HÁLÓZATI RENDSZEREK ÉS SZOLGÁLTATÁSOK TANSZÉK

MŰEGYETEM 1782

Budapest,
2025.04.29.

SZÁMÍTÓGÉP ARCHITEKTÚRÁK

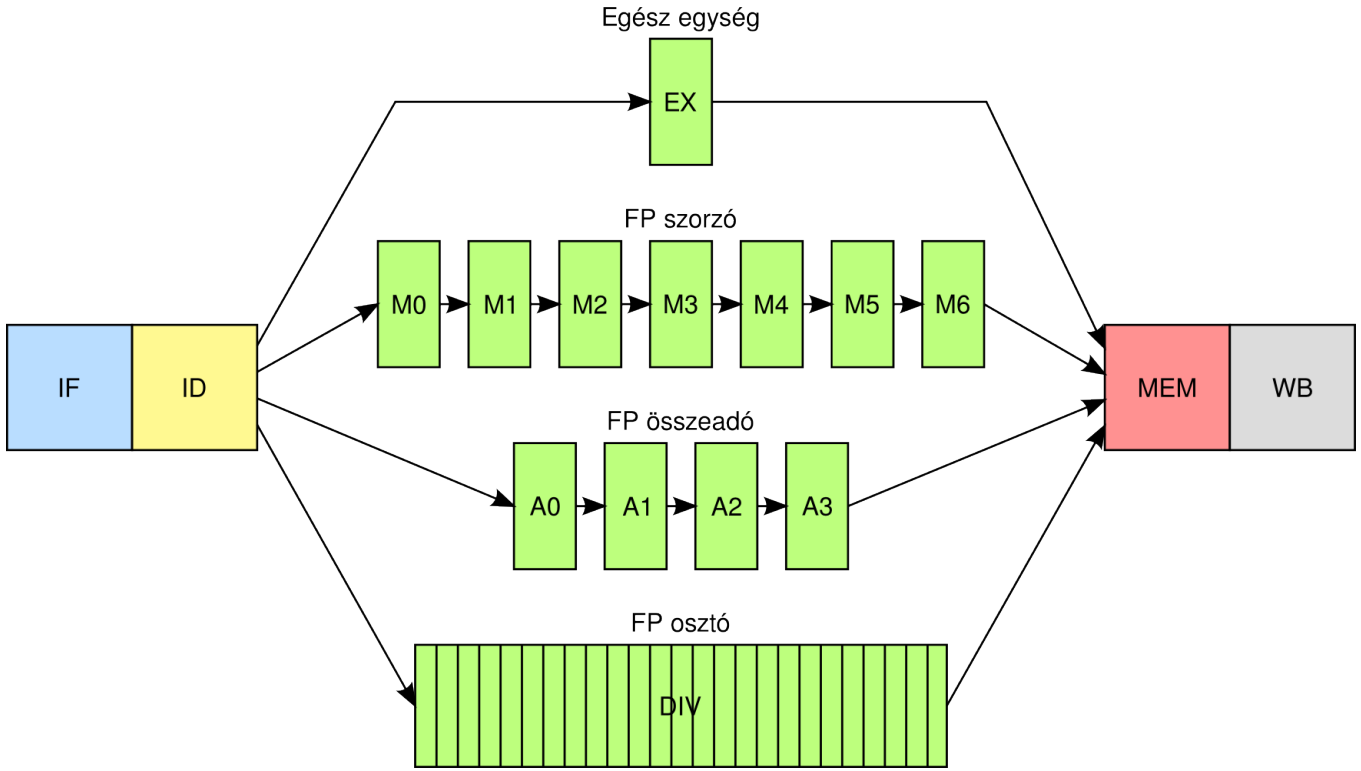
Soron kívüli utasítás-végrehajtás

Horváth Gábor, Belső Zoltán

BME Hálózati Rendszerek és Szolgáltatások Tanszék
ghorvath@hit.bme.hu, belso@hit.bme.hu

KÜLÖNBÖZŐ KÉSLELTETÉSŰ MŰVELETEK

- Láttuk, hogy a lebegőpontos műveletek késleltetése nagyobb



KÜLÖNBÖZŐ KÉSLELTETÉSŰ MŰVELETEK

C kód:

```
for (i=0; i<N; i++)
    Z[i]=A*X[i];
```

Elemi utasítások:

```
D2 ← MEM[R1]
D3 ← D2 * D0
MEM[R2] ← D3
R1 ← R1 + 8
R2 ← R2 + 8
```

A: D0
X[i]: MEM[R1]
Z[i]: MEM[R2]

Utasítások ütemezése: (szorzás: 5 órajel, egész és memóriaművelet: 1 órajel)

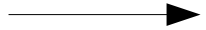
Utasítások:	1.	2.	3.	4.	5.	6.	7.	8.	9.	10.	11.	12.	13.	14.
D2 ← MEM[R1]	IF	ID	EX	MEM	WB									
D3 ← D2 * D0		IF	ID	A*	M0	M1	M2	M3	M4	MEM	WB			
MEM[R2] ← D3			IF	F*	ID	A*	A*	A*	A*	EX	MEM	WB		
R1 ← R1 + 8					IF	F*	F*	F*	F*	ID	EX	MEM	WB	
R2 ← R2 + 8										IF	ID	EX	MEM	WB

OPTIMALIZÁLÁS

- Optimalizáljuk a programot a sorok átrendezésével!

Eredeti utasítássorozat:

```
D2 ← MEM[R1]
D3 ← D2 * D0
MEM[R2] ← D3
R1 ← R1 + 8
R2 ← R2 + 8
```



Optimalizált utasítássorozat:

```
D2 ← MEM[R1]
D3 ← D2 * D0
R1 ← R1 + 8
MEM[R2] ← D3
R2 ← R2 + 8
```

Utasítások:	1.	2.	3.	4.	5.	6.	7.	8.	9.	10.	11.	12.	13.	14.
D2 ← MEM[R1]	IF	ID	EX	MEM	WB									
D3 ← D2 * D0		IF	ID	A*	M0	M1	M2	M3	M4	MEM	WB			
R1 ← R1 + 8			IF	F*	ID	EX	MEM	WB						
MEM[R2] ← D3					IF	ID	A*	A*	A*	EX	MEM	WB		
R2 ← R2 + 8						IF	F*	F*	F*	ID	EX	MEM	WB	

OPTIMALIZÁLÁS

- Utasítások átrendezésével gyorsabb lett a program
- Problémák:
  - Kihasználása a programozón / fordítón múlik
  - Minden pipeline struktúrára máshogy kell
- Kívánatos megoldás:
  - Csinálja a CPU, röptében!
  - Rendezze át az utasításokat kénye-kedve szerint
    - Úgy, hogy gyorsabb legyen
    - Úgy, hogy a szemantika ne változzon

  → **Soron kívüli (out-of-order) utasítás-végrehajtás**
- Megvalósítás:
  - Nem is túl bonyolult
  - Régóta alkalmazott elv
    - Első: Scoreboard – 1964
    - Fejlettebb: **Tomasulo** – 1967
      - Mind a mai napig használják (pl. Intel Core i7)

Soron kívüli utasítás-végrehajtás

A KEZDETEK...

- Első out-of-order processzor:
  - 1964: CDC 6600
- Tervező: Seymour Cray
- Tulajdonságok:
  - 16 műveleti egység
  - 10 MHz órajel
  - >400.000 tranzisztor
  - 5 tonna
  - Huzalozott vezérlés
- 5 éven át a leggyorsabb a világon

CDC-6600

Freon alapú hűtés
minden szekrényben:

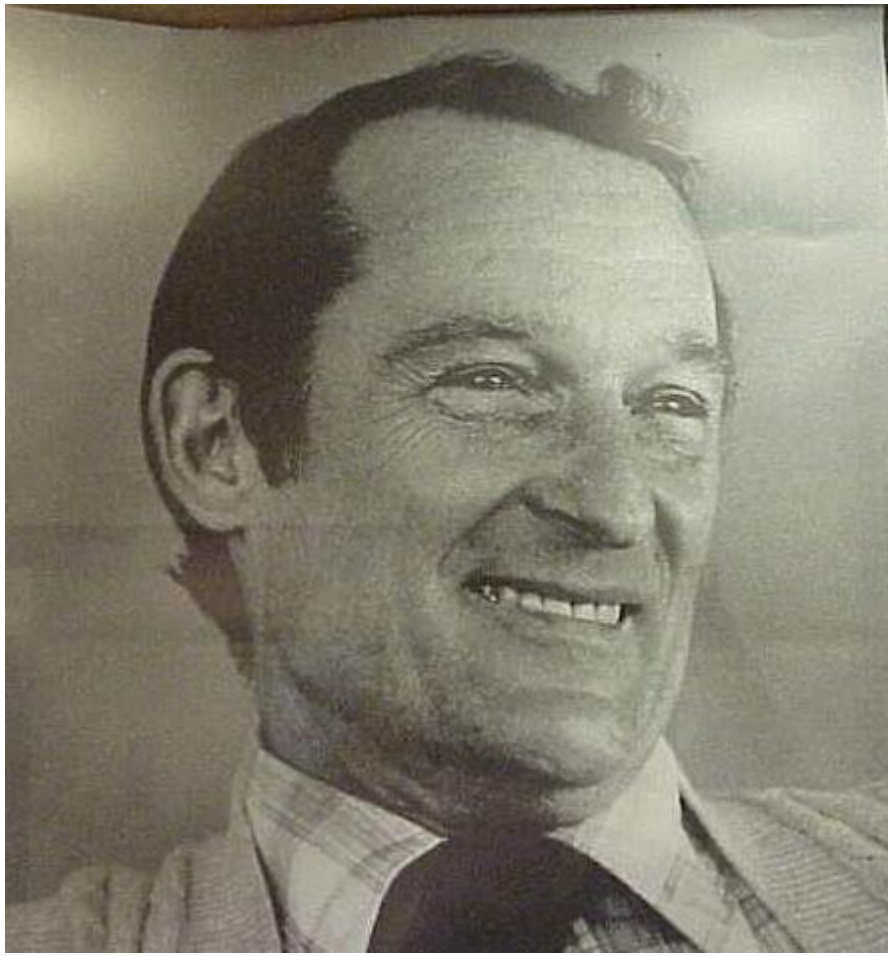
Duál vektorgrafikus megjelenítő:

CRAY VS. IBM

- **Thomas Watson (IBM):**
  „Múlt héten megjelent a CDC 6600. Annál a cégnél összesen 34-en dolgoznak a portással együtt. Közülük csak 14 mérnök és csak 4 programozó. Nem értem, hogy a mi hatalmas fejlesztési ráfordításaink mellett hogy veszthettük el a vezető pozíciónkat, hogy hagyhattuk, hogy más építse meg a világ leggyorsabb számítógépét.”

CRAY VS. IBM

- **Seymour Cray (CD)**
  „Úgy tűnik, Mr. Watson már meg is válaszolta a kérdését”



SORONKÍVÜLI VÉGREHAJTÁS HOZZÁVALÓI

- Több utasítást is le kell hívnunk, melyek közül válogatunk
  - Ezeket tárolni kell valahol:

    → **utasítástároló**
- Egy eljárás, ami eldönti az utasítások végrehajtási sorrendjét

  → **dinamikus ütemező**
- Egy jó ötlet, amivel az utasítások közötti függőségek csökkenthetők

  → **regiszterátnevezés**
- Egy trükk, hogy a külvilág sorrendi végrehajtást lásson

  → **sorrend-visszaállító buffer**

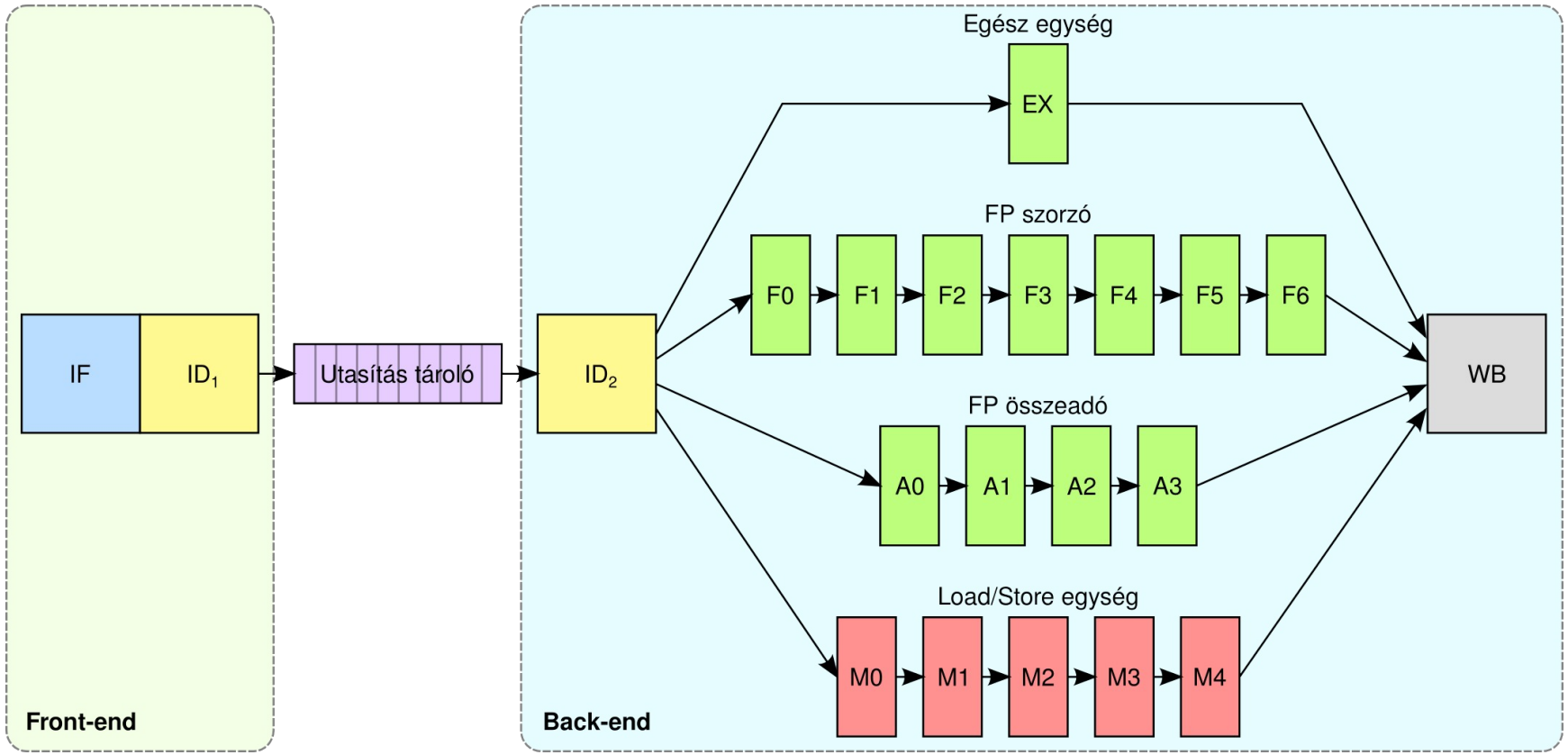
SORONKÍVÜLI VÉGREHAJTÁS HOZZÁVALÓI

- Több utasítást is le kell hívnunk, melyek közül válogatunk
  - Ezeket tárolni kell valahol:
    → **utasítástároló**
- Egy eljárás, ami eldönti az utasítások végrehajtási sorrendjét
  → **dinamikus ütemező**
- Egy jó ötlet, amivel az utasítások közötti függőségek csökkenthetők
  → **regiszterátnevezés**
- Egy trükk, hogy a külvilág sorrendi végrehajtást lásson
  → **sorrend-visszaállító buffer**

UTASÍTÁSTÁROLÓ

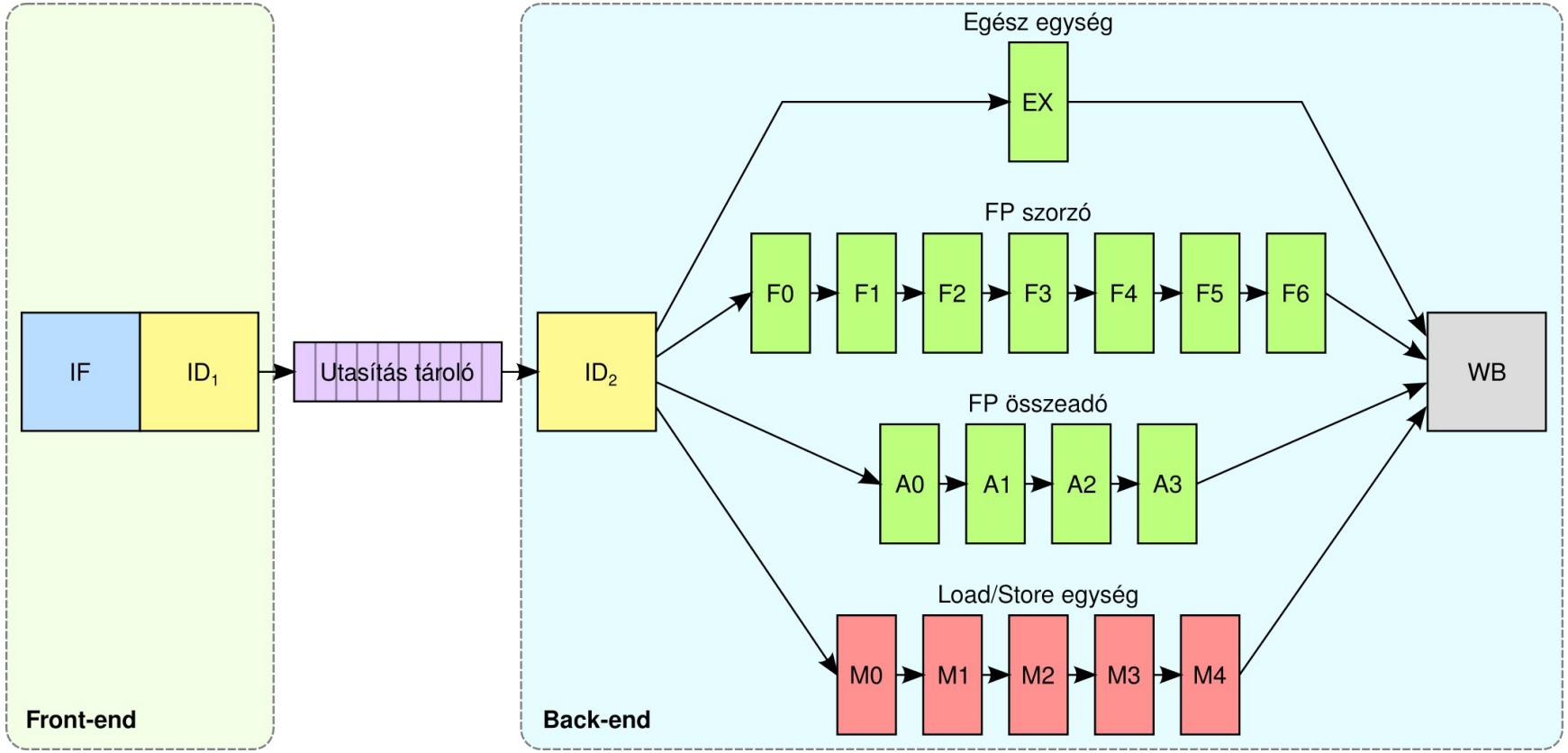
- Utasítások lehívása (IF) → sorrendben
- Utasítások végrehajtása (EX) → soron kívül

  → az utasítástároló csak az ID fázisban lehet!



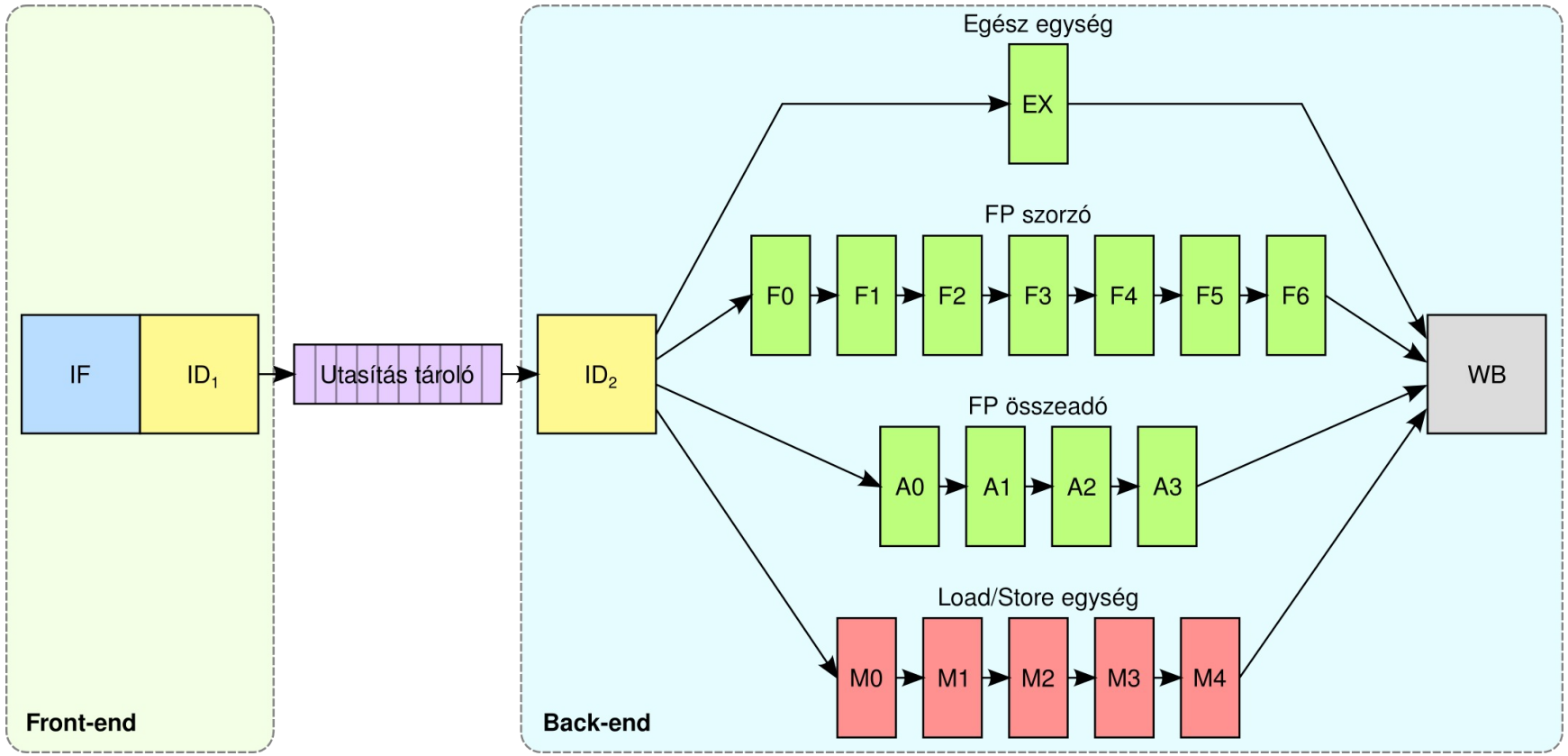
UTASÍTÁSTÁROLÓ

- Az utasítástároló két részre vágja a futószalagot:
  - Front-end: sorrendi belépés
  - Back-end: soron kívül belépés



UTASÍTÁSTÁROLÓ

- ID ketté bomlik (nem egységes elnevezések!):
  - ID1: Dispatch (DS) – dekódolás, tárolóba tétel
  - ID2: Issue (IS) – operandusok összevadászása, műveleti egységhez rendelés



UTASÍTÁSTÁROLÓ

- Alternatív nevek: Reservation Station, Instruction Window, stb.
- Felépítés lehet:
  - *elosztott* (műveleti egységenként külön)
  - *centralizált* (minden műveleti egységre közös)

CPU	Utasítás tároló típusa	mérete
IBM PowerPC4	Elosztott	31
Intel Pentium III	Centralizált	20
Intel Pentium 4	Hibrid (1 mem. Műv, 1 többi)	126 (72/54)
Intel Core	Centralizált	32
AMD K6	Centralizált	72
AMD Opteron	Elosztott	60

SORONKÍVÜLI VÉGREHAJTÁS HOZZÁVALÓI

- Több utasítást is le kell hívnunk, melyek közül válogatunk.
  - Ezeket tárolni kell valahol:
    → **utasítástároló**
- Egy eljárás, ami eldönti az utasítások végrehajtási sorrendjét
  → **dinamikus ütemező**
- Egy jó ötlet, amivel az utasítások közötti függőségek csökkenthetők
  → **regiszterátnevezés**
- Egy trükk, hogy a külvilág sorrendi végrehajtást lásson
  → **sorrend-visszaállító buffer**

SORONKÍVÜLI VÉGREHAJTÁS HOZZÁVALÓI

- Több utasítást is le kell hívnunk, melyek közül válogatunk.
  - Ezeket tárolni kell valahol:
    → **utasítástároló**
- Egy eljárás, ami eldönti az utasítások végrehajtási sorrendjét
  → **dinamikus ütemező**
- Egy jó ötlet, amivel az utasítások közötti függőségek csökkenthetők
  → **regiszterátnevezés**
- Egy trükk, hogy a külvilág sorrendi végrehajtást lásson
  → **sorrend-visszaállító buffer**

DINAMIKUS UTASÍTÁS-ÜTEMEZÉS

- Utasítások végrehajtási sorrendje:

  Adatáramlásos modell szerint!

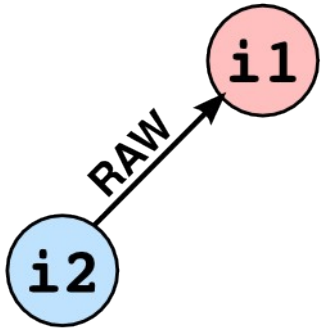
  - Az utasítássorozatból egy precedenciagráfot építünk
    - Csomópontok: utasítások
    - Élek: mely másik utasítást kell bevárni a végrehajtás előtt
  - Egy utasítás *végrehajtható,*

    ha minden utasítás, amitől függ, lefutott.
  - Ha egy utasítás végrehajtható, és van szabad egység, végrehajtjuk

PÉLDA

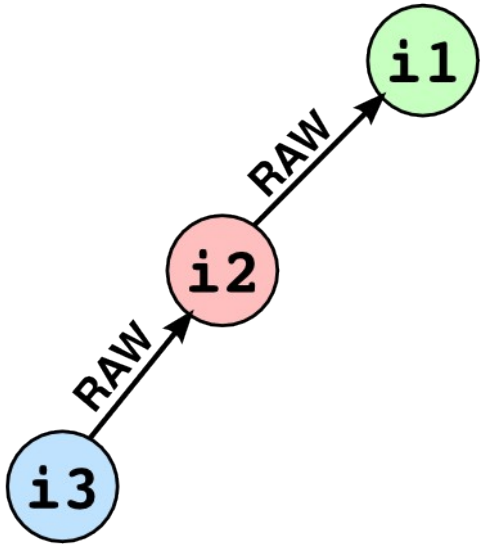

```
i1:  D2 ← MEM[R1]
i2:  D3 ← D2 * D0
i3:  MEM[R2] ← D3
i4:  R1 ← R1 + 8
i5:  R2 ← R2 + 8

i6:  D2 ← MEM[R1]
i7:  D3 ← D2 * D0
i8:  MEM[R2] ← D3
i9:  R1 ← R1 + 8
i10: R2 ← R2 + 8
```

PÉLDA



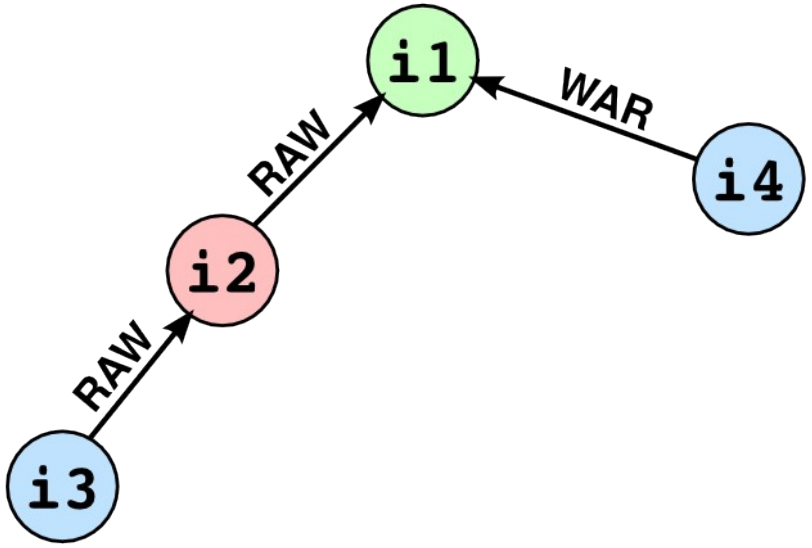

```
i1:   D2 ← MEM[R1]
i2:   D3 ← D2 * D0
i3:   MEM[R2] ← D3
i4:   R1 ← R1 + 8
i5:   R2 ← R2 + 8

i6:   D2 ← MEM[R1]
i7:   D3 ← D2 * D0
i8:   MEM[R2] ← D3
i9:   R1 ← R1 + 8
i10:  R2 ← R2 + 8
```

PÉLDA



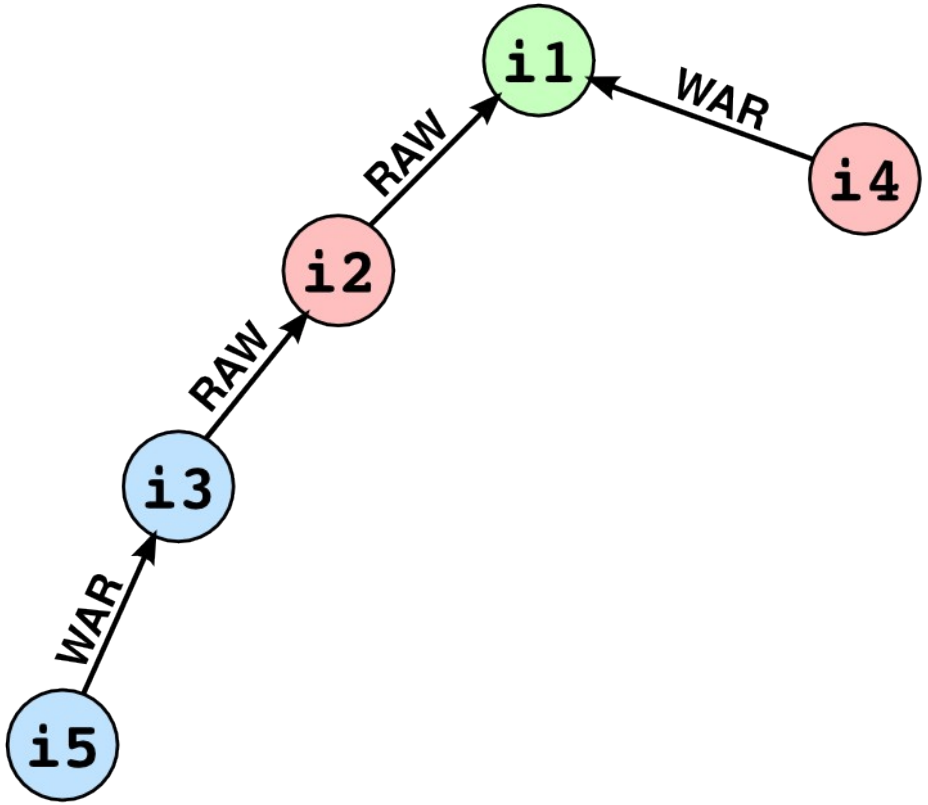

```
i1:   D2 ← MEM[R1]
i2:   D3 ← D2 * D0
i3:   MEM[R2] ← D3
i4:   R1 ← R1 + 8
i5:   R2 ← R2 + 8

i6:   D2 ← MEM[R1]
i7:   D3 ← D2 * D0
i8:   MEM[R2] ← D3
i9:   R1 ← R1 + 8
i10:  R2 ← R2 + 8
```

PÉLDA



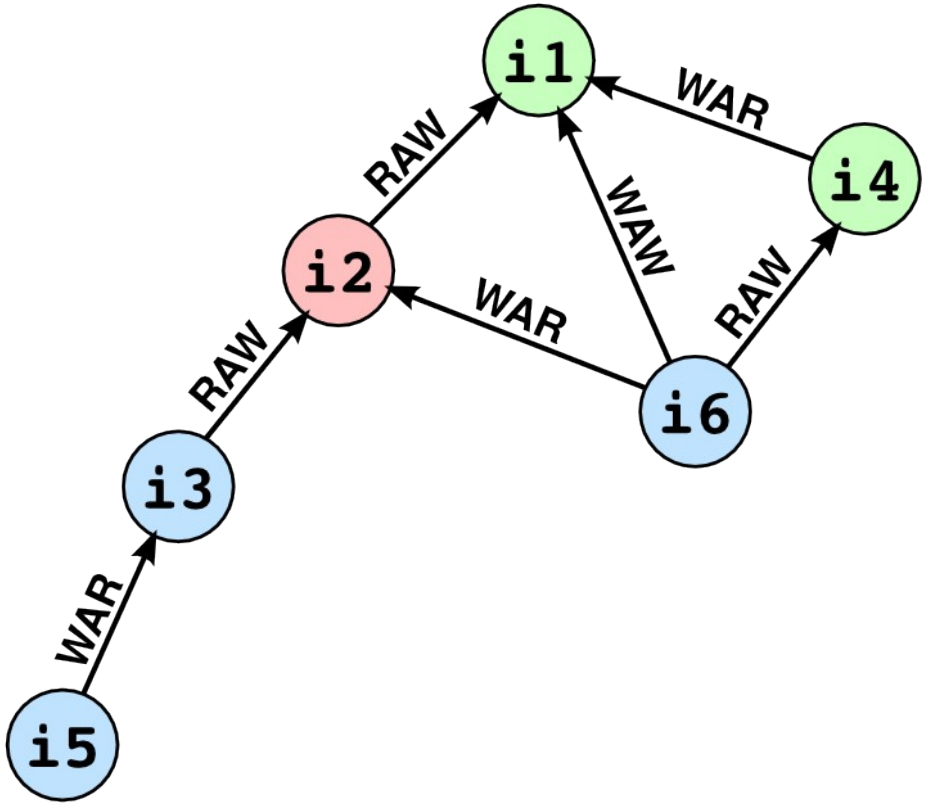

```
i1:   D2 ← MEM[R1]
i2:   D3 ← D2 * D0
i3:   MEM[R2] ← D3
i4:   R1 ← R1 + 8
i5:   R2 ← R2 + 8

i6:   D2 ← MEM[R1]
i7:   D3 ← D2 * D0
i8:   MEM[R2] ← D3
i9:   R1 ← R1 + 8
i10:  R2 ← R2 + 8
```

PÉLDA



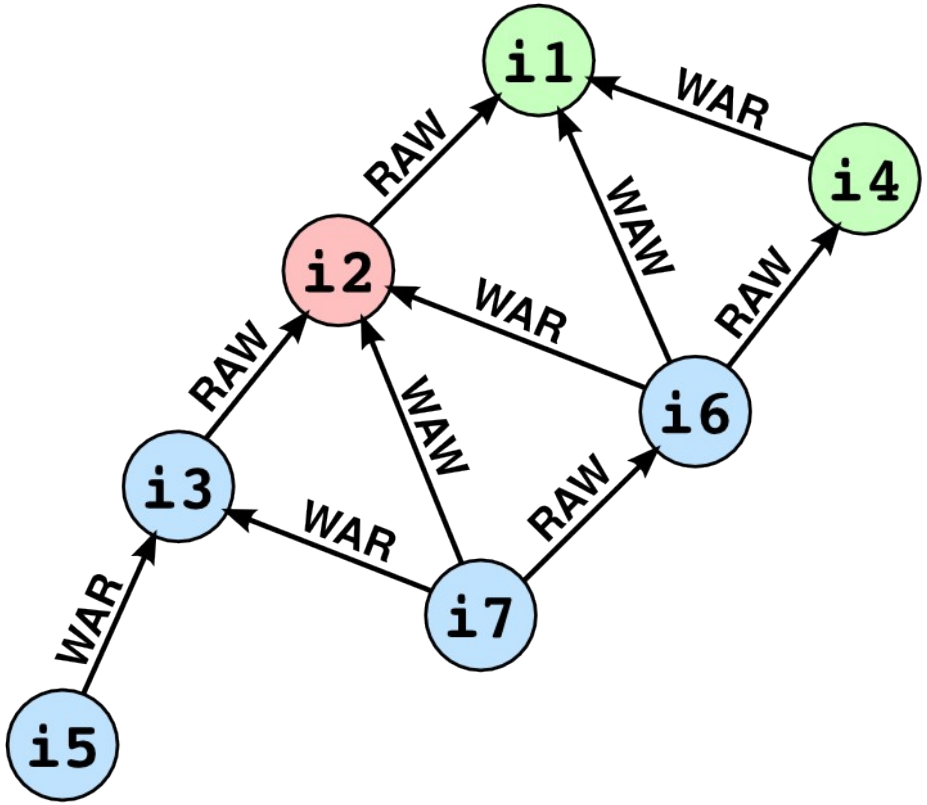

```
i1:   D2 ← MEM[R1]
i2:   D3 ← D2 * D0
i3:   MEM[R2] ← D3
i4:   R1 ← R1 + 8
i5:   R2 ← R2 + 8

i6:   D2 ← MEM[R1]
i7:   D3 ← D2 * D0
i8:   MEM[R2] ← D3
i9:   R1 ← R1 + 8
i10:  R2 ← R2 + 8
```

PÉLDA



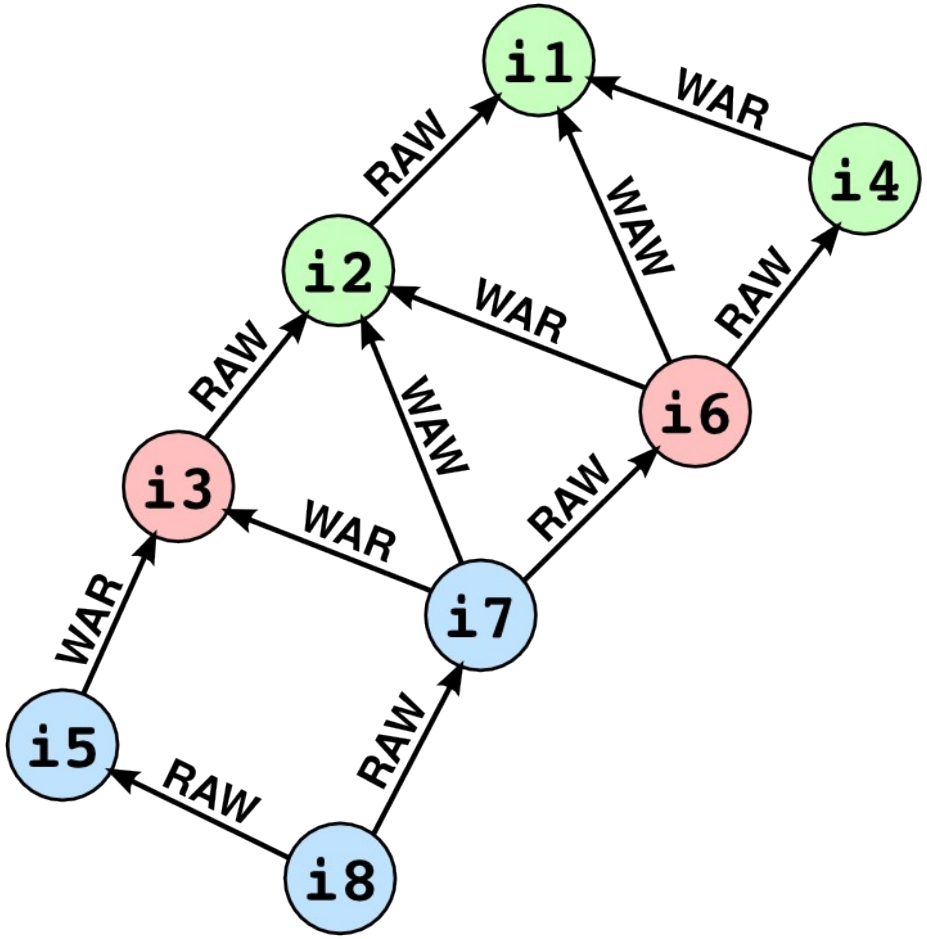

```
i1:   D2 ← MEM[R1]
i2:   D3 ← D2 * D0
i3:   MEM[R2] ← D3
i4:   R1 ← R1 + 8
i5:   R2 ← R2 + 8

i6:   D2 ← MEM[R1]
i7:   D3 ← D2 * D0
i8:   MEM[R2] ← D3
i9:   R1 ← R1 + 8
i10:  R2 ← R2 + 8
```

PÉLDA



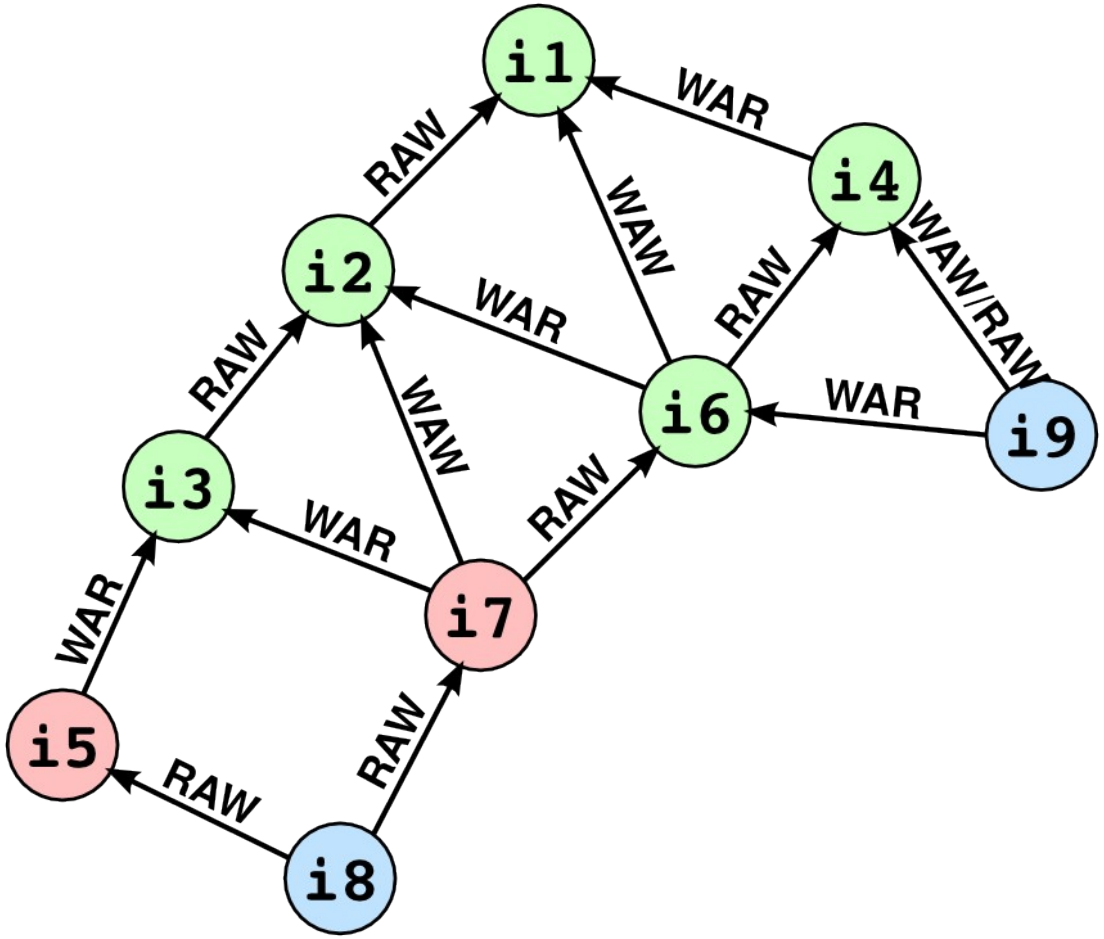

```
i1:   D2 ← MEM[R1]
i2:   D3 ← D2 * D0
i3:   MEM[R2] ← D3
i4:   R1 ← R1 + 8
i5:   R2 ← R2 + 8

i6:   D2 ← MEM[R1]
i7:   D3 ← D2 * D0
i8:   MEM[R2] ← D3
i9:   R1 ← R1 + 8
i10:  R2 ← R2 + 8
```

PÉLDA




```
i1:   D2 ← MEM[R1]
i2:   D3 ← D2 * D0
i3:   MEM[R2] ← D3
i4:   R1 ← R1 + 8
i5:   R2 ← R2 + 8

i6:   D2 ← MEM[R1]
i7:   D3 ← D2 * D0
i8:   MEM[R2] ← D3
i9:   R1 ← R1 + 8
i10:  R2 ← R2 + 8
```

PÉLDA




```
i1:   D2 ← MEM[R1]
i2:   D3 ← D2 * D0
i3:   MEM[R2] ← D3
i4:   R1 ← R1 + 8
i5:   R2 ← R2 + 8

i6:   D2 ← MEM[R1]
i7:   D3 ← D2 * D0
i8:   MEM[R2] ← D3
i9:   R1 ← R1 + 8
i10:  R2 ← R2 + 8
```

SORONKÍVÜLI VÉGREHAJTÁS HOZZÁVALÓI

- Több utasítást is le kell hívnunk, melyek közül válogatunk
  - Ezeket tárolni kell valahol:
    → **utasítástároló**
- Egy eljárás, ami eldönti az utasítások végrehajtási sorrendjét
    → **dinamikus ütemező**
- Egy jó ötlet, amivel az utasítások közötti függőségek csökkenthetők
    → **regiszterátnevezés**
- Egy trükk, hogy a külvilág sorrendi végrehajtást lásson
    → **sorrend-visszaállító buffer**

SORONKÍVÜLI VÉGREHAJTÁS HOZZÁVALÓI

- Több utasítást is le kell hívnunk, melyek közül válogatunk
  - Ezeket tárolni kell valahol:
    → **utasítástároló**
- Egy eljárás, ami eldönti az utasítások végrehajtási sorrendjét
    → **dinamikus ütemező**
- Egy jó ötlet, amivel az utasítások közötti függőségek csökkenthetők
    → **regiszterátnevezés**
- Egy trükk, hogy a külvilág sorrendi végrehajtást lásson
    → **sorrend-visszaállító buffer**

REGISZTERÁTNEVEZÉS

- Cél: a függőségi gráf „megritkítása”
- Milyen függőségeket ismerünk:
  - RAW:

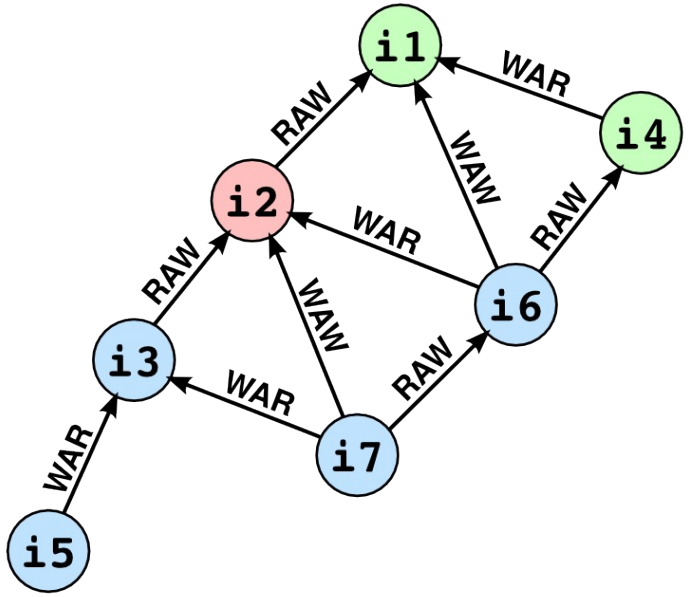
```
D3 ← D2 * D0
MEM[R2] ← D3
```

    → Valódi függőség, nem feloldható
  - WAR:

```
D3 ← D2 * D0
...
D2 ← MEM[R1]
```

  - WAW:

```
D3 ← D2 * D0
...
D3 ← MEM[R1]
```



- Meglepetés: a WAR és WAW függőségek feloldhatók!
  → Álfüggőségek

REGISZTERÁTNEVEZÉS

- Miért ír a programozó ilyen kódot?
  - WAR:

```
D3 ← D2 * D0
...
D2 ← MEM[R1]
```

  - WAW:

```
D3 ← D2 * D0
...
D3 ← MEM[R1]
```

- Újra felhasználja a regisztereket, mert nincs elég!
- Tegyünk a processzorba nagyon sok regisztert
  → **Fizikai regiszterek**
- De a programozó ezt nem látja
  → **Logikai/architekturális regiszterek**kel dolgozik
- A CPU a beolvasott programot röptében átírja
  → **Regiszterátnevezés**

REGISZTERÁTNEVEZÉS

- Eszköz: **Regiszter leképző tábla**
  - i. bejegyzése: az i. logikai regiszter melyik fizikaihoz van rendelve
- Eljárás:
  - Az utasítás operandusait fizikai regiszterekre cseréli
  - Az eredményt mindig egy új, nem használt fizikai regiszterbe teszi
  - Frissíti a regiszter leképző táblát

Eredeti:

```
i1:  D2 ← MEM[R1]
```

Átnevezés után:

```
i1:  U25 ← MEM[T3]
```

Regiszter leképző tábla:

Logikai reg.	Fizikai reg.
R0	T21
R1	T3
R2	T46
R3	T8
D0	U9
D1	U24
D2	U17
D3	U4

REGISZTERÁTNEVEZÉS

- Eszköz: **Regiszter leképző tábla**
  - i. bejegyzése: az i. logikai regiszter melyik fizikaihoz van rendelve
- Eljárás:
  - Az utasítás operandusait fizikai regiszterekre cseréli
  - Az eredményt mindig egy új, nem használt fizikai regiszterbe teszi
  - Frissíti a regiszter leképző táblát

Eredeti:

```
i1:  D2 ← MEM[R1]
```

Átnevezés után:

```
i1:  U25 ← MEM[T3]
```

Regiszter leképző tábla:

Logikai reg.	Fizikai reg.
R0	T21
R1	T3
R2	T46
R3	T8
D0	U9
D1	U24
D2	**U25**
D3	U4

REGISZTERÁTNEVEZÉS

- Eszköz: **Regiszter leképző tábla**
  - i. bejegyzése: az i. logikai regiszter melyik fizikaihoz van rendelve
- Eljárás:
  - Az utasítás operandusait fizikai regiszterekre cseréli
  - Az eredményt mindig egy új, nem használt fizikai regiszterbe teszi
  - Frissíti a regiszter leképző táblát

Eredeti:

```
i1:  D2 ← MEM[R1]
i2:  D3 ← D2 * D0
```

Átnevezés után:

```
i1:  U25 ← MEM[T3]
i2:  U26 ← U25 * U9
```

Regiszter leképző tábla:

Logikai reg.	Fizikai reg.
R0	T21
R1	T3
R2	T46
R3	T8
D0	U9
D1	U24
D2	U25
D3	U4

REGISZTERÁTNEVEZÉS

- Eszköz: **Regiszter leképző tábla**
  - i. bejegyzése: az i. logikai regiszter melyik fizikaihoz van rendelve
- Eljárás:
  - Az utasítás operandusait fizikai regiszterekre cseréli
  - Az eredményt mindig egy új, nem használt fizikai regiszterbe teszi
  - Frissíti a regiszter leképző táblát

Eredeti:

```
i1:  D2 ← MEM[R1]
i2:  D3 ← D2 * D0
```

Átnevezés után:

```
i1:  U25 ← MEM[T3]
i2:  U26 ← U25 * U9
```

Regiszter leképző tábla:

Logikai reg.	Fizikai reg.
R0	T21
R1	T3
R2	T46
R3	T8
D0	U9
D1	U24
D2	U25
D3	**U26**

REGISZTERÁTNEVEZÉS

- Eszköz: **Regiszter leképző tábla**
  - i. bejegyzése: az i. logikai regiszter melyik fizikaihoz van rendelve
- Eljárás:
  - Az utasítás operandusait fizikai regiszterekre cseréli
  - Az eredményt mindig egy új, nem használt fizikai regiszterbe teszi
  - Frissíti a regiszter leképző táblát

Eredeti:

```
i1:  D2 ← MEM[R1]
i2:  D3 ← D2 * D0
i3:  MEM[R2] ← D3
```

Átnevezés után:

```
i1:  U25 ← MEM[T3]
i2:  U26 ← U25 * U9
i3:  MEM[T46] ← U26
```

Regiszter leképző tábla:

Logikai reg.	Fizikai reg.
R0	T21
R1	T3
R2	T46
R3	T8
D0	U9
D1	U24
D2	U25
D3	U26

REGISZTERÁTNEVEZÉS

- Eszköz: **Regiszter leképző tábla**
  - i. bejegyzése: az i. logikai regiszter melyik fizikaihoz van rendelve
- Eljárás:
  - Az utasítás operandusait fizikai regiszterekre cseréli
  - Az eredményt mindig egy új, nem használt fizikai regiszterbe teszi
  - Frissíti a regiszter leképző táblát

Eredeti:

```
i1:  D2 ← MEM[R1]
i2:  D3 ← D2 * D0
i3:  MEM[R2] ← D3
i4:  R1 ← R1 + 8
```

Átnevezés után:

```
i1:  U25 ← MEM[T3]
i2:  U26 ← U25 * U9
i3:  MEM[T46] ← U26
i4:  T47 ← T3 + 8
```

Regiszter leképző tábla:

Logikai reg.	Fizikai reg.
R0	T21
R1	T3
R2	T46
R3	T8
D0	U9
D1	U24
D2	U25
D3	U26

REGISZTERÁTNEVEZÉS

- Eszköz: **Regiszter leképző tábla**
  - i. bejegyzése: az i. logikai regiszter melyik fizikaihoz van rendelve
- Eljárás:
  - Az utasítás operandusait fizikai regiszterekre cseréli
  - Az eredményt mindig egy új, nem használt fizikai regiszterbe teszi
  - Frissíti a regiszter leképző táblát

Eredeti:

```
i1:  D2 ← MEM[R1]
i2:  D3 ← D2 * D0
i3:  MEM[R2] ← D3
i4:  R1 ← R1 + 8
```

Átnevezés után:

```
i1:  U25 ← MEM[T3]
i2:  U26 ← U25 * U9
i3:  MEM[T46] ← U26
i4:  T47 ← T3 + 8
```

Regiszter leképző tábla:

Logikai reg.	Fizikai reg.
R0	T21
R1	**T47**
R2	T46
R3	T8
D0	U9
D1	U24
D2	U25
D3	U26

REGISZTERÁTNEVEZÉS

- Eszköz: **Regiszter leképző tábla**
  - i. bejegyzése: az i. logikai regiszter melyik fizikaihoz van rendelve
- Eljárás:
  - Az utasítás operandusait fizikai regiszterekre cseréli
  - Az eredményt mindig egy új, nem használt fizikai regiszterbe teszi
  - Frissíti a regiszter leképző táblát

Eredeti:

```
i1:  D2 ← MEM[R1]
i2:  D3 ← D2 * D0
i3:  MEM[R2] ← D3
i4:  R1 ← R1 + 8
i5:  R2 ← R2 + 8
```

Átnevezés után:

```
i1:  U25 ← MEM[T3]
i2:  U26 ← U25 * U9
i3:  MEM[T46] ← U26
i4:  T47 ← T3 + 8
i5:  T48 ← T46 + 8
```

Regiszter leképző tábla:

Logikai reg.	Fizikai reg.
R0	T21
R1	T47
R2	T46
R3	T8
D0	U9
D1	U24
D2	U25
D3	U26

REGISZTERÁTNEVEZÉS

- Eszköz: **Regiszter leképző tábla**
  - i. bejegyzése: az i. logikai regiszter melyik fizikaihoz van rendelve
- Eljárás:
  - Az utasítás operandusait fizikai regiszterekre cseréli
  - Az eredményt mindig egy új, nem használt fizikai regiszterbe teszi
  - Frissíti a regiszter leképző táblát

Eredeti:

```
i1:  D2 ← MEM[R1]
i2:  D3 ← D2 * D0
i3:  MEM[R2] ← D3
i4:  R1 ← R1 + 8
i5:  R2 ← R2 + 8
```

Átnevezés után:

```
i1:  U25 ← MEM[T3]
i2:  U26 ← U25 * U9
i3:  MEM[T46] ← U26
i4:  T47 ← T3 + 8
i5:  T48 ← T46 + 8
```

Regiszter leképző tábla:

Logikai reg.	Fizikai reg.
R0	T21
R1	T47
R2	T48
R3	T8
D0	U9
D1	U24
D2	U25
D3	U26

REGISZTERÁTNEVEZÉS

- Eszköz: **Regiszter leképző tábla**
  - i. bejegyzése: az i. logikai regiszter melyik fizikaihoz van rendelve
- Eljárás:
  - Az utasítás operandusait fizikai regiszterekre cseréli
  - Az eredményt mindig egy új, nem használt fizikai regiszterbe teszi
  - Frissíti a regiszter leképző táblát

Eredeti:

```
i1:  D2 ← MEM[R1]
i2:  D3 ← D2 * D0
i3:  MEM[R2] ← D3
i4:  R1 ← R1 + 8
i5:  R2 ← R2 + 8

i6:  D2 ← MEM[R1]
```

Átnevezés után:

```
i1:  U25 ← MEM[T3]
i2:  U26 ← U25 * U9
i3:  MEM[T46] ← U26
i4:  T47 ← T3 + 8
i5:  T48 ← T46 + 8

i6:  U27 ← MEM[T47]
```

Regiszter leképző tábla:

Logikai reg.	Fizikai reg.
R0	T21
R1	T47
R2	T48
R3	T8
D0	U9
D1	U24
D2	U25
D3	U26

REGISZTERÁTNEVEZÉS

- Eszköz: **Regiszter leképző tábla**
  - i. bejegyzése: az i. logikai regiszter melyik fizikaihoz van rendelve
- Eljárás:
  - Az utasítás operandusait fizikai regiszterekre cseréli
  - Az eredményt mindig egy új, nem használt fizikai regiszterbe teszi
  - Frissíti a regiszter leképző táblát

Eredeti:

```
i1:  D2 ← MEM[R1]
i2:  D3 ← D2 * D0
i3:  MEM[R2] ← D3
i4:  R1 ← R1 + 8
i5:  R2 ← R2 + 8

i6:  D2 ← MEM[R1]
```

Átnevezés után:

```
i1:  U25 ← MEM[T3]
i2:  U26 ← U25 * U9
i3:  MEM[T46] ← U26
i4:  T47 ← T3 + 8
i5:  T48 ← T46 + 8

i6:  U27 ← MEM[T47]
```

Regiszter leképző tábla:

Logikai reg.	Fizikai reg.
R0	T21
R1	T47
R2	T48
R3	T8
D0	U9
D1	U24
D2	**U27**
D3	U26

REGISZTERÁTNEVEZÉS

- Eszköz: **Regiszter leképző tábla**
  - i. bejegyzése: az i. logikai regiszter melyik fizikaihoz van rendelve
- Eljárás:
  - Az utasítás operandusait fizikai regiszterekre cseréli
  - Az eredményt mindig egy új, nem használt fizikai regiszterbe teszi
  - Frissíti a regiszter leképző táblát

Eredeti:

```
i1:  D2 ← MEM[R1]
i2:  D3 ← D2 * D0
i3:  MEM[R2] ← D3
i4:  R1 ← R1 + 8
i5:  R2 ← R2 + 8

i6:  D2 ← MEM[R1]
i7:  D3 ← D2 * D0
```

Átnevezés után:

```
i1:  U25 ← MEM[T3]
i2:  U26 ← U25 * U9
i3:  MEM[T46] ← U26
i4:  T47 ← T3 + 8
i5:  T48 ← T46 + 8

i6:  U27 ← MEM[T47]
i7:  U28 ← U27 * U9
```

Regiszter leképző tábla:

Logikai reg.	Fizikai reg.
R0	T21
R1	T47
R2	T48
R3	T8
D0	U9
D1	U24
D2	U27
D3	U26

REGISZTERÁTNEVEZÉS

- Eszköz: **Regiszter leképző tábla**
  - i. bejegyzése: az i. logikai regiszter melyik fizikaihoz van rendelve
- Eljárás:
  - Az utasítás operandusait fizikai regiszterekre cseréli
  - Az eredményt mindig egy új, nem használt fizikai regiszterbe teszi
  - Frissíti a regiszter leképző táblát

Eredeti:

```
i1:  D2 ← MEM[R1]
i2:  D3 ← D2 * D0
i3:  MEM[R2] ← D3
i4:  R1 ← R1 + 8
i5:  R2 ← R2 + 8

i6:  D2 ← MEM[R1]
i7:  D3 ← D2 * D0
```

Átnevezés után:

```
i1:  U25 ← MEM[T3]
i2:  U26 ← U25 * U9
i3:  MEM[T46] ← U26
i4:  T47 ← T3 + 8
i5:  T48 ← T46 + 8

i6:  U27 ← MEM[T47]
i7:  U28 ← U27 * U9
```

Regiszter leképző tábla:

Logikai reg.	Fizikai reg.
R0	T21
R1	T47
R2	T48
R3	T8
D0	U9
D1	U24
D2	U27
D3	**U28**

REGISZTERÁTNEVEZÉS

- Eszköz: **Regiszter leképző tábla**
  - i. bejegyzése: az i. logikai regiszter melyik fizikaihoz van rendelve
- Eljárás:
  - Az utasítás operandusait fizikai regiszterekre cseréli
  - Az eredményt mindig egy új, nem használt fizikai regiszterbe teszi
  - Frissíti a regiszter leképző táblát

Eredeti:

```
i1:  D2 ← MEM[R1]
i2:  D3 ← D2 * D0
i3:  MEM[R2] ← D3
i4:  R1 ← R1 + 8
i5:  R2 ← R2 + 8

i6:  D2 ← MEM[R1]
i7:  D3 ← D2 * D0
i8:  MEM[R2] ← D3
```

Átnevezés után:

```
i1:  U25 ← MEM[T3]
i2:  U26 ← U25 * U9
i3:  MEM[T46] ← U26
i4:  T47 ← T3 + 8
i5:  T48 ← T46 + 8

i6:  U27 ← MEM[T47]
i7:  U28 ← U27 * U9
i8:  MEM[T48] ← U28
```

Regiszter leképző tábla:

Logikai reg.	Fizikai reg.
R0	T21
R1	T47
R2	T48
R3	T8
D0	U9
D1	U24
D2	U27
D3	U28

REGISZTERÁTNEVEZÉS

- Eszköz: **Regiszter leképző tábla**
  - i. bejegyzése: az i. logikai regiszter melyik fizikaihoz van rendelve
- Eljárás:
  - Az utasítás operandusait fizikai regiszterekre cseréli
  - Az eredményt mindig egy új, nem használt fizikai regiszterbe teszi
  - Frissíti a regiszter leképző táblát

Eredeti:

```
i1:  D2 ← MEM[R1]
i2:  D3 ← D2 * D0
i3:  MEM[R2] ← D3
i4:  R1 ← R1 + 8
i5:  R2 ← R2 + 8

i6:  D2 ← MEM[R1]
i7:  D3 ← D2 * D0
i8:  MEM[R2] ← D3
i9:  R1 ← R1 + 8
```

Átnevezés után:

```
i1:  U25 ← MEM[T3]
i2:  U26 ← U25 * U9
i3:  MEM[T46] ← U26
i4:  T47 ← T3 + 8
i5:  T48 ← T46 + 8

i6:  U27 ← MEM[T47]
i7:  U28 ← U27 * U9
i8:  MEM[T48] ← U28
i9:  T49 ← T47 + 8
```

Regiszter leképző tábla:

Logikai reg.	Fizikai reg.
R0	T21
R1	T47
R2	T48
R3	T8
D0	U9
D1	U24
D2	U27
D3	U28

REGISZTERÁTNEVEZÉS

- Eszköz: **Regiszter leképző tábla**
  - i. bejegyzése: az i. logikai regiszter melyik fizikaihoz van rendelve
- Eljárás:
  - Az utasítás operandusait fizikai regiszterekre cseréli
  - Az eredményt mindig egy új, nem használt fizikai regiszterbe teszi
  - Frissíti a regiszter leképző táblát

Eredeti:

```
i1:  D2 ← MEM[R1]
i2:  D3 ← D2 * D0
i3:  MEM[R2] ← D3
i4:  R1 ← R1 + 8
i5:  R2 ← R2 + 8

i6:  D2 ← MEM[R1]
i7:  D3 ← D2 * D0
i8:  MEM[R2] ← D3
i9:  R1 ← R1 + 8
```

Átnevezés után:

```
i1:  U25 ← MEM[T3]
i2:  U26 ← U25 * U9
i3:  MEM[T46] ← U26
i4:  T47 ← T3 + 8
i5:  T48 ← T46 + 8

i6:  U27 ← MEM[T47]
i7:  U28 ← U27 * U9
i8:  MEM[T48] ← U28
i9:  T49 ← T47 + 8
```

Regiszter leképző tábla:

Logikai reg.	Fizikai reg.
R0	T21
R1	**T49**
R2	T48
R3	T8
D0	U9
D1	U24
D2	U27
D3	U28

REGISZTERÁTNEVEZÉS

- Eszköz: **Regiszter leképző tábla**
  - i. bejegyzése: az i. logikai regiszter melyik fizikaihoz van rendelve
- Eljárás:
  - Az utasítás operandusait fizikai regiszterekre cseréli
  - Az eredményt mindig egy új, nem használt fizikai regiszterbe teszi
  - Frissíti a regiszter leképző táblát

Eredeti:

```
i1:   D2 ← MEM[R1]
i2:   D3 ← D2 * D0
i3:   MEM[R2] ← D3
i4:   R1 ← R1 + 8
i5:   R2 ← R2 + 8

i6:   D2 ← MEM[R1]
i7:   D3 ← D2 * D0
i8:   MEM[R2] ← D3
i9:   R1 ← R1 + 8
i10:  R2 ← R2 + 8
```

Átnevezés után:

```
i1:   U25 ← MEM[T3]
i2:   U26 ← U25 * U9
i3:   MEM[T46] ← U26
i4:   T47 ← T3 + 8
i5:   T48 ← T46 + 8

i6:   U27 ← MEM[T47]
i7:   U28 ← U27 * U9
i8:   MEM[T48] ← U28
i9:   T49 ← T47 + 8
i10:  T50 ← T48 + 8
```

Regiszter leképző tábla:

Logikai reg.	Fizikai reg.
R0	T21
R1	T49
R2	T48
R3	T8
D0	U9
D1	U24
D2	U27
D3	U28

REGISZTERÁTNEVEZÉS

- Eszköz: **Regiszter leképző tábla**
  - i. bejegyzése: az i. logikai regiszter melyik fizikaihoz van rendelve
- Eljárás:
  - Az utasítás operandusait fizikai regiszterekre cseréli
  - Az eredményt mindig egy új, nem használt fizikai regiszterbe teszi
  - Frissíti a regiszter leképző táblát

Eredeti:

```
i1:   D2 ← MEM[R1]
i2:   D3 ← D2 * D0
i3:   MEM[R2] ← D3
i4:   R1 ← R1 + 8
i5:   R2 ← R2 + 8

i6:   D2 ← MEM[R1]
i7:   D3 ← D2 * D0
i8:   MEM[R2] ← D3
i9:   R1 ← R1 + 8
i10:  R2 ← R2 + 8
```

Átnevezés után:

```
i1:   U25 ← MEM[T3]
i2:   U26 ← U25 * U9
i3:   MEM[T46] ← U26
i4:   T47 ← T3 + 8
i5:   T48 ← T46 + 8

i6:   U27 ← MEM[T47]
i7:   U28 ← U27 * U9
i8:   MEM[T48] ← U28
i9:   T49 ← T47 + 8
i10:  T50 ← T48 + 8
```

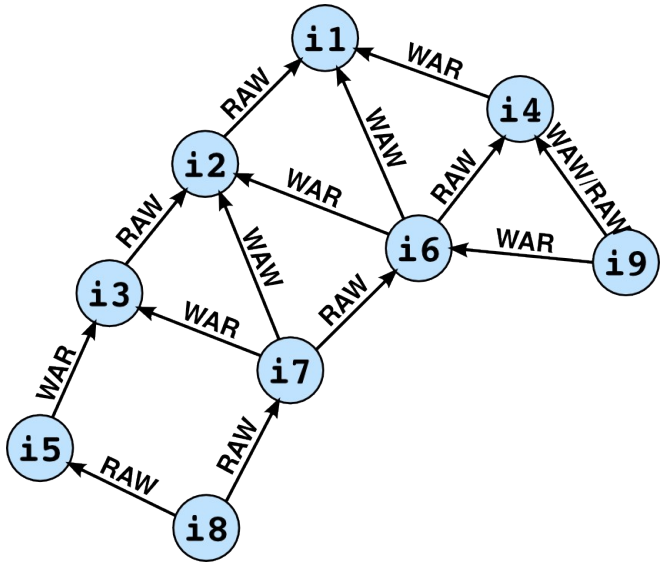
Regiszter leképző tábla:

Logikai reg.	Fizikai reg.
R0	T21
R1	T49
R2	**T50**
R3	T8
D0	U9
D1	U24
D2	U27
D3	U28

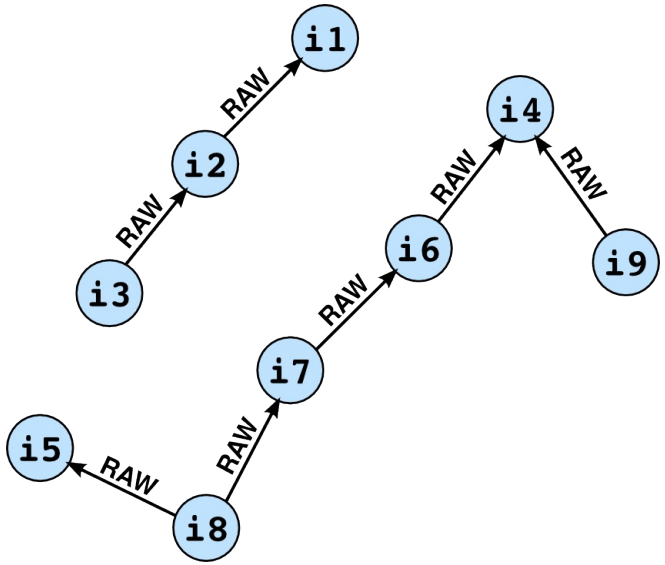
REGISZTERÁTNEVEZÉS

- Eredmény:
  - Eltűnt minden WAW és WAR!
  - ...hiszen az eredményt mindig „tiszta” regiszterbe tettük
- A precedenciagráf:

Regiszterátnevezés előtt:



Regiszterátnevezés után:

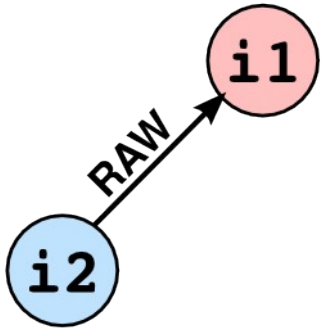


PÉLDA

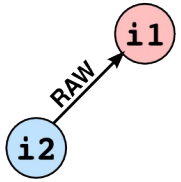

```
i1:  U25 ← MEM[T3]
i2:  U26 ← U25 * U9
i3:  MEM[T46] ← U26
i4:  T47 ← T3 + 8
i5:  T48 ← T46 + 8

i6:  U27 ← MEM[T47]
i7:  U28 ← U27 * U9
i8:  MEM[T48] ← U28
i9:  T49 ← T47 + 8
i10: T50 ← T48 + 8
```


```
i1:  D2 ← MEM[R1]
i2:  D3 ← D2 * D0
i3:  MEM[R2] ← D3
i4:  R1 ← R1 + 8
i5:  R2 ← R2 + 8

i6:  D2 ← MEM[R1]
i7:  D3 ← D2 * D0
i8:  MEM[R2] ← D3
i9:  R1 ← R1 + 8
i10: R2 ← R2 + 8
```

PÉLDA

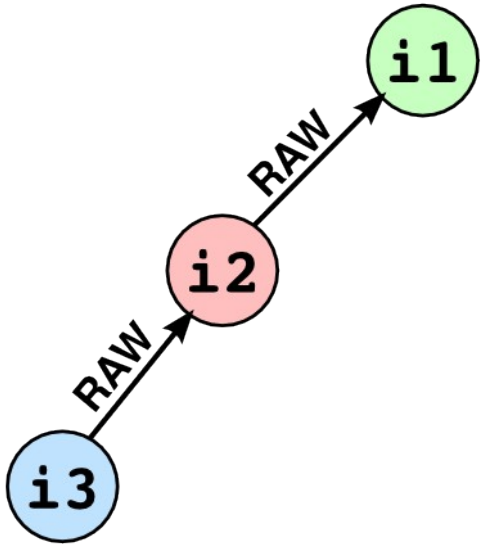



```
i1:   U25 ← MEM[T3]
i2:   U26 ← U25 * U9
i3:   MEM[T46] ← U26
i4:   T47 ← T3 + 8
i5:   T48 ← T46 + 8

i6:   U27 ← MEM[T47]
i7:   U28 ← U27 * U9
i8:   MEM[T48] ← U28
i9:   T49 ← T47 + 8
i10:  T50 ← T48 + 8
```

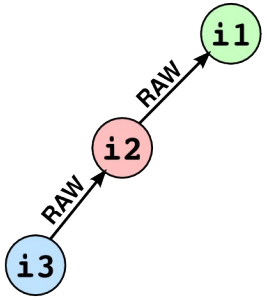



```
i1:   D2 ← MEM[R1]
i2:   D3 ← D2 * D0
i3:   MEM[R2] ← D3
i4:   R1 ← R1 + 8
i5:   R2 ← R2 + 8

i6:   D2 ← MEM[R1]
i7:   D3 ← D2 * D0
i8:   MEM[R2] ← D3
i9:   R1 ← R1 + 8
i10:  R2 ← R2 + 8
```

PÉLDA

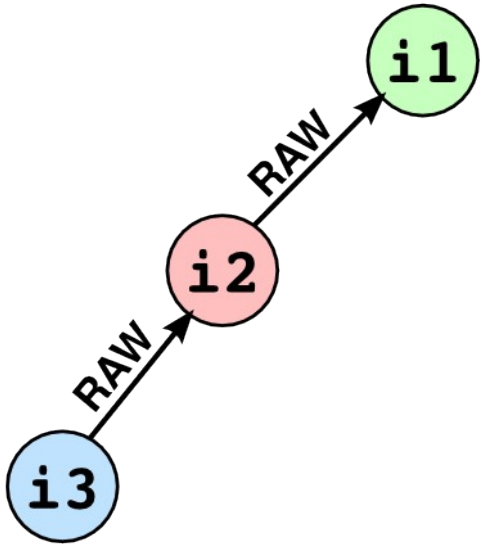


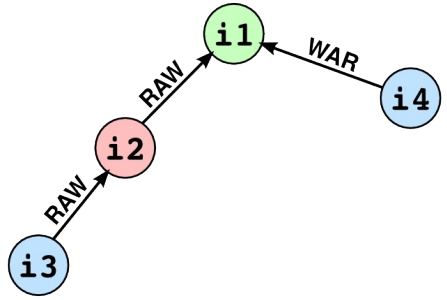

```
i1:   U25 ← MEM[T3]
i2:   U26 ← U25 * U9
i3:   MEM[T46] ← U26
i4:   T47 ← T3 + 8
i5:   T48 ← T46 + 8

i6:   U27 ← MEM[T47]
i7:   U28 ← U27 * U9
i8:   MEM[T48] ← U28
i9:   T49 ← T47 + 8
i10:  T50 ← T48 + 8
```




```
i1:   D2 ← MEM[R1]
i2:   D3 ← D2 * D0
i3:   MEM[R2] ← D3
i4:   R1 ← R1 + 8
i5:   R2 ← R2 + 8

i6:   D2 ← MEM[R1]
i7:   D3 ← D2 * D0
i8:   MEM[R2] ← D3
i9:   R1 ← R1 + 8
i10:  R2 ← R2 + 8
```

PÉLDA

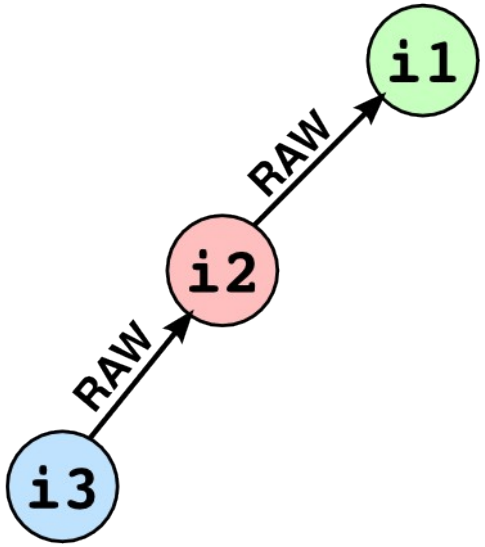


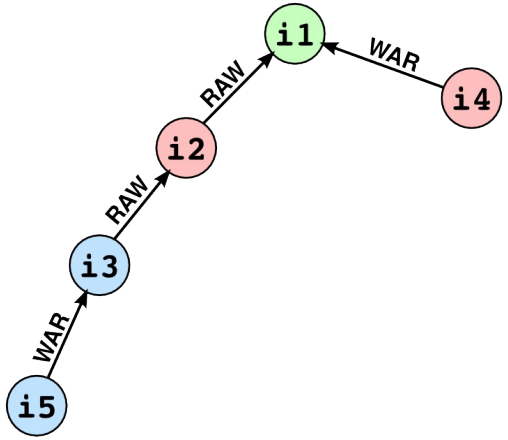



```
i1:   U25 ← MEM[T3]
i2:   U26 ← U25 * U9
i3:   MEM[T46] ← U26
i4:   T47 ← T3 + 8
i5:   T48 ← T46 + 8

i6:   U27 ← MEM[T47]
i7:   U28 ← U27 * U9
i8:   MEM[T48] ← U28
i9:   T49 ← T47 + 8
i10:  T50 ← T48 + 8
```

```
i1:   D2 ← MEM[R1]
i2:   D3 ← D2 * D0
i3:   MEM[R2] ← D3
i4:   R1 ← R1 + 8
i5:   R2 ← R2 + 8

i6:   D2 ← MEM[R1]
i7:   D3 ← D2 * D0
i8:   MEM[R2] ← D3
i9:   R1 ← R1 + 8
i10:  R2 ← R2 + 8
```

PÉLDA

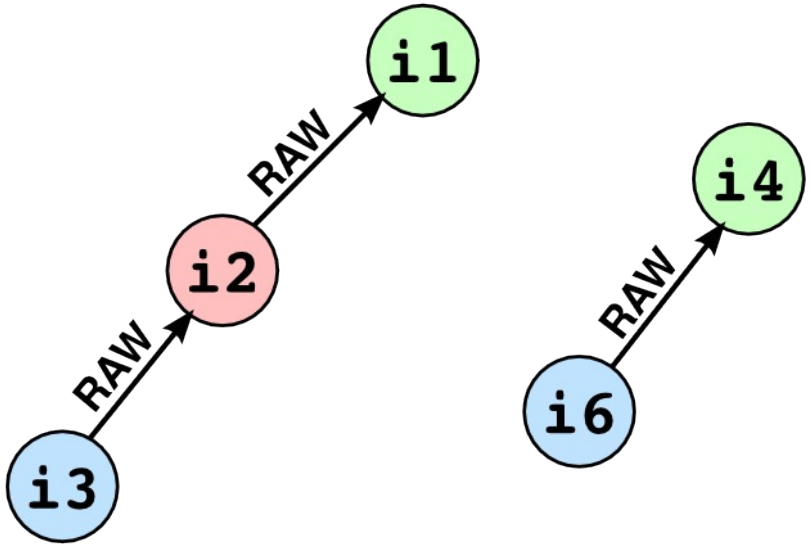


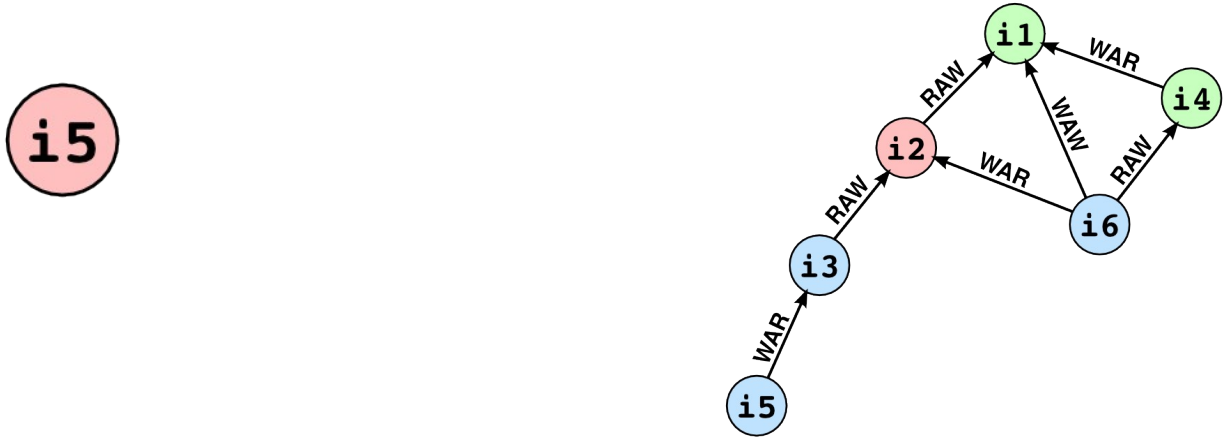



```
i1:   U25 ← MEM[T3]
i2:   U26 ← U25 * U9
i3:   MEM[T46] ← U26
i4:   T47 ← T3 + 8
i5:   T48 ← T46 + 8

i6:   U27 ← MEM[T47]
i7:   U28 ← U27 * U9
i8:   MEM[T48] ← U28
i9:   T49 ← T47 + 8
i10:  T50 ← T48 + 8
```

```
i1:   D2 ← MEM[R1]
i2:   D3 ← D2 * D0
i3:   MEM[R2] ← D3
i4:   R1 ← R1 + 8
i5:   R2 ← R2 + 8

i6:   D2 ← MEM[R1]
i7:   D3 ← D2 * D0
i8:   MEM[R2] ← D3
i9:   R1 ← R1 + 8
i10:  R2 ← R2 + 8
```

PÉLDA

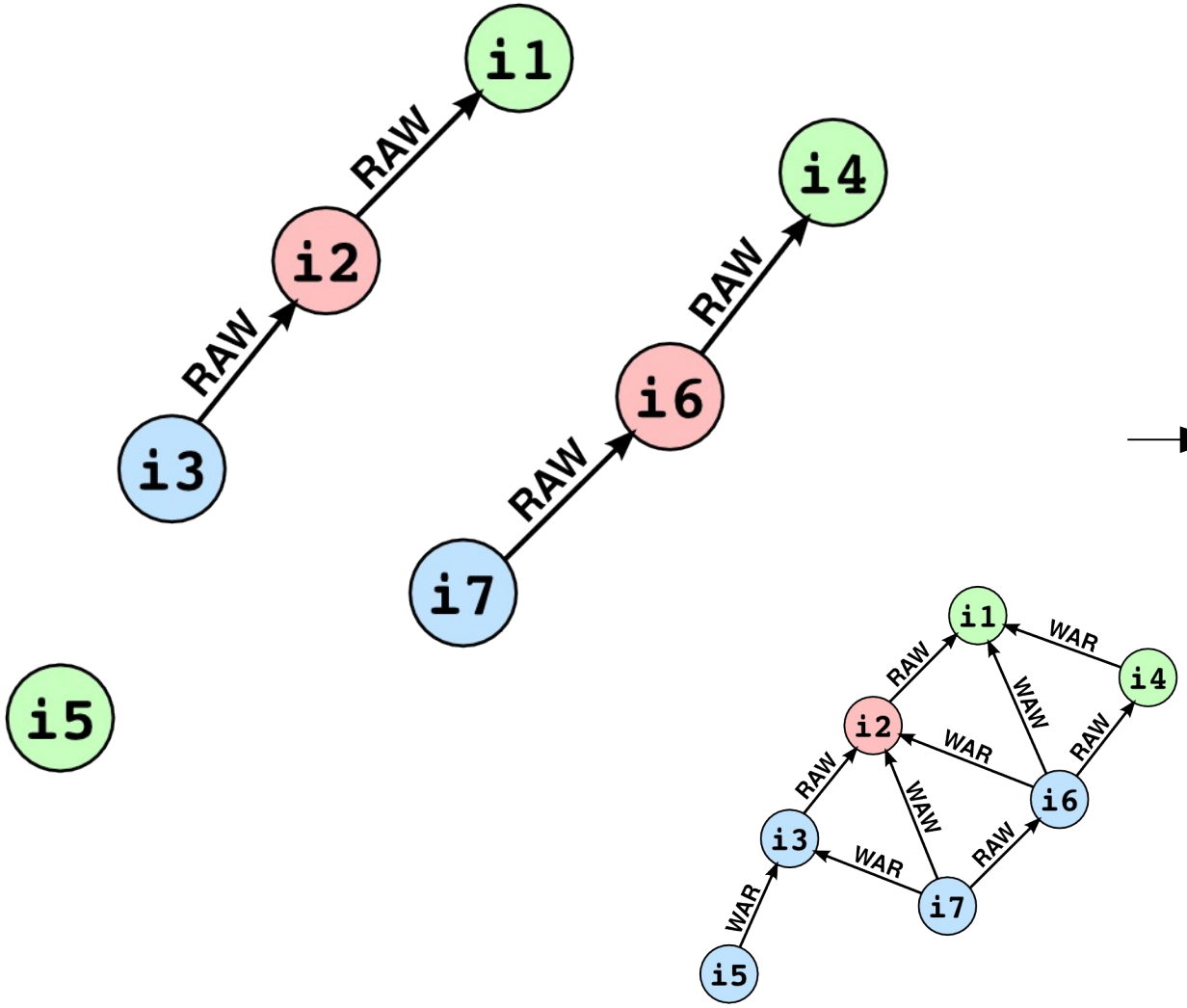





```
i1:   U25 ← MEM[T3]
i2:   U26 ← U25 * U9
i3:   MEM[T46] ← U26
i4:   T47 ← T3 + 8
i5:   T48 ← T46 + 8

i6:   U27 ← MEM[T47]
i7:   U28 ← U27 * U9
i8:   MEM[T48] ← U28
i9:   T49 ← T47 + 8
i10:  T50 ← T48 + 8
```

```
i1:   D2 ← MEM[R1]
i2:   D3 ← D2 * D0
i3:   MEM[R2] ← D3
i4:   R1 ← R1 + 8
i5:   R2 ← R2 + 8

i6:   D2 ← MEM[R1]
i7:   D3 ← D2 * D0
i8:   MEM[R2] ← D3
i9:   R1 ← R1 + 8
i10:  R2 ← R2 + 8
```

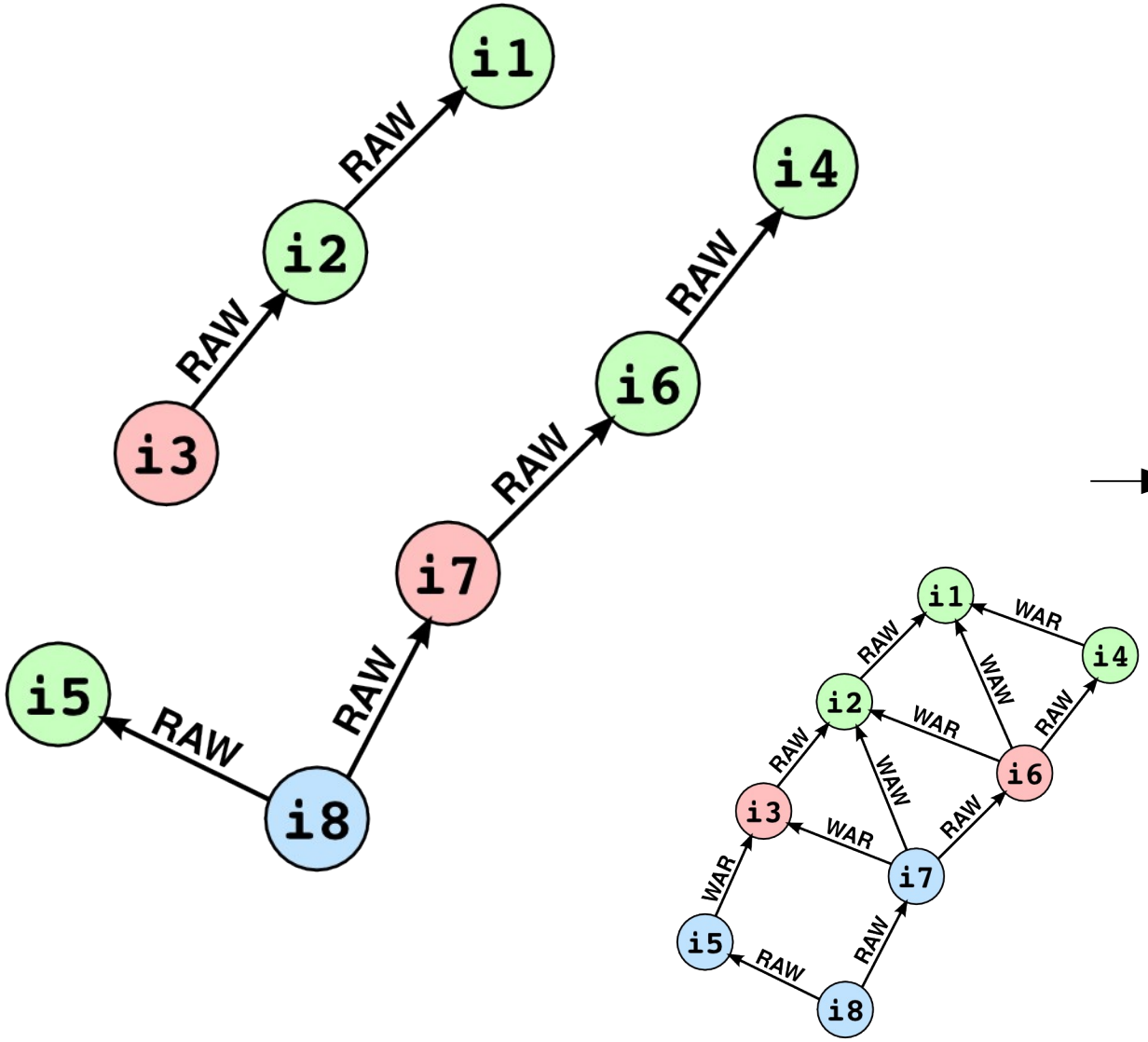
PÉLDA




```
i1:   U25 ← MEM[T3]
i2:   U26 ← U25 * U9
i3:   MEM[T46] ← U26
i4:   T47 ← T3 + 8
i5:   T48 ← T46 + 8

i6:   U27 ← MEM[T47]
i7:   U28 ← U27 * U9
i8:   MEM[T48] ← U28
i9:   T49 ← T47 + 8
i10:  T50 ← T48 + 8
```

```
i1:   D2 ← MEM[R1]
i2:   D3 ← D2 * D0
i3:   MEM[R2] ← D3
i4:   R1 ← R1 + 8
i5:   R2 ← R2 + 8

i6:   D2 ← MEM[R1]
i7:   D3 ← D2 * D0
i8:   MEM[R2] ← D3
i9:   R1 ← R1 + 8
i10:  R2 ← R2 + 8
```

PÉLDA




```
i1:   U25 ← MEM[T3]
i2:   U26 ← U25 * U9
i3:   MEM[T46] ← U26
i4:   T47 ← T3 + 8
i5:   T48 ← T46 + 8

i6:   U27 ← MEM[T47]
i7:   U28 ← U27 * U9
i8:   MEM[T48] ← U28
i9:   T49 ← T47 + 8
i10:  T50 ← T48 + 8
```



```
i1:   D2 ← MEM[R1]
i2:   D3 ← D2 * D0
i3:   MEM[R2] ← D3
i4:   R1 ← R1 + 8
i5:   R2 ← R2 + 8

i6:   D2 ← MEM[R1]
i7:   D3 ← D2 * D0
i8:   MEM[R2] ← D3
i9:   R1 ← R1 + 8
i10:  R2 ← R2 + 8
```

REGISZTERÁTNEVEZÉS

- Konklúzió:
  - Hatásos!
  - Minél nagyobb az utasítástároló, annál inkább

SORONKÍVÜLI VÉGREHAJTÁS HOZZÁVALÓI

- Több utasítást is le kell hívnunk, melyek közül válogatunk
  - Ezeket tárolni kell valahol:
    → **utasítástároló**
- Egy eljárás, ami eldönti az utasítások végrehajtási sorrendjét
    → **dinamikus ütemező**
- Egy jó ötlet, amivel az utasítások közötti függőségek csökkenthetők
    → **regiszterátnevezés**
- Egy trükk, hogy a külvilág sorrendi végrehajtást lásson
    → **sorrend-visszaállító buffer**

SORONKÍVÜLI VÉGREHAJTÁS HOZZÁVALÓI

- Több utasítást is le kell hívnunk, melyek közül válogatunk
  - Ezeket tárolni kell valahol:
    → **utasítástároló**
- Egy eljárás, ami eldönti az utasítások végrehajtási sorrendjét
    → **dinamikus ütemező**
- Egy jó ötlet, amivel az utasítások közötti függőségek csökkenthetők
    → **regiszterátnevezés**
- Egy trükk, hogy a külvilág sorrendi végrehajtást lásson
    → **sorrend-visszaállító buffer**

SORREND-VISSZAÁLLÍTÓ BUFFER

- Soron kívüli végrehajtás nemkívánatos mellékhatása:
  - Regiszterek/memória nem a program sorrendjében változnak
  - A memóriát több szerepélő is használhatja
    - Más processzorok,
    - perifériák,
    - stb.
  - Nem biztos, hogy erre fel vannak készülve!
- Megoldás: sorrend-visszaállító buffer (ROB)
  - Utasítás belépésekor kap egy bejegyzést
  - Ha a végrehajtás kész, beíródik, hogy
    - Mi az eredménye
    - Melyik regiszterbe / memóriacímre szánja az eredményt
  - Akkor érvényesülnek a változások, ha minden korábbi utasításé már érvényesült!

PÉLDA

Utasítás:	Kész?	Eredm.	Hova?
U25 ← MEM[T3]			U25

PÉLDA

Utasítás:	Kész?	Eredm.	Hova?
U25 ← MEM[T3]			U25
U26 ← U25 * U9			U26

PÉLDA

Utasítás:	Kész?	Eredm.	Hova?
U25 ← MEM[T3]	✔	<érték>	U25
U26 ← U25 * U9			U26
MEM[T46] ← U26			MEM[T46]

PÉLDA

Utasítás:	Kész?	Eredm.	Hova?
U25 ← MEM[T3]	✔	<érték>	U25
U26 ← U25 * U9			U26
MEM[T46] ← U26			MEM[T46]
T47 ← T3 + 8			T47

PÉLDA

Utasítás:	Kész?	Eredm.	Hova?
U25 ← MEM[T3]	✔	<érték>	U25
U26 ← U25 * U9			U26
MEM[T46] ← U26			MEM[T46]
T47 ← T3 + 8			T47
T48 ← T46 + 8			T48

PÉLDA

Utasítás:	Kész?	Eredm.	Hova?
U25 ← MEM[T3]	✔	<érték>	U25
U26 ← U25 * U9			U26
MEM[T46] ← U26			MEM[T46]
T47 ← T3 + 8	✔	<érték>	T47
T48 ← T46 + 8			T48
U27 ← MEM[T47]			U27

Kiszámolta a T47 értékét, de nem írhatja be a regiszter tárolóba.
Akinek kell, az innen kiveheti.

PÉLDA

Utasítás:	Kész?	Eredm.	Hova?
U25 ← MEM[T3]	✔	<érték>	U25
U26 ← U25 * U9			U26
MEM[T46] ← U26			MEM[T46]
T47 ← T3 + 8	✔	<érték>	T47
T48 ← T46 + 8	✔	<érték>	T48
U27 ← MEM[T47]			U27
U28 ← U27 * U9			U28

PÉLDA

Utasítás:	Kész?	Eredm.	Hova?
U25 ← MEM[T3]	✔	<érték>	U25
U26 ← U25 * U9	✔	<érték>	U26
MEM[T46] ← U26			MEM[T46]
T47 ← T3 + 8	✔	<érték>	T47
T48 ← T46 + 8	✔	<érték>	T48
U27 ← MEM[T47]	✔	<érték>	U27
U28 ← U27 * U9			U28
MEM[T48] ← U28			MEM[T48]

PÉLDA

Utasítás:	Kész?	Eredm.	Hova?
U25 ← MEM[T3]	✔	<érték>	U25
U26 ← U25 * U9	✔	<érték>	U26
MEM[T46] ← U26	✔	<érték>	MEM[T46]
T47 ← T3 + 8	✔	<érték>	T47
T48 ← T46 + 8	✔	<érték>	T48
U27 ← MEM[T47]	✔	<érték>	U27
U28 ← U27 * U9			U28
MEM[T48] ← U28			MEM[T48]
T49 ← T47 + 8			T49

Az alapelveken túl...

AZ ALAPELVEKEN TÚL...

- Mit tegyünk elágazás esetén?
  - Állítsuk meg az utasítások lehívását?
  - Nem! Inkább találjuk ki, merre folytatódik a program!
- RAW egymásrahatások nem csak a regiszterek között lehetnek!
  - Vannak memória egymásrahatások is
  - Ezek is kezelést igényelnek

SPEKULATÍV VÉGREHAJTÁS

- Utasítás csere-berének korlátot szabnak az elágazások
  - Két feltételes ugrás közé eső utasítások: ***basic block***
- Alapeset:
  - Utasítás csere-bere csak basic blokkon belül
  - ...hiszen ki tudja előre, hol folytatódik a program?
- ***Spekulatív végrehajtás:***
  - Megtippeljük a feltételes elágazások kimenetelét
    → elágazásbecslés
  - A tippelt ágról is töltünk be utasításokat
    ...és csere-beréljük őket basic blokkok *között* is!
  - Sokkal hatékonyabb
    - Mert általában sok a feltételes ugrás
  - Többlet adminisztációt igényel:
    - Spekulatíve betöltött utasítások kapnak egy flag-et a ROB-ban
    - Ha nem jön be a spekuláció (rossz elágazásbecslés):
      - A ROB-ból minden tévedésből betöltött utasítást törölni kell

MEMÓRIA RAW EGYMÁSRAHATÁS

- Milyen is a „sima” RAW egymásrahatás?

```
i1: R1 ← R2 / R3
i2: R4 ← R1 + R5
```

- Nincs mit tenni, i2 meg kell, hogy várja i1-et
- Scoreboard és Tomasulo sem tudott jobbat
- Könnyen detektálható, kezelhető

- Milyen is a memória RAW egymásrahatás?

```
i1: R2 ← R1 + 4
i2: D0 ← D1 / D2
i3: MEM[R1+8] ← D0
i4: D3 ← MEM[R2+4]
```

- i3 ↔ i4 ilyen!

MEMÓRIA RAW EGYMÁSRAHATÁS

- Milyen is a memória RAW egymásrahatás?

```
i1: R2 ← R1 + 4
i2: D0 ← D1 / D2
i3: MEM[R1+8] ← D0
i4: D3 ← MEM[R2+4]
```

- Nem lehet előre detektálni!
- Egészen i3 és i4 cím számításáig nem derül ki, hogy RAW áll fent
- Miért számít?
  - RAW egymásrahatás nélkül i3 és i4 felcserélhető
    - Jó is lenne felcserélni, mert D0 csak sokára lesz kész
    - Addig is haladnánk, ha közben i4-et feldolgoznánk
  - RAW egymásrahatás esetén nem cserélhetők fel!

MEMÓRIA RAW EGYMÁSRAHATÁS

- Mit tehetünk, ha Load-ba futunk? Végrehajtsuk?
  - ROB-ban nincs előtte Store → mehet
  - ROB-ban van előtte befejezett Store
    - más címre → mehet
    - ugyanarra a címre → mehet, de nem a memóriából olvasunk, hanem a ROB-ból
  - ROB-ban van előtte befejezetlen Store, cím kiszámolva
    - más címre → mehet
    - ugyanarra a címre → meg kell várni az érték megjelenését
  - ROB-ban van előtte befejezetlen Store, cím sincs kiszámolva
    - Bátrak vagy gyávák legyünk?
    - Várjunk, attól félve, hátha RAW lesz?
    - Haladjunk, arra számítva, hogy úgysem lesz RAW?

MEMÓRIA EGYÉRTELMŰSÍTÉS

- Belefutunk egy Load-ba
- Feldolgozás alatt van a ROB-ban előtte egy Store
- Végre merjük-e hajtani a Load-ot?

  → ***Memória egyértelműsítés (memory disambiguation)***
- Nagyon gyakori!
  - Utasítások közel harmada memóriaművelet
  - Lokalitási elvek miatt gyakran vonatkoznak egyazon címre

MEMÓRIA EGYÉRTELMŰSÍTÉS

- Lehetséges stratégiák:
  - **Konzervatív** (gyáva):
    - Load-ok kötelesek bevárni minden előttük lévő Store-t
  - **Optimista** (vakmerő)
    - Ne vegyünk tudomást a problémáról
    - Gyorsabb, de kiderülhet, hogy rossz volt a döntés (írási és olvasási cím megegyezik)
    - Ilyenkor vissza kell pörgetni az eseményeket:
      - Load utáni utasítások invalidálása és újrakezdése a ROB-ban
  - **Spekulatív** (okostojás)
    - Optimistán kezd
    - Ha egy utasítással egyszer pórul jár, megjegyzi: vele legközelebb konzervatív lesz

ÖSSZEFOGLALÁS

- Sok mindent tanultunk:
  - Soron kívüli végrehajtás alapelve (precedenciagráf, adatáramlásos elv)
  - Eszközei:
    - Utasítástároló
    - Regiszterátnevezés
    - Sorrend-visszaállító buffer
  - Fejlett technikák:
    - Spekulatív végrehajtás
    - Memória egyértelműsítés

HÁLÓZATI RENDSZEREK
ÉS SZOLGÁLTATÁSOK
TANSZÉK