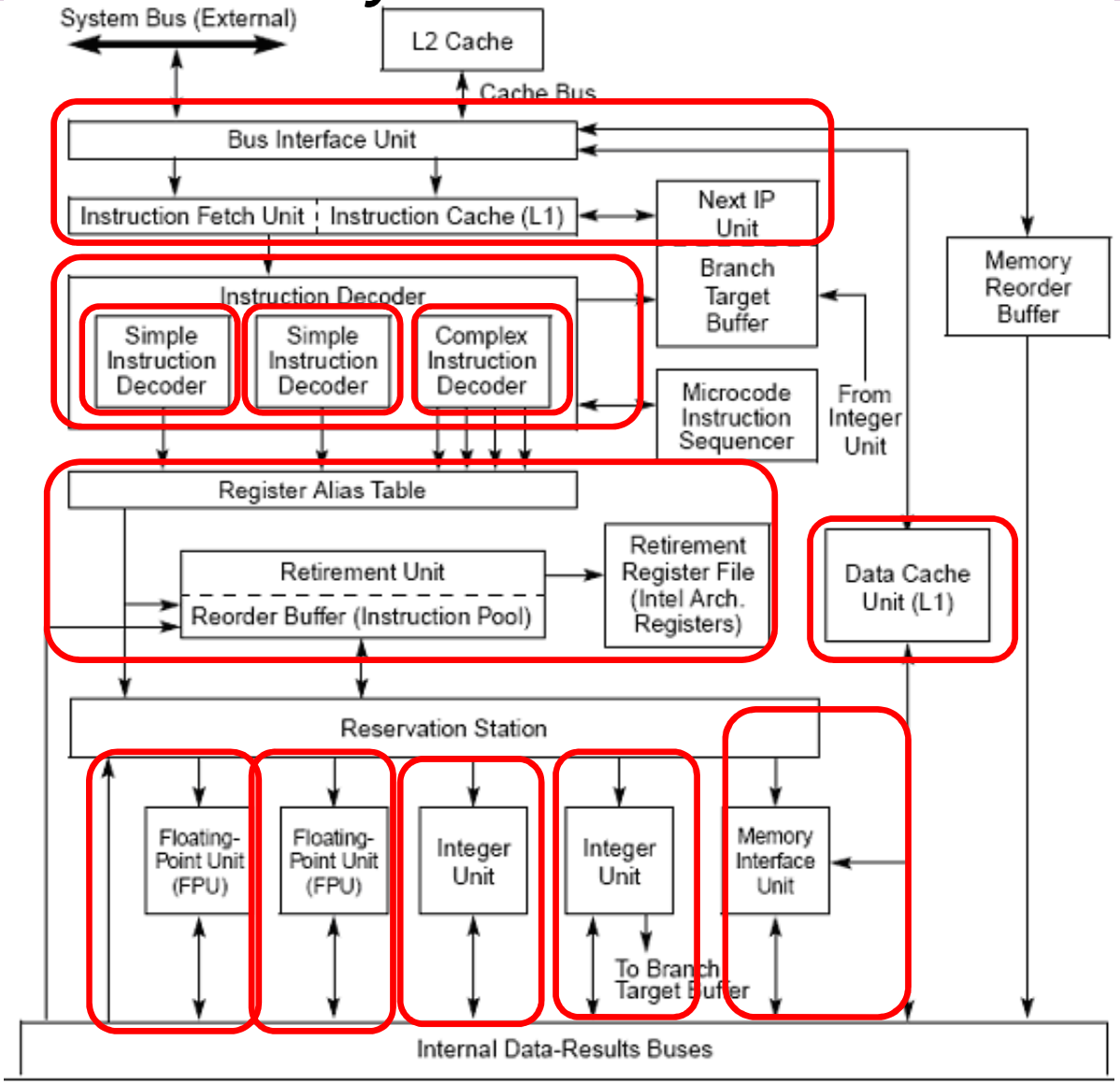


Több műveletvégző sor

Esettanulmány Pentium MMX



Esettanulmány Pentium Pro



Pentium 4 /***Intel technology journál Q1 2001/ /***

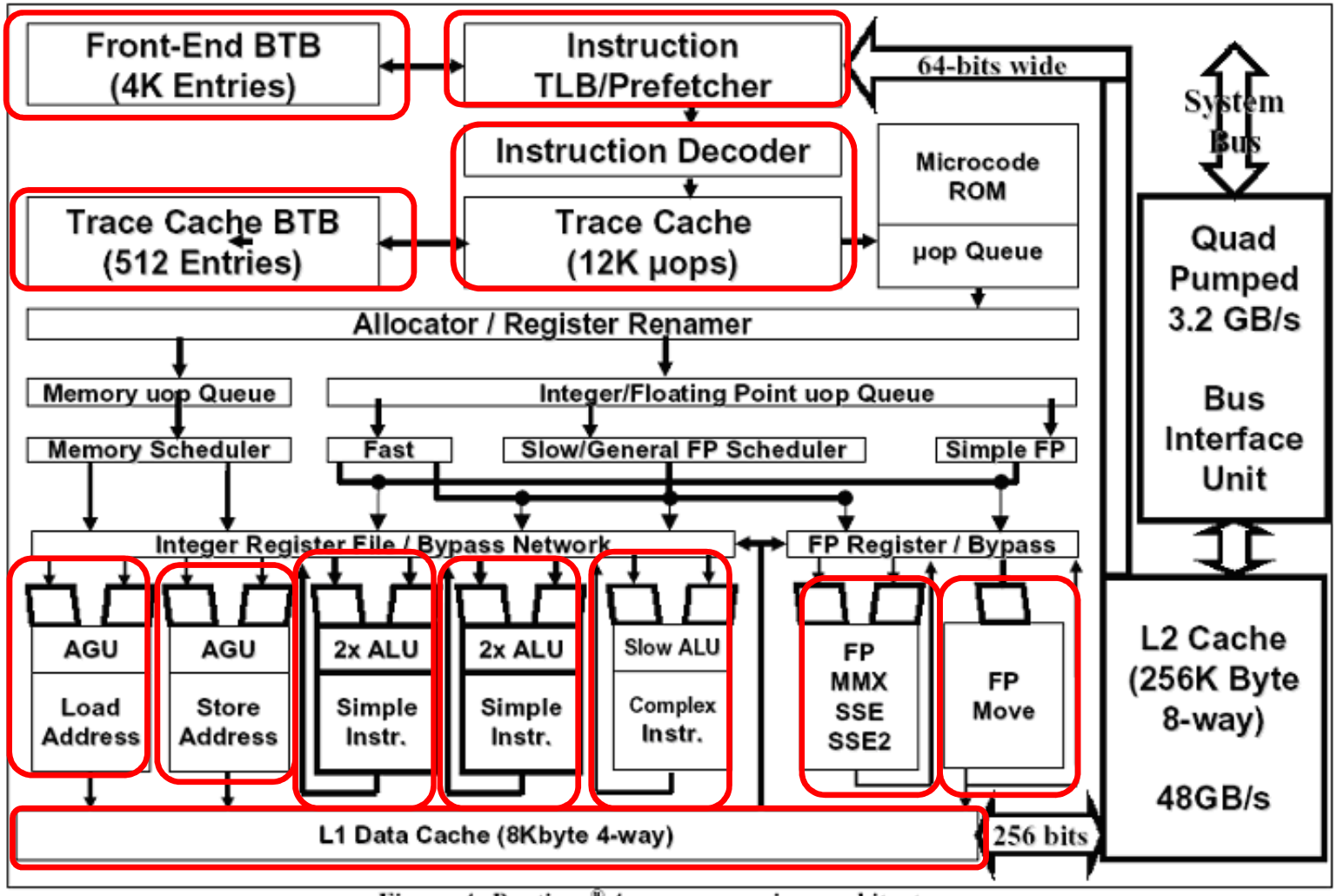


Figure 4: Pentium® 4 processor microarchitecture

Pentium III

Pentium 4 hiper feldolgozó szalag

Pentium 4 eltérő frekvenciát használ az egyes feldolgozó szalagokon→pl.: 2xALU

Hibás becslés hatása

Basic Pentium III Processor Misprediction Pipeline

1	2	3	4	5	6	7	8	9	10
Fetch	Fetch	Decode	Decode	Decode	Rename	ROB Rd	Rdy/Sch	Dispatch	Exec

Basic Pentium 4 Processor Misprediction Pipeline

1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20
TC Nxt IP		TC Fetch		Drive	Alloc	Rename		Que	Sch	Sch	Sch	Disp	Disp	RF	RF	Ex	Flgs	Br Ck	Drive

- Utasítás egymásra hatás
  - ❑FELDOLGOZÁSI */**Structural**/*
    Egy időben ugyanarra a műveletvégzőre van szükség
    – Megoldás lehet
    Kétutas adatút memória eléréshez utasítás/adat
    Többutas feldolgozószalag struktúra
  - ❑ADAT */Data/*
    Az utasításnak szüksége van az előző utasítás/ok/ még nem meglévő, eredményére
    – Műveletek más sorrendben történő végrehajtása
      » *Fordítóprogram figyelheti*
      *Pl.:load időszelet alatt más utasítást hajt v*égre */MIPS/*

Utasítás egymásra hatás

- » *Fordító jelzi a regiszterhasználatból eredő hatást. Nem kell hardver vezérlés a kizárásra*
- » NOP/üres lépés beiktatása

– Előre csatolás

Az eredmény a szalag előző egységeiről elérhető!? 👍 👎

❑ *PROCEDURÁLIS /**Control**/*

☞ *Mivel folytassa?* ☞ ***már megkezdte más utasítások feldolgozását!?***

✧ Vezérlésátadásnál

☞ Hatásfok csökkenés ***/feleslegesen betöltött rész ürítése/*** 👎

☞ *Ezt is kezelheti a fordító? Időszelet ?!* 👍

■ Feltételes vezérlésátadás

☞ *Nem ismert, melyik ágat kell előre betölteni* ☞

☞ Esettanulmány ***a procedurális utasítás egymásra hatás kezelésére***

Esettanulmány i386, i486, Pentium, Itanium

- *Procedúrális utasítás egymásra hatás kezelése*
  - *i386-leállítja a szalag további töltését a feltétel kiértékelődéséig* 👍 👎
  - *i486-előre rögzített módon → pl.: nem ág*
    Megjósolja az utat, megjelölve tölt
    A nem visszaállítható művelet végrehajtásával vár a kiértékelődésig
  - *Pentium-dinamikusan számítja az előzmények alapján a valószínű ágat*
    tovább mint az i486-nál 👍 👍 👍
  - *Itanium → Spekulatív végrehajtás*
    Spekulatív végrehajtás→Ha előnnyel jár
    a.)Sokszor fordul elő b.)Kicsi a visszaállítás „költsége”
    - Pl.:vezérlés függés:

```
if(a>b)     load(ld_adrr1,target1)
else        load(ld_adrr1,target2)
```

Esettanulmány i386, i486, Pentium, Itanium - Procedúrális utasítás egymásra hatás kezelése

Korábban végrehajtva ***/Itt nem időkritikus/*** *sload(ld_addr1,target1)* ***NaTs bit állítása kivétel kezeléshez***

sload(ld_addr2,target2) ***NaTs bit állítása***

Hibás ág visszaállítás

Az eredeti helyen ***/ha a betöltés kivételkezelést eredményezett volna/***

if (a>b) scheck(target1,recovery_addr1) ***NaTs bit alapján***

else scheck(target2, recovery_addr2)

- ***Pl.:adatfüggés:***

  store(st_addr,data) ***nincs adat függés, ha a két cím különböző***

  load(ld_addr,target) ***adat függés, ha a két címben átfedés lehet***

  use(target)….

Korábban végrehajtva ***/Itt nem időkritikus/***

aload(ld_addr,target) ***/spekulatív load, advenced load/***

Az eredeti helyen

store(st_addr,data) ***adat függés, ha ld és st címekben átfedés van futáskor***

acheck(target,recovery_addr) ***NaTs, adat függés esetén visszaállítás***

Predikáció p0-p63 1 bites predikátum regiszter

A feltételes vezérlésátadás helyett, feltételes utasítások →Feltételes utasítás végrehajtás

Feltétel vizsgáló utasítás

Beállít egy adott pr regisztert /pT/ 1-re ha teljesül

Egy másikat /pF/ 0-*ra a nem ágon* ***/1-re ha nem teljesül/***

*Pl.: „**C**”*

```
if (r1)
    r2 = r3 + r4;
else
    r7 = r6 - r5;
```

```
cmp.ne p1,p2 = r1,0; ;   pT,pF beállítása,leállás
(p1) add r2 = r3,r4 ;    feltételes végrehajtás
(p2) sub r7 = r6,r5 ;    feltételes végrehajtás
```

Mindkét ág utasításait párhuzamosan hajthatja végre

A feltételnek ellentmondó ág „NOP” */felülírás elnyomása/*

A szalag végén az eredményt felhasználja, vagy nem

Nincs hatásfok csökkenés az elágazás miatt👍👍👍👍

- *IT érvényre jutás!* ⇦ **mikor milyen állapotot lát?**
- *PC tartalmak, stack tartalmak megőrzése*
- *pl.: PC relatív vezérlés átadás /Branch, IT/*

Összefoglalás

- Pipe-line *nem csökkenti egy utasí*tás végrehajtási idejét
- *Növeli* az utasítás kibocsájtást /TP/ az elemi műveletvégzők párhuzamos működtetésével

$$P = \frac{IPC \cdot f_{U_{Ki}}}{I_N(A)} = \frac{TP}{I_N(A)}$$

- Optimális esetben $TP \approx 1ut/t_L$
- EMV-k működési idejének *közel azonosnak kell* lenniük
- *Szinkron ütemezést* a leglassabb műveletvégző idejével kell végezni
- Hatékonyságát a sebesség növekedéssel és annak arányával mérhetjük $/Su_{PL}/$

$$H_{Su} = \frac{Su}{n} \leq 1$$

- Optimális *esetben /EMV t_L-k egyenlők, kis tároló töltési idő/*

  n EMV esetén, $Su_{PL} \approx EMVsz = n$, $H_{SU} \approx 1$
- Optimálistól *eltérő esetek csökkentik* a hatékonyságot
- *Utasítás egymásra hatás eseteit kezelni kell*