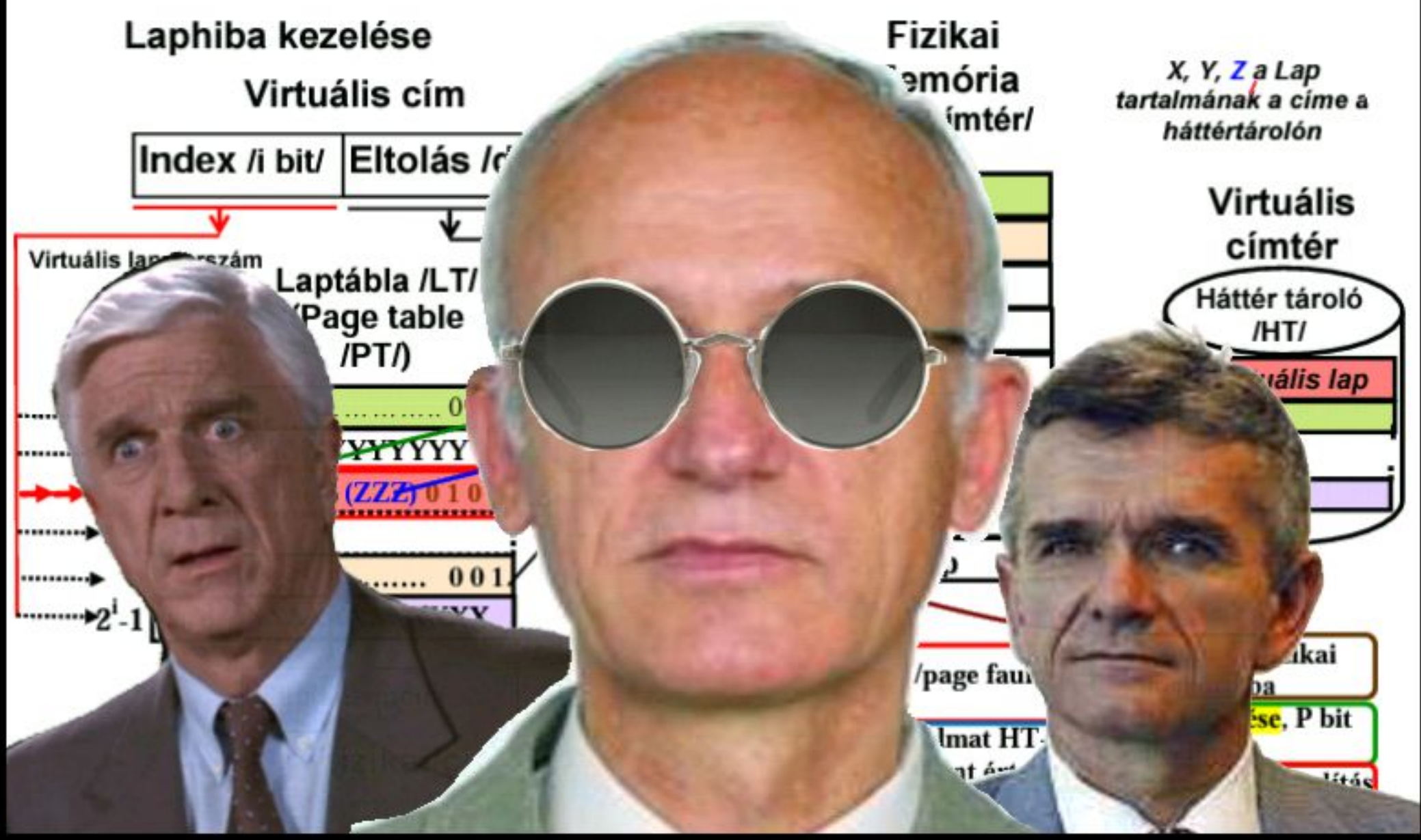
The Ultimate INFO 1 Handbook

I. SZgArch



Veremkeret

pascal-nál, a változók értéke adott, muszáj annyi paraméter amenniy a fgv-ben adott és a hívbott program hívta meg a fgv-t. pascal-nál a változók sorrendje mnwgwgyeztik a fgv hívásnál deklarált változók sorrendjével, a C-nél foedítva van mert ott kevesebbel is meghívhatjuk. A hívóprogram tudja, hogy mennyivel hívja meg. sorrend: SP a fgv meghívása előtt - fgvparaméterek - regiszterek, azon belül ha van távoli eljárás akkor CS utána IP és utána BP. és egyéb buziságok. Utána a lokális változók, és a lokális változók alatti értékre bővül a SP végrehajtá alatt.

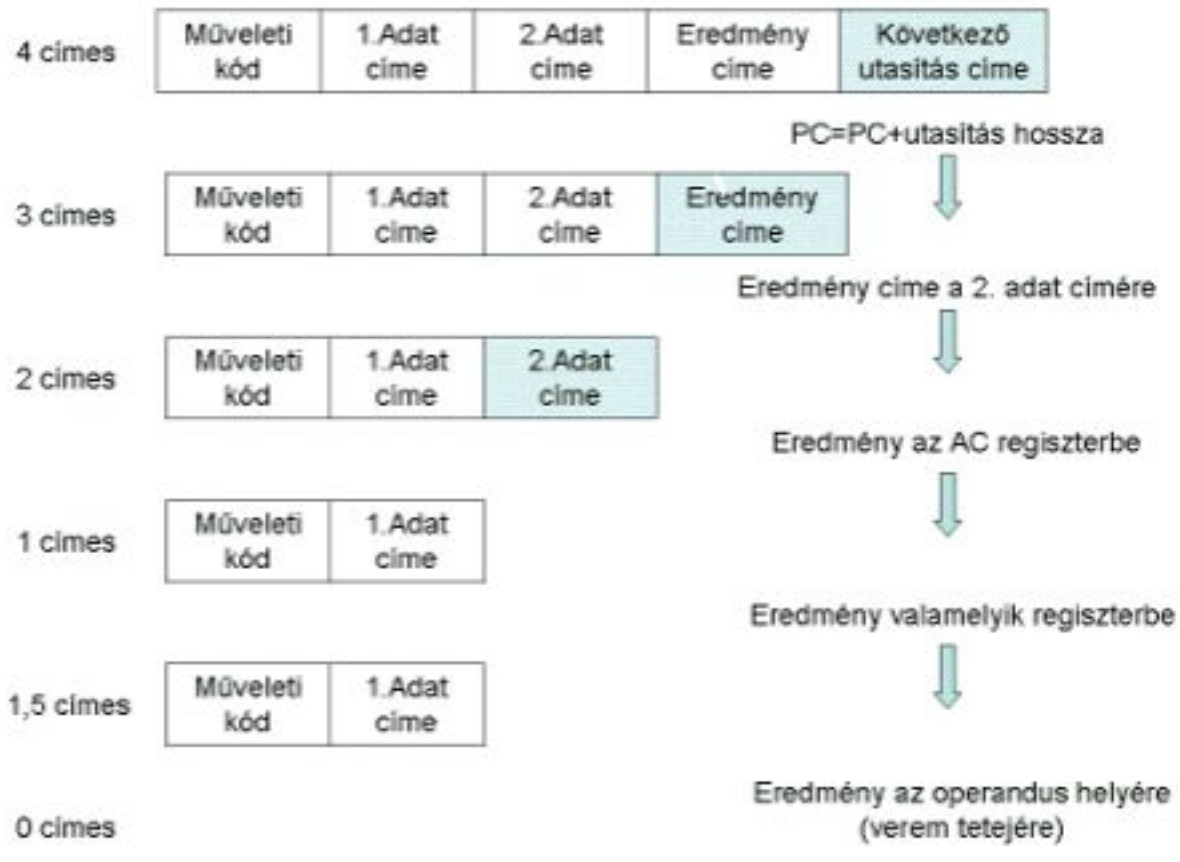
a címzés mindig báziscímzés, azért jó, mert elég a BP-t tárolni, és csak hozzá kell adni az offszetet. Relkoálhatóság. a RET értéke a változók száma pascal-nál, C-nél csak simán RET.

Pipeline

szinkron ütemezés, ott a leghosszabb utasítás lesz a szinkron idő.

$$Speedup = \frac{T_{pl\ nélkül}}{T_{pl} + \text{átlapolási idő}}$$

Címzési módok:

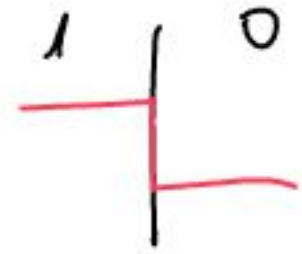


Az utasításkészlet két fő típusa		
CISC	⟷	RISC
sok */100-500/* komplex	- utasítás -	kevés */50-100/* egyszerű
sok */10-20/* bonyolult	- címzési mód -	kevés */2-4/* egyszerű
kevés */8-32/*	regiszter	sok /32-…x100/
regiszterben és memóriában is	- operandus -	regiszterben memória elérésre csak Load/Store
változó-hosszú több órajel idejű	műveleti idő	állandó-rövid egy órajel idejű is

NRZ: Non Return Zero

1-nél felmegy és annyi 1 fent is marad

0, nál egyből lemegy



NRZi: inverse

középen vált 1-nél, 0-ra nem reagál

PE: phase encoding

csak középen vált, szaggatottnál meg mindig változik, kivéve 1→0 és 0→1 átmenetnél, mert ott ingamarad. CSAK ITT VAN Ø középen!!!

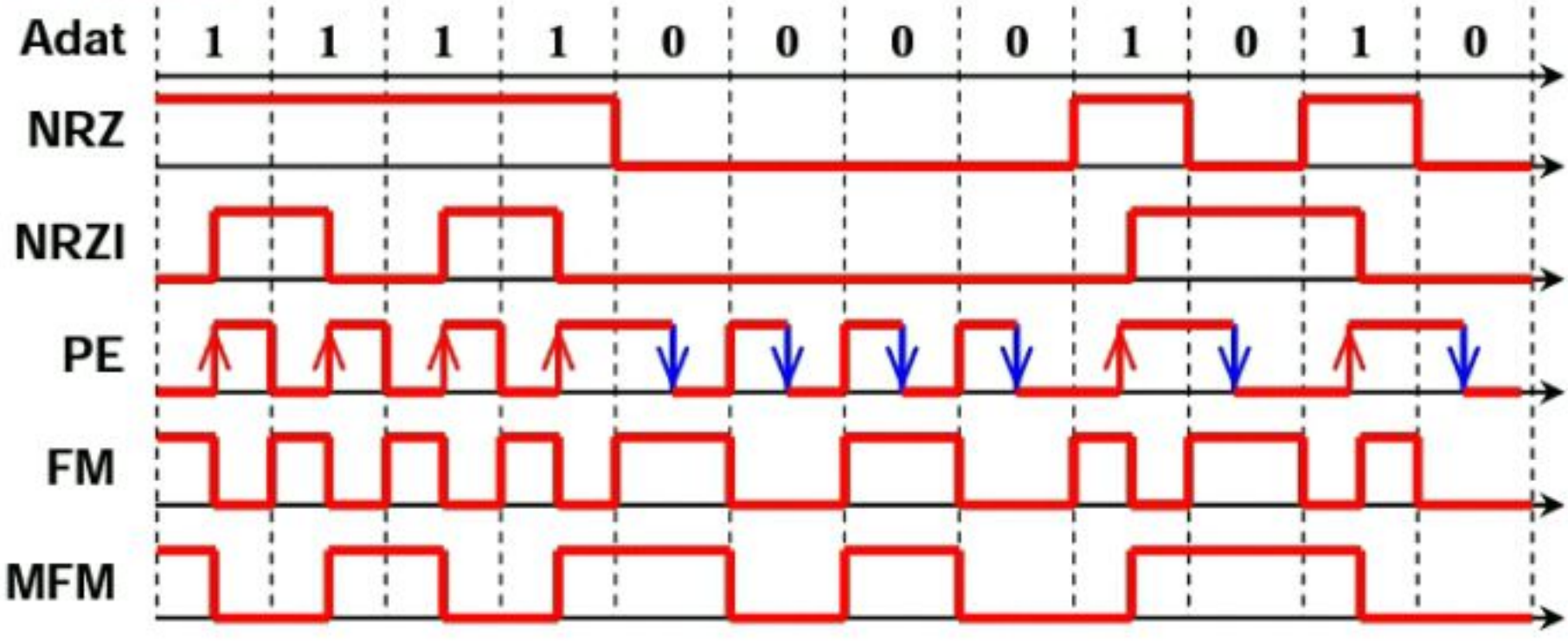
FM: frequence modulation

1-nél mindig vált középen, Ø nál mindig vált szélen

MFM: modified FM

1-nél középen vált 0-nál a periódus végén, kivéve ha 1 jön utána, akkor nem.

❖ *Adatkódolás*



- **NRZ** *Non Return Zero*
- **NRZI** *Non return Zero Inverz* ***/létezik ettől eltérő is, mi ezt használjuk!!/***
- **PE** *Phase Encoding*
- **FM** *Frequency Modulation*
- **MFM** *Modified FM*

Bitsűrűség → fluxusváltozás

Run Length Limited

GCR fajtája az $RLL_{x,y}$

group coded rec.

x: minimum
y: maximum
} egymás után következő 0-k száma

minél sűrűbb a fluxusváltozás, annál nagyobb az adatsűrűség!!!

PIPELINE:

t_L: EMU végrehajtási idő

ut. végrehajtás részei:

UB

UD

UV → EX → EC
EX → MEM
UV → VIR

T_L: lappangási idő

$\sum$ amennyi idő alatt 1 db ut. végrehajtódik

t_T: tárolási idő

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

TP: throughput: időegység alatt bef. ut. száma

$$TP = \frac{1 ut}{t_L}$$

SU: speedup, mennyivel gyorsul, ha PL-t használunk

$$SU_{PL} = \frac{N \cdot ut.szám \cdot t}{N \cdot t_L + (EMV_{S2} - 1) t_L}$$

Nlegyen elég nagy

(kb. 10^5)

$$H_{SU} = \frac{Su}{EMV_{S2}} \leq 1$$

ASAP ütemezés:

$$T_i = T_u \leq \sum_i (t_{EMV_i} + t_{T_i} + t_{vez})$$

Szinkron ütemezés:

$t_{legh.} + t_{áttöltő}$

újraindítási idő: $T_{uj} = t_{L\,max} = t_{CLK}$

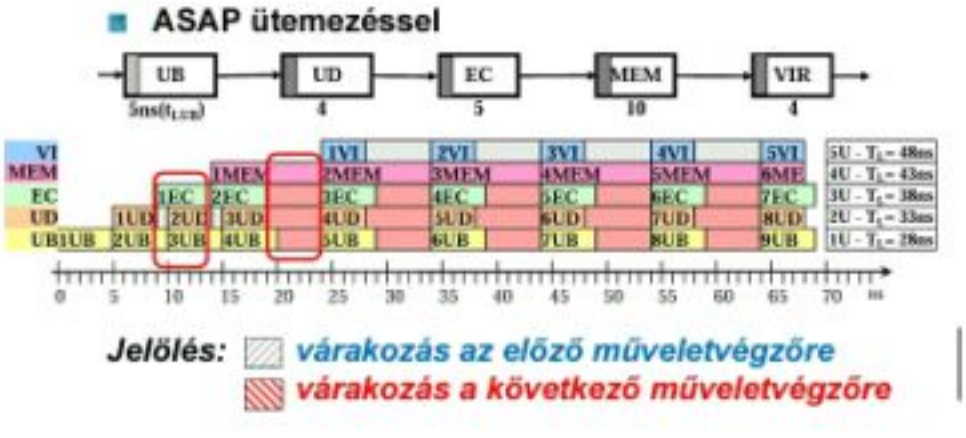
$T_L = T_{ui}$ állandó! $\quad TP = \frac{1ut}{t_{CLK}}$

$$Su_{PL} \leq \frac{(\text{összes művelet áttöltési idővel}) \cdot N}{\text{leglassabb} \cdot N + (1 - EMV_{S2}) \cdot \text{leglassabb}}$$

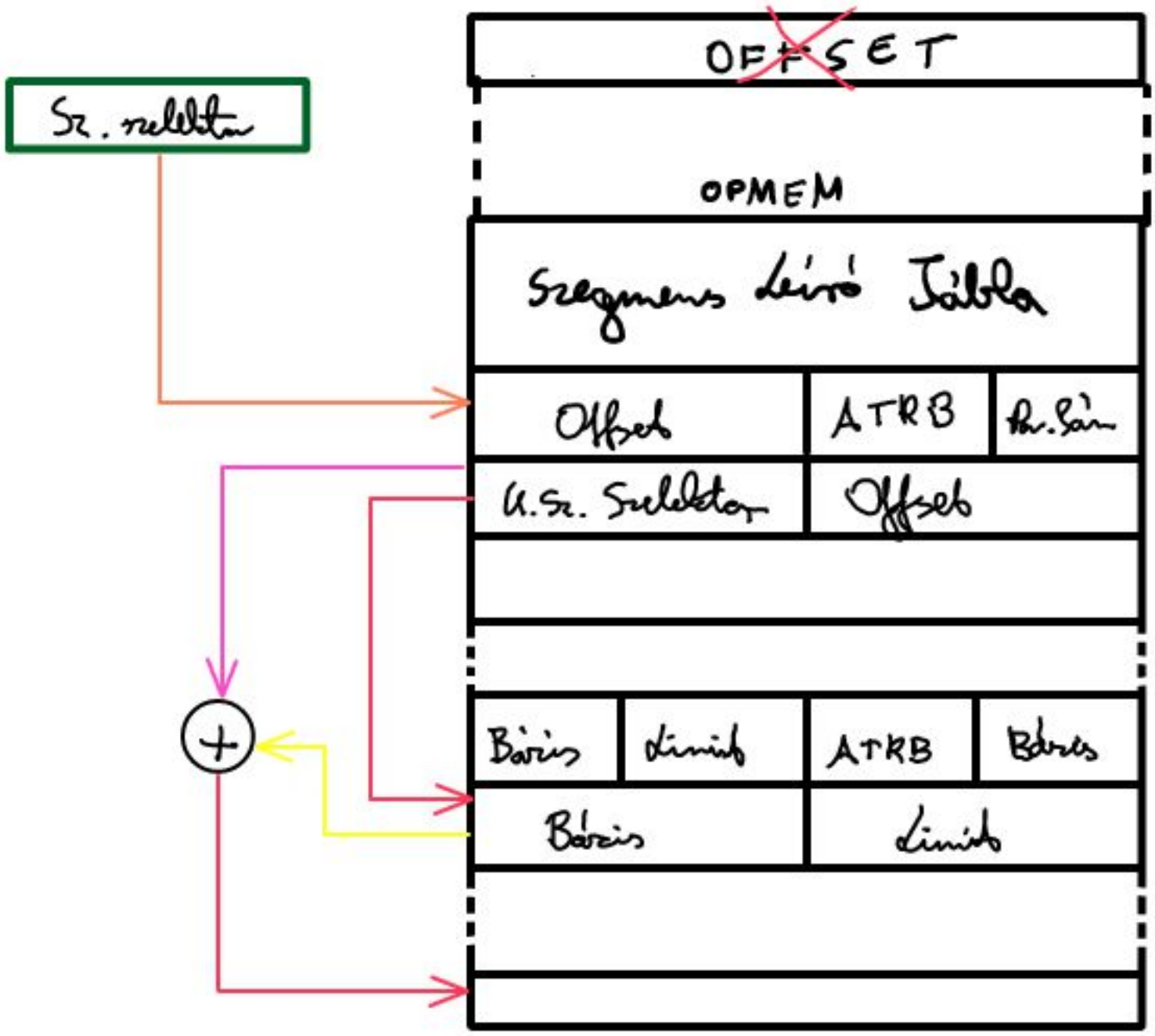
leglassabb műv. áttöltés

Egymásra hatások:

- feldolgozási egymásra hatás: két egymást követő utasítás ugyanazt az erőforrást igényli
- adat egymásrahatás: utasítás egyik operandusa az előző, utasítás végrehajtásának eredménye
- procedurális egymásrahatás: a második utasítás függhet az előző által meghatározott úttól



CALL GATE:


Sz. selektor
OFFSET
OPMEM
Szegmens leíró Tábla
Offset
ATRB
B. Szám
U. Sz. Selektor
Offset
Bázis
Limit
ATRB
Bázis
Bázis
Limit
+

CACHE:

DIREKT (lineáris)	N-UTÁS	TELJESEN ASSZOC.
CBA:	CBA:	CBA:
blokkszám = $2^{(CBA)}$	$\frac{\text{blokkszám}}{n} = 2^{(CBA)}$	NINCS!
COMP:	COMP:	COMP:
1 db	n db	akárhány cache blokk. annyi komp.
SZÓSZÁM	SZ. $\frac{\text{blokkszám}}{n}$	SZ.
blokkszám		blokkszám

MINDEGYIKNÉL:

- blokkméret = $2^{(offset)}$
- OPMEM 3 részre van osztva

- OPMEM ahány bites, annyi az össz. bit
- TAG = OPMEM - CBA - OFFSET

van, hogy a telj cache méretet adott, és egy blokk mérete

> teljes cache = blokkszám · blokkméret

SZÓSZÁM = $2^{(CBA)}$

BITSZÁM = TAG + VEZ. BIT.

MEM. ÁTL. ELÉRÉSI IDEJE:

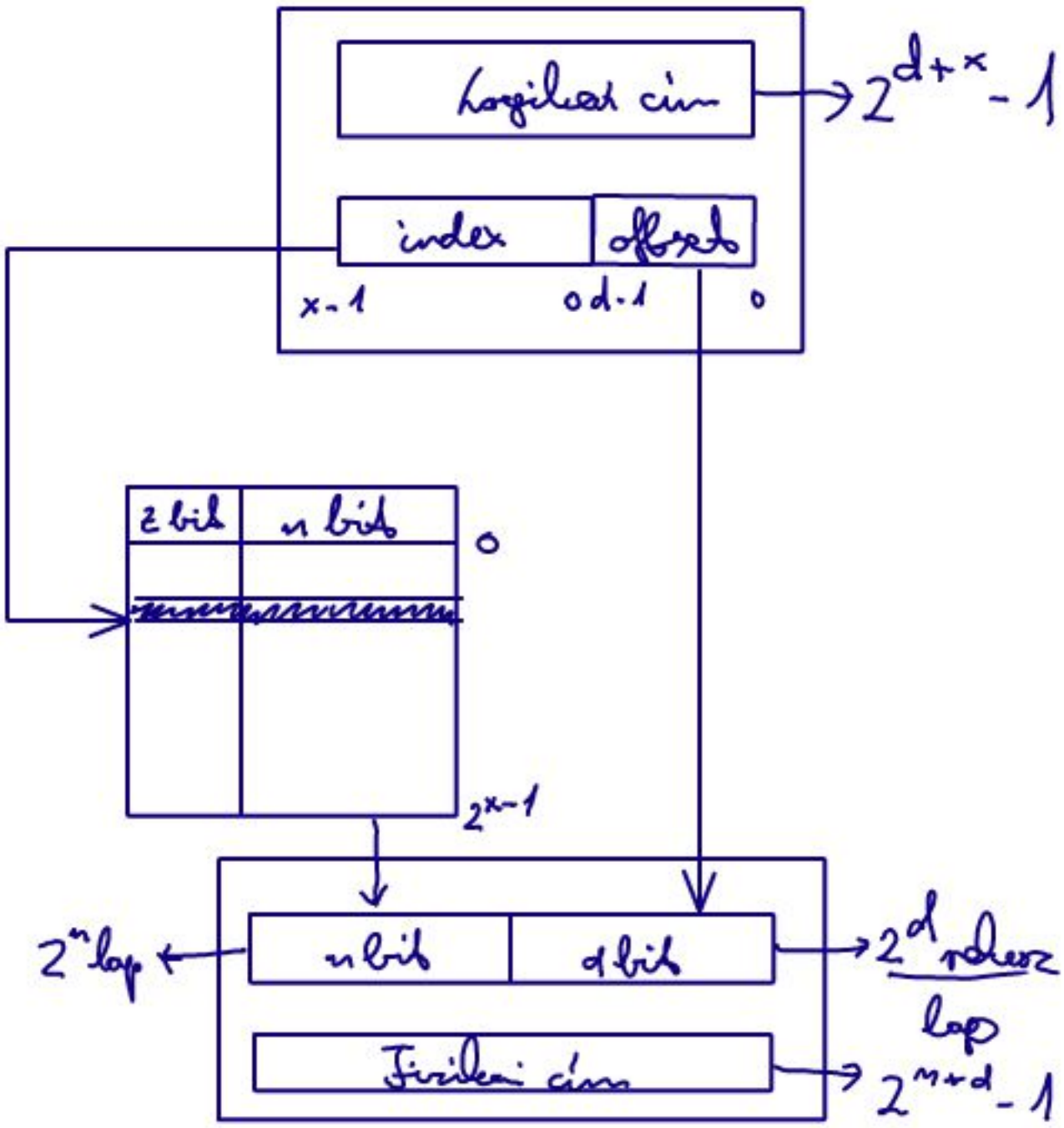
Miss Rate $M_r = 1 - H_r$

$T_t = H_r \cdot T_c + (1 - H_r) T_{OM}$

T_c: cache hozzáf. idő

T_{OM}: qmem hozzáf. idő

INDEX: mekkora lesz a proci által megcímezhető terület



16 bites cím

BCM

1 MByte a teljes → 2^{20} → 20 biten címezhető meg

Indexreg tömb mérete 32 bit → 2^{index}

3 vezérlőbit

offset = logikai cím − index

címtérj = BCM − offset

bitszám = reg. mérete = címtérj + vezérlés

Blokkok (lapok) száma = $\frac{\text{teljes}}{2^{\text{offset}}}$

rószám ≡ indexregisztdue száma

INDEXELT LEK.: 69. feladat

logikai cím: 16 bit

index: 4 bit → 16 indexreg ① SZÓSZÁM: 2^{INDEX}

fiz. mem.: 1MB ⇒ 2^{20} → indexreg. tömb mérete 20 bájt ②

verz.: 2 bit

③ 16-4 = 12 offset bit

⊙ 1MB : 20 verdélből 12 offset

④ 20-12 = 8 a címkéjéig

BITSZÁM ⊙ 1 reg. ⑤ mérete: címkéjéig + vezérlő = 10

⑥ ⊙ reg. tömb mérete = 1 reg mérete x indexreg.-de

⑦ 10 x 16 = 160 bit = 20 Byte

① Feltűnik, hogy a logikai címmel mekkora a teljes megcímezhető tartomány

0000h - FFFFh

② Első számjegy az index (sorszám) mert itt 4 bites

Utolsó 3 számjegy az offset, mert itt 12 bites

③ offset max 4kB

(3.) mert 000 - FFF

↑ 12 biten most megjelölhető = 4 kB

(4.) teljes lapméret 12 kB, tehát 3 lapon fér el!

(5.) Ha a kezdőcím 1000 h, akkor 0h-s sorszám da'ff core

(6.) címtérj 8 bit : 2 hexa érték

sorszám	címtérj.	mennyi:
0h	don't care	nem szóre
1h	80h	1.4kB
2h	81h	2.4kB
3h	FFh	3.4kB

(7.) ha meg van adva szerélőbit, arra oda kell írni elé!

14.) (1.) log. cím : 16 bit

(2.) index : 4 MSB → 16 indexreg. tömb

(3.) oldam : 512 kB → 2^{19} → (4.) teljes mem: $\frac{2^{19}}{2^{12}} = 2^7$

(5.) 16 - 4 = 12 offset

(6.) címtérj. 7 bit

(7.) indexreg. szám : 7+2 = 9

b.) Ha egy 0E80h bázis hosszú programterület a lineáris 8080h címre fordítunk le, hányas indexű reg.-ek tartalmát kell beállítani a futáshoz?

1. 8080h → 12 bites offszetje 080h
2. 1 blokk mérete: 2^{offszet} = FFFh
3. E80h + 080h = F00h ezekkel, mint FFFh, tehát belefér 1 blokkba
4. 1 indexreg.-et kell módosítani annak címe (szám.) 4 MSB
5. annak 8h.

c.) milyen hexa értéket kell abba az indexreg.-be írni ha a fizikai mem.-ba 7A000-7AFFF tartományban íres helyre van és a vez. bit értéke 11?

1. címbitei.: 7 bit:

   111'1010b

2. ehhez jön az 1 vez bit:

   1'111'1010b

   ami 1FAh

Indexelt leképerés, címsín 16 bites, index 3MSB

Opmem 256kB → 2^{19}

↓

8 indexreg ⇒ rögzített

v : 2 bit

offset: 16-3 = 13

E. (só x bit)?

18-13 címbiteg. = 5 bit + vezbit 5+2 = 7

(8 x 7)

1000 0000 1000 0000

index = 4 offset

0001 1001 1000 0000

0666 6650 1000 0000

1 A 6 0

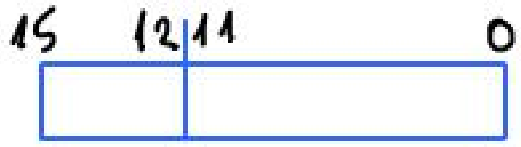
8kB: 0000 - 1FFF > 1A00

belépés „a 4h indexreg-be.

00010 101

1 5 h

Konkrét példa:



1 blokk (lap) = 4 kB

512 kB = 2^{19} ⇒ 19

4 indexxel 16 bitet tudunk kiválasztani

⋮

(b) rész:

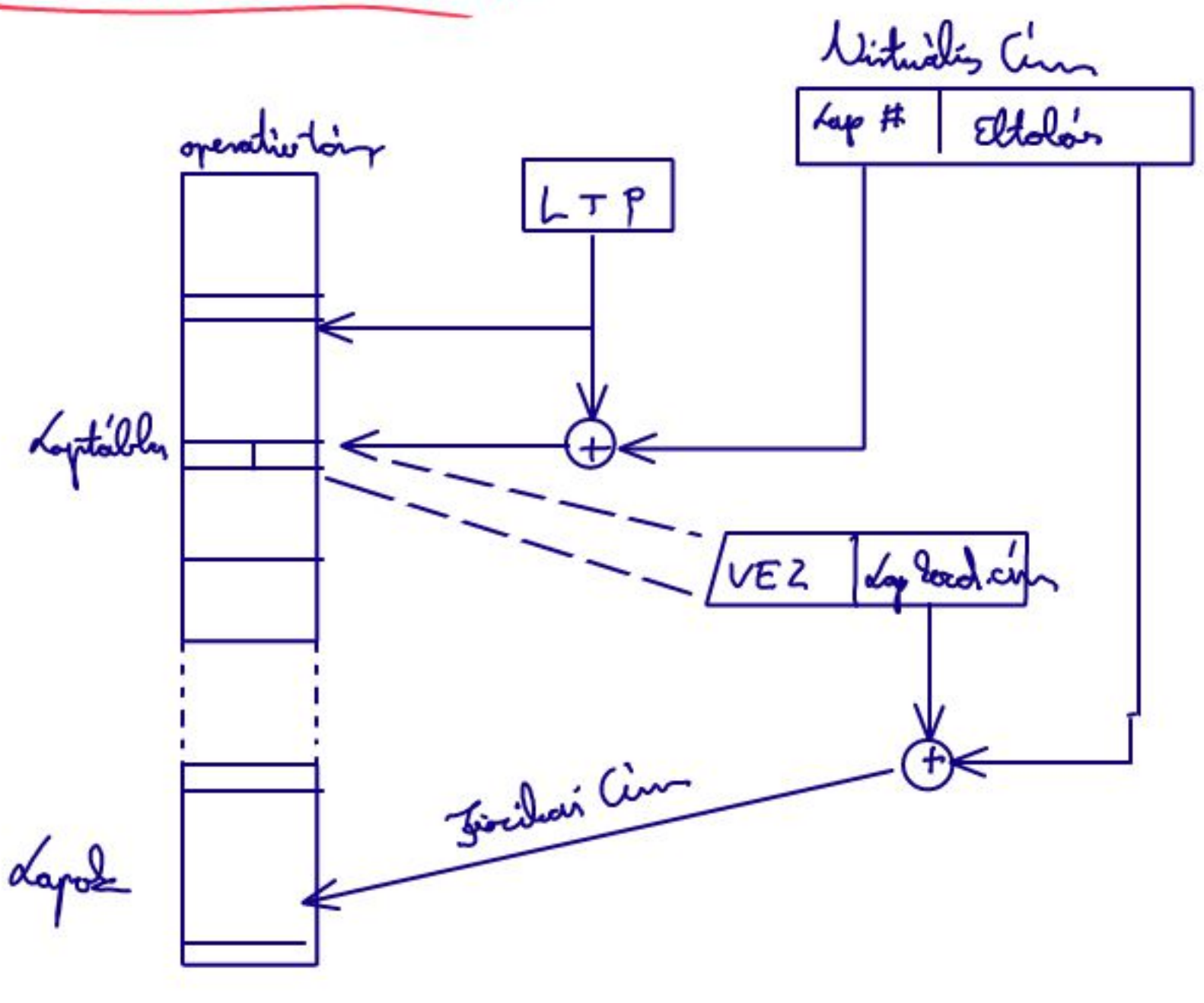
0E80 8080-ról indul

↓

1000 0000 1000 0000

mivel belefér egy kB-ba, ezért egy elég. 8-as index reg-et kell beállítani

LAPSZERVEZÉSŰ V.T.:



OLVASÓ-FEJMOZGÁS:

- 200 sávos lemez
- 143. sáv áll

igények:

88, 147, 91, 177, 94, 150, 102, 175, 130

FCFS (first come first served)

sorjában.

SSTF (legjobb fejmozgás)

SCAN (pásztázó, amíg lehet felfelé, aztán lefelé megy)

C-SCAN (csak felfelé megy, amint elfogy lefelé legalulra)

VME, MULTI BUS II.

⊙ Hány trbiter és hány master lehet VME rendszerben?

- 1 trbiter, ami a hierarchiát eldönti

- Elvileg ∞ Master felfűzhető Daisy-chain-be.

⊙ Daisy-chain esetén busszmegszerz. stratégia?

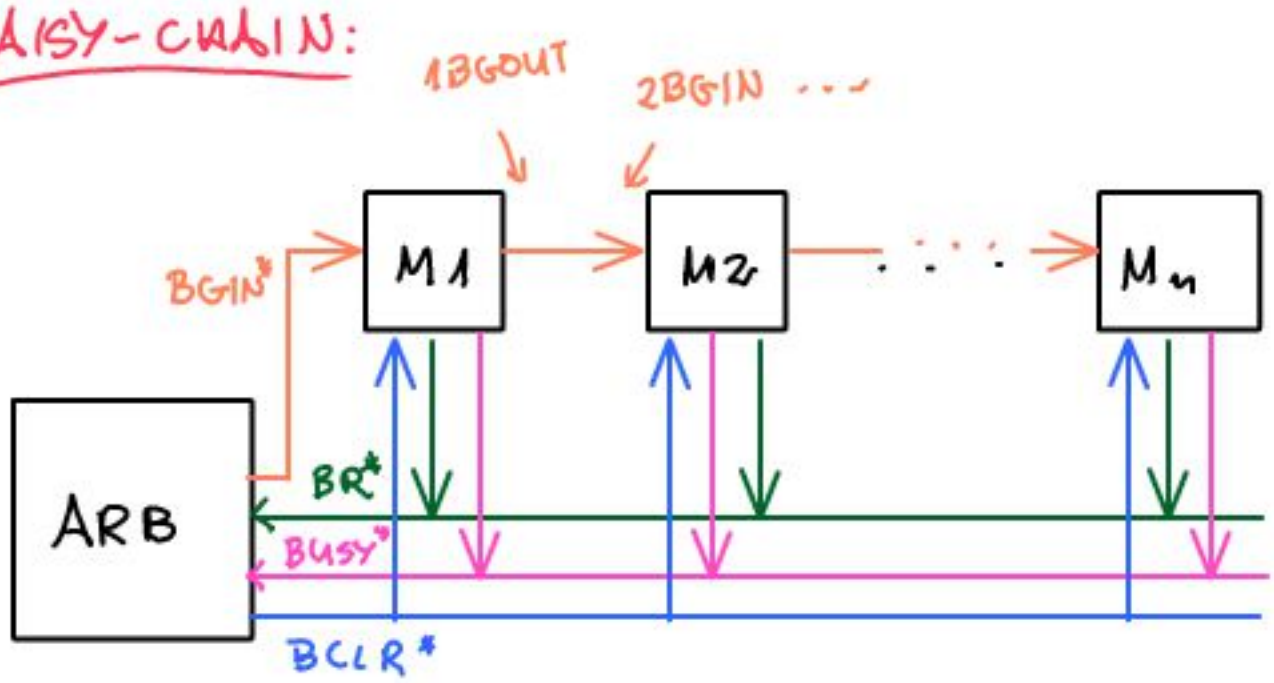
- Prioritásos

⊙ Hány megszakításkérő és megszak. kezelő lehet a rendszerben?

- 8 bites buszcímtér miatt 256 db lét

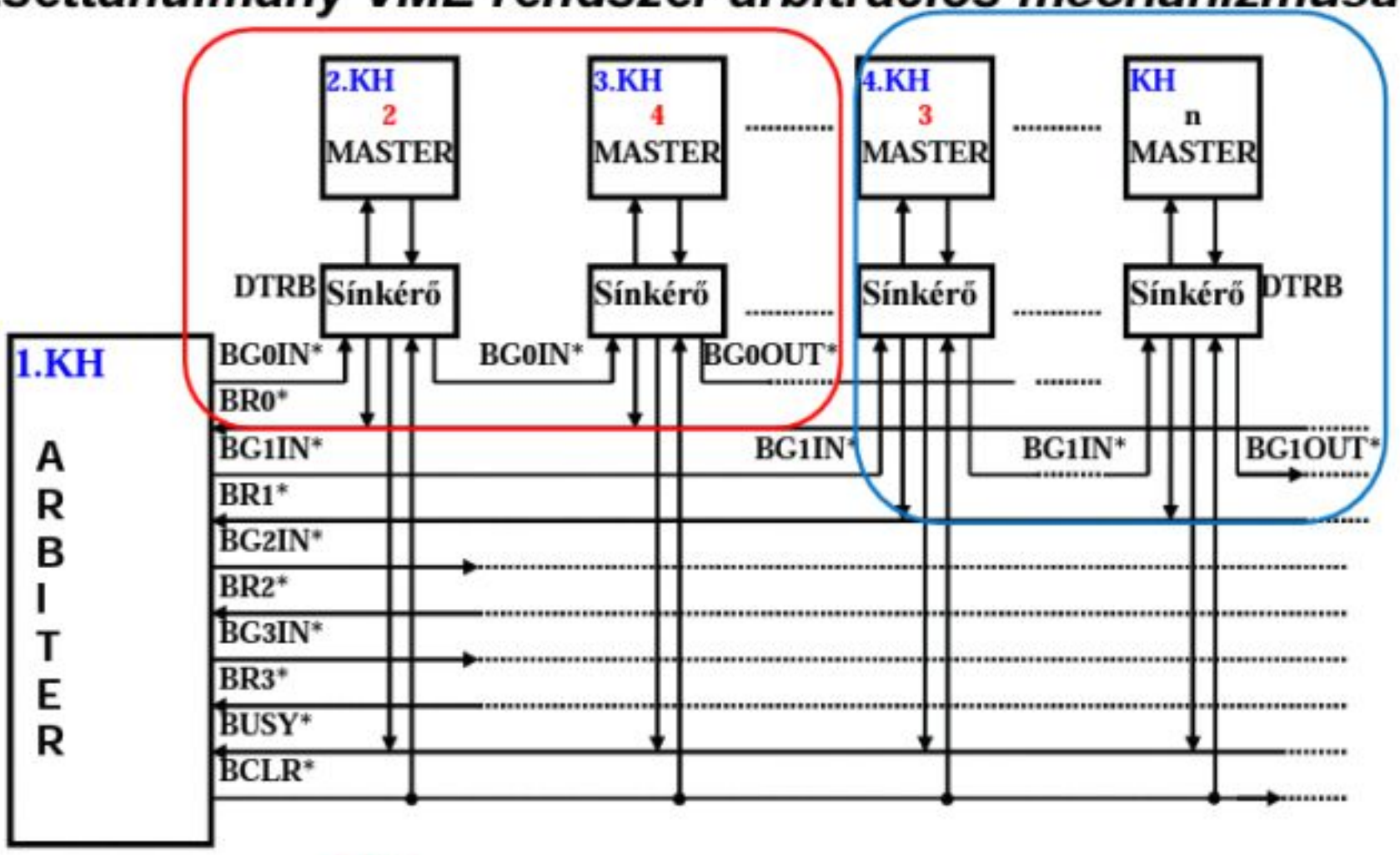
- IRQ 1...7 miatt 7 db kezelő.

DAISY-CHAIN:



- **Kombinált mechanizmus**
  - ***A két módszer kombinációja***

Esettanulmány VME rendszer arbitrációs mechanizmusa

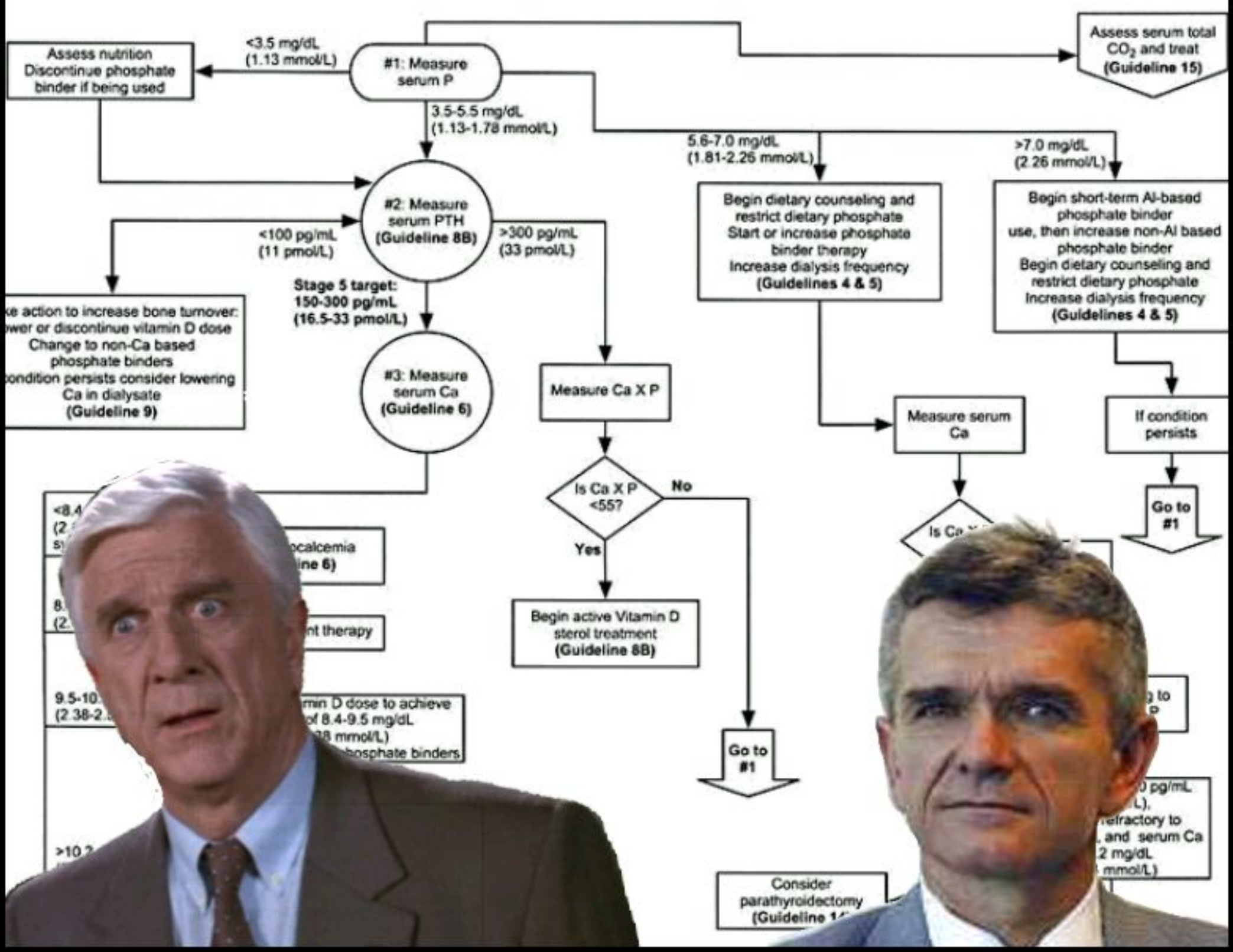


1.

2.

	ARB3*	ARB0*										
A	1	0	1	0	0	0	1	0	1	1	1	1
B	1	0	0	0	1	1	1	0	0	0	1	1
C	1	0	0	0	1	0	1	0	0	0	1	0
	1	0	0	0	0	0	1	0	0	0	1	0

II. Opre


Assess nutrition
Discontinue phosphate binder if being used
<3.5 mg/dL
(1.13 mmol/L)
#1: Measure serum P
Assess serum total CO_2 and treat (Guideline 15)
3.5-5.5 mg/dL
(1.13-1.78 mmol/L)
5.6-7.0 mg/dL
(1.81-2.26 mmol/L)
>7.0 mg/dL
(2.26 mmol/L)
Begin dietary counseling and restrict dietary phosphate
Start or increase phosphate binder therapy
Increase dialysis frequency
(Guidelines 4 & 5)
Begin short-term Al-based phosphate binder
use, then increase non-Al based phosphate binder
Begin dietary counseling and restrict dietary phosphate
Increase dialysis frequency
(Guidelines 4 & 5)
#2: Measure serum PTH (Guideline 8B)
<100 pg/mL
(11 pmol/L)
>300 pg/mL
(33 pmol/L)
Stage 5 target:
150-300 pg/mL
(16.5-33 pmol/L)
action to increase bone turnover:
wer or discontinue vitamin D dose
Change to non-Ca based phosphate binders
condition persists consider lowering Ca in dialysate
(Guideline 9)
#3: Measure serum Ca (Guideline 6)
Measure Ca X P
Is Ca X P <55?
No
Yes
Begin active Vitamin D sterol treatment (Guideline 8B)
Go to #1
Measure serum Ca
If condition persists
Go to #1
Consider parathyroidectomy
(Guideline 14

OPRE:

Peterson-algoritmus:

folyamatok: P_i, P_j

változók: r_i, F_i, r_j, F_j, turn

közös változó aggregálja folyamatra mutat

BELÉPÉS:

write (F_i, 1)

write (turn, j)

n × { read (F_j, r_j)

r_j == 0 → belép, KS

m × { read (turn, r_j)

r_j == i → belép, KS }

KILÉPÉS:

write (F_i, 0)

Batcher-algoritmus

⊙ n folyamatra kölcsönös kizárás

folyamatokhoz tartozik 1 szám

ha ∅ nem kilép belépni

ha nem ∅, az a sorszámos

⊙ jelző bit, amíg 1-es, addig sorszámok válaszhatók magának, más nem választhatók. Ha kiválasztották, akkor visszaállítja ∅-ba.

Testandset: record (F, r)

write (r, 'foglalt')

Exchange: GF ⟶ TMP

LF ⟶ GF

TMP ⟶ LF

while TestAndSet(F) == 'foglalt' do skip

KS

F = 'szabad'

Exchange.el

LF = 'foglalt'

n [XCHG(GF, LF)
    LF == 'szabad' → KS

Szemafor:

DEF:

var s: integer = k;

P(s):
    while s ≤ 0 do skip;
    s = s - 1;

V(s): s = s + 1;

Fajtái:

Precedencia

sorendben egymás után következhetnek s=0

P1	P2
S1	P(s)
V(s)	S2

Rendezési

kölcsönös egymásra s1 = s2 = 0

P1	P2
V(s1)	V(s2)
||	||
P(s2)	P(s1)

Kölcsönös kizárás

nem lehet átlapolódás

s = 1

P1	P2
P(s)	P(s)
KS	KS
V(s)	V(s)

Üzenet alapú együttműködés:

itt a folyamatoknak nincs KM-ja, hanem egy adatátviteli hálózaton keresztül kommunikálnak

adatcseréhez 2 művelet: send
receive

fajtái: - direkt címzés
- indirekt címzés
- aszimmetrikus címzés
- csoportcímzés

Szemantika:

send tényleg elküldte az adatot?

megold: nyugtázás

üzenetből ne legyen több példány:

mo: sorrendiség vagy pufferelés

véges - betelhet

végtelen - lehet adatvesztés

Megbízhatóság: timeout

nyugtázás

Ha a nyugtázás alapú el: üzenet szimulálása

Üzenet rongálódik, pl. EM zavar → hibaelkenző (redundáns) kódolás

↓

kettős boríték módszere

Holtpont:

Mind KM-nél, mind ÜA-nél ugyanaz játszódik le, csak más a neve

KM:	ÜA:
- RQST(E)	- SEND(P)
- RLSE(E)	- RECEIVE(P)

⊙ Holtpont az, hogy kialakul egy olyan helyzet, amikor a folyamatok egymásra várnak. Erre algoritmusok:

Strucc: szaladunk bele a holtpontba, úgy. ritkán történik meg.

HOLT PONT FELADAT:

eddig adjuk folyamatokat nem engedtük el:

P1	P2	P3
RQST(R3)	RQST(R2)	RQST(R3)
RQST(R2)	RLSE(R2)	RQST(R2)
RQST(R1)	RQST(R1)	RLSE(R2)
	RQST(R2)	RQST(R1)
RLSE	RLSE	RLSE
3-2-1	1-2	3-2, 3-1

HOLTPONT!

Megelőzés: tervezésnél struktúrálisan holtpontmentes rendszerek tervezünk

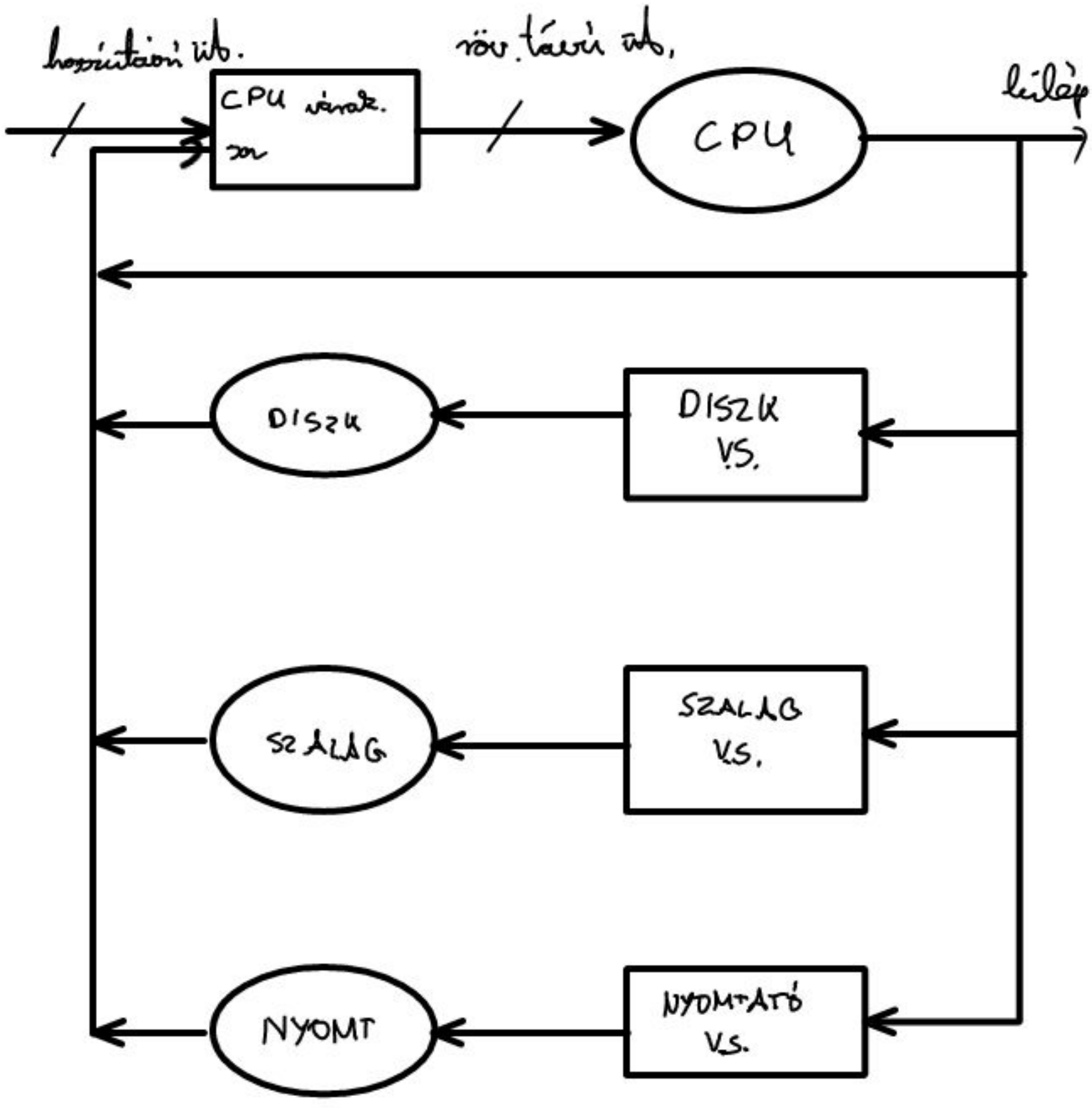
erőforrások csoportosítása:

PÉLDÁNY SZ.:
- egypéldányos
- többpéldányos

ELMENTHETŐSÉG SZ.:
- proc. állapota menthető 1 ut. lépésből
- mem. állapota menthető háttértárra
- periféria nem menthető

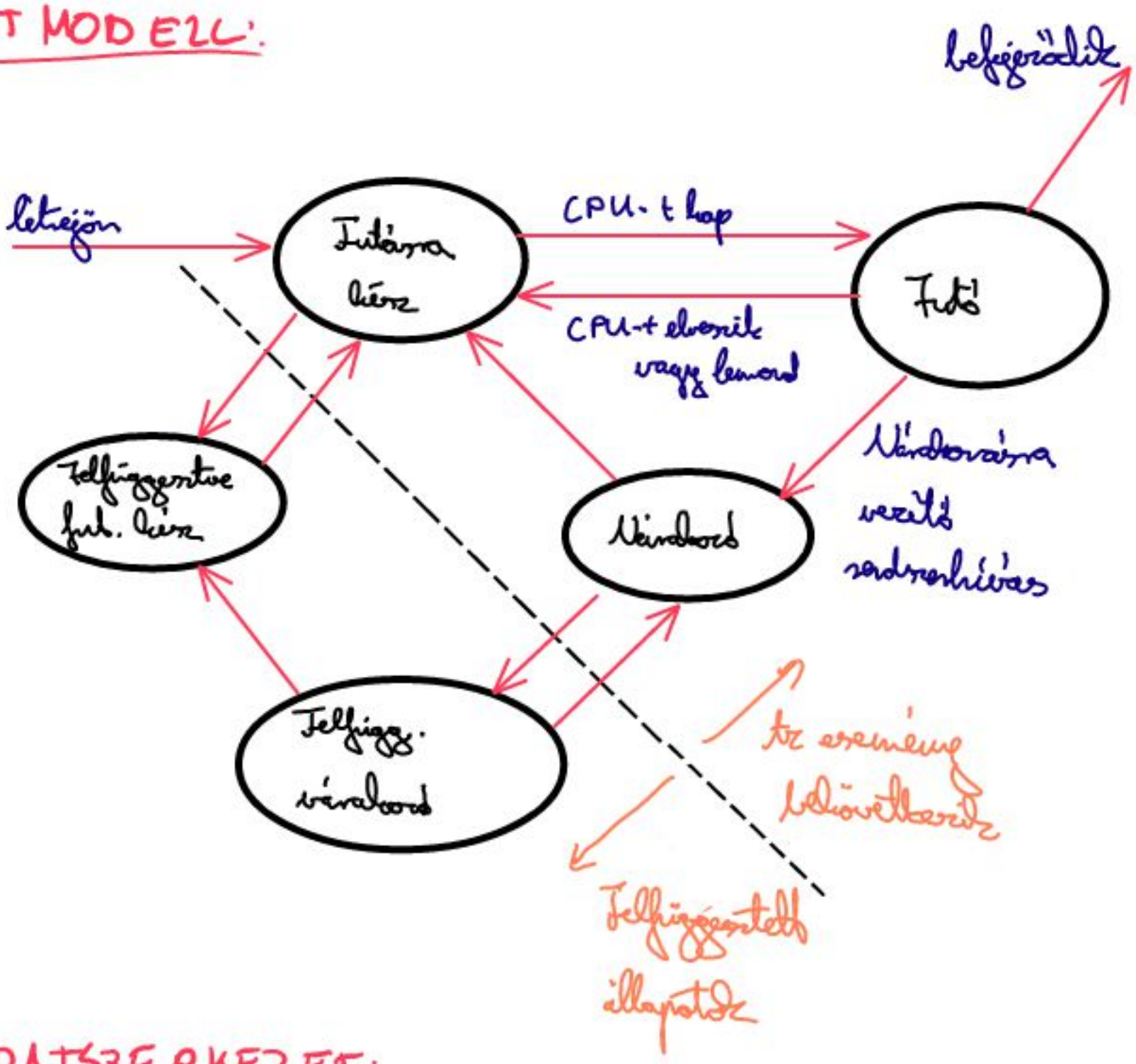
holtpont kialakulás feltételei:
- kölcsönös kizárás
- hold-and-wait
- no preemption
- circular wait

SORBANÁLLÁSI MODELL:

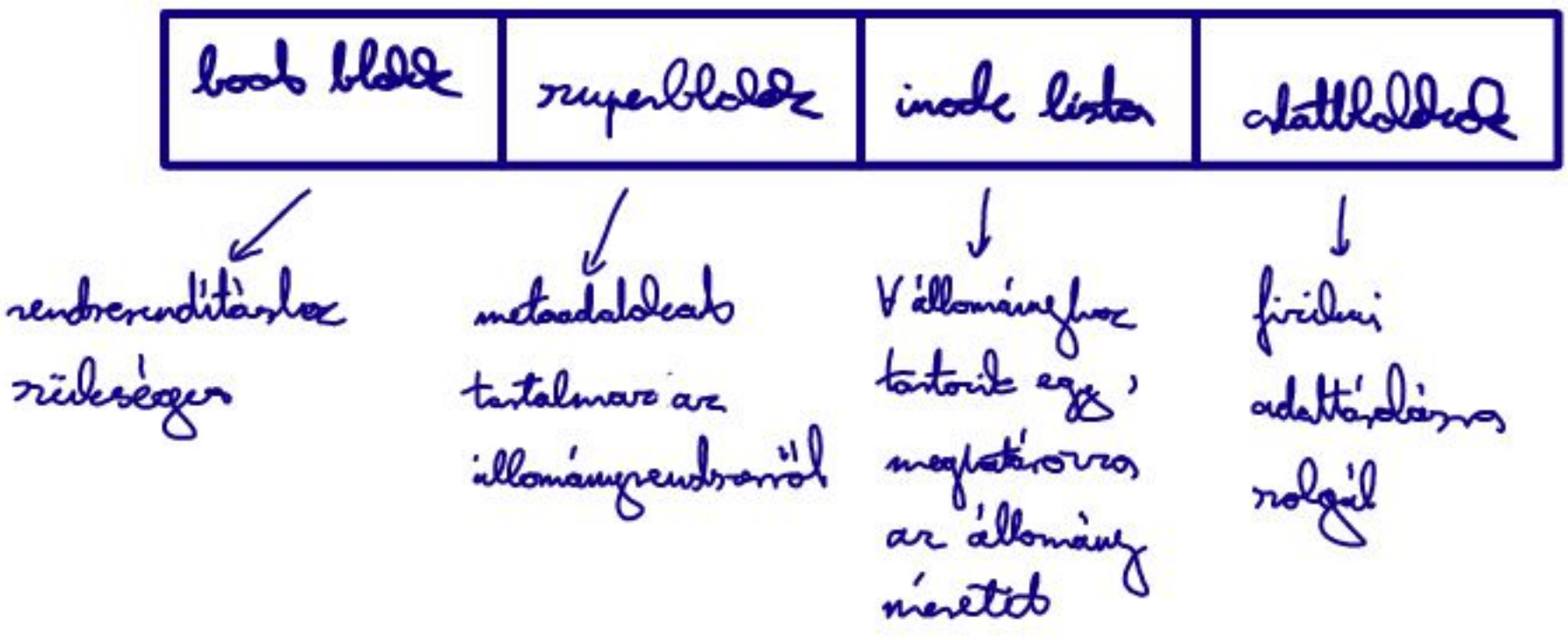


- nincs benne memória
- ki kerül be, hogy erőforrást
- meghatározható vele a multiprogramozás foka.

ÁLLAPOTMODELL:



S5FS ADATSZERKEZET:



EFFEKTÍV HOZZÁFÉRÉSI IDŐ

LAPHIBA:

t_m ← memória

T_{eff} ← effektív

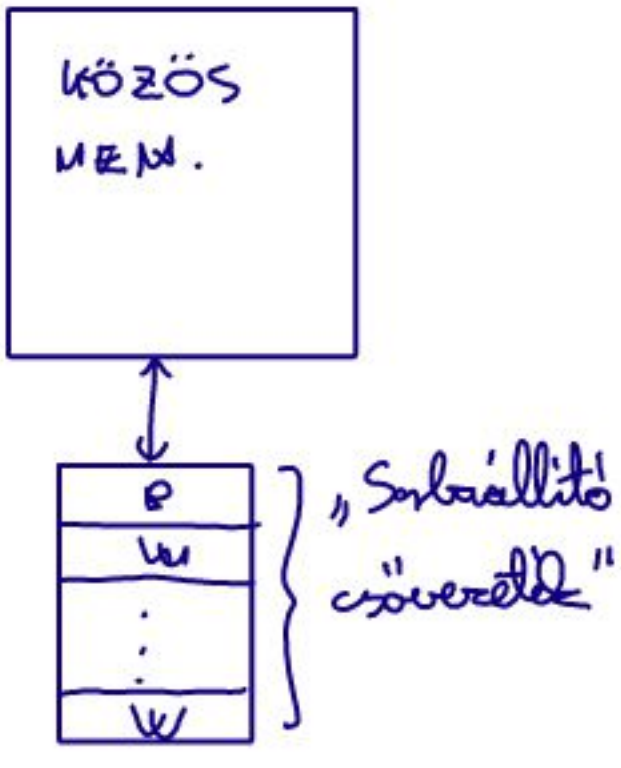
$q_h = \frac{1}{\frac{T_e}{t_m}} = \frac{t_m}{T_e}$ ← lemez

elhanyagolható

$$T_{eff} = q T_e + (1-q) t_m = \frac{t_m}{T_e} \cdot T_e + t_m - \left(\frac{t_m}{T_e} t_m\right) \sim 2t_m$$

KÖZÖS MEMÓRIÁS EGYÜTTMŰKÖDÉS

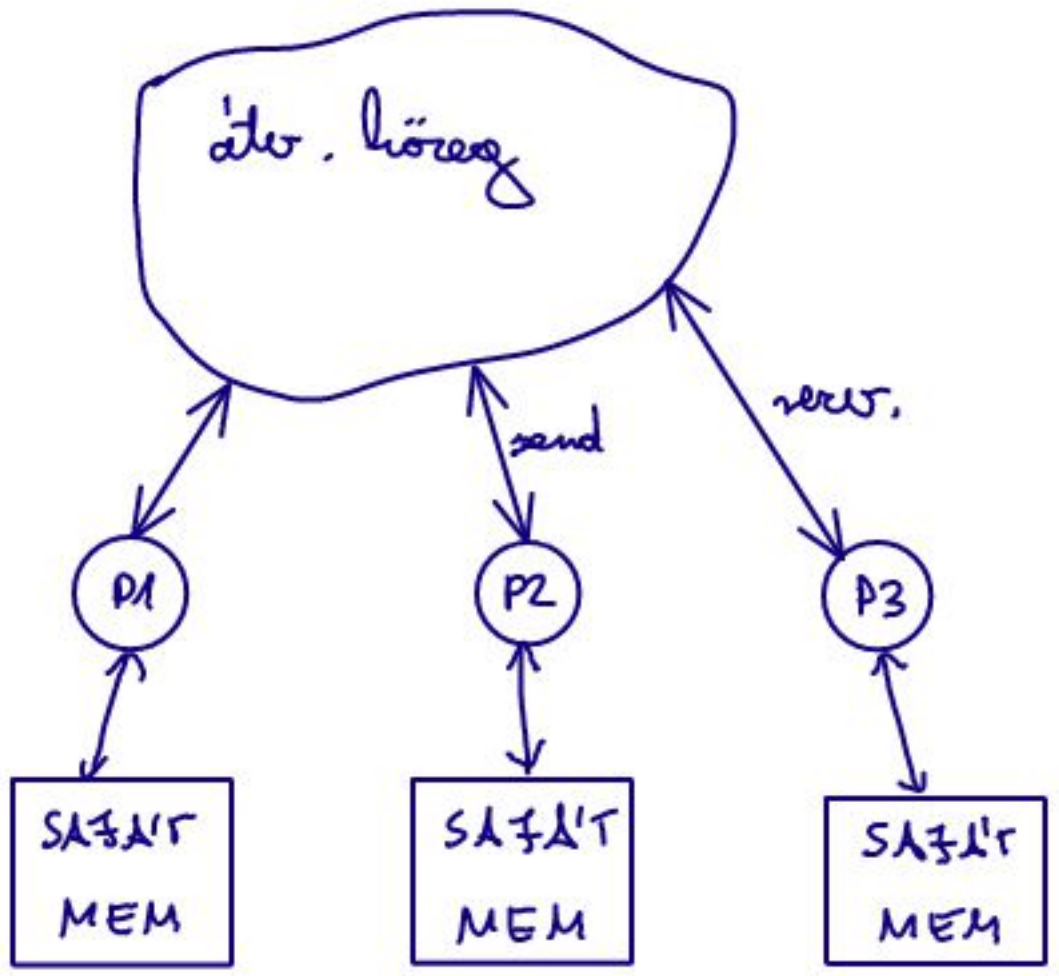
- Egyszerre történő R-W, W-W, W-R, R-R ütközésekből kiküszöbölni
- Egy kapcsolótáblával valósítható meg, sorrendbe állítja az érkező R, W igényeket, eggyenként hajtja végre.
  - ↳ R és W atomi műveletek, NEM interferálhatnak



ÜZENETVÁLTÁSOS E.M.:

⊙ Nincs ÚM, kis adatszerkezettel „üzenetekkel"
kommunikálnak egy közegen keresztül.

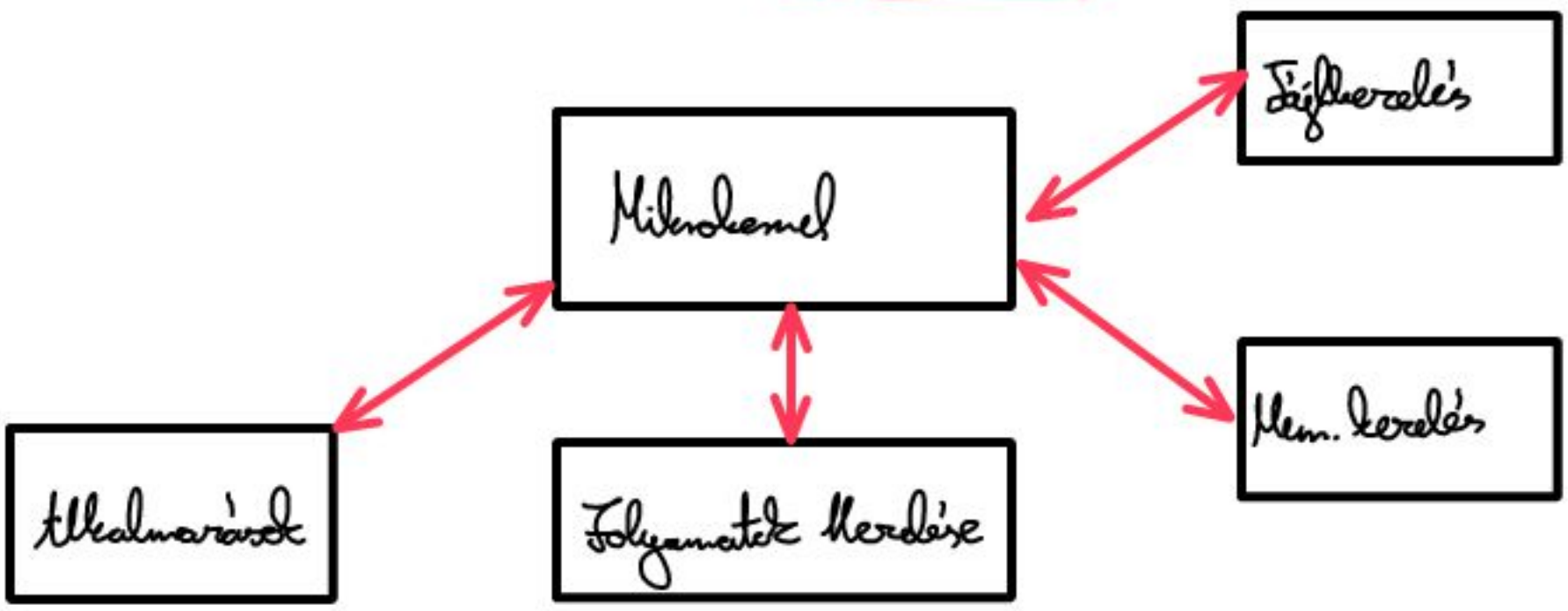
Utasítások: Send(cím, folyamat)
Receive(cím, folyamat)



OPRENDSZER LEMEZKEZELŐ FUNKCIÓI:

- Szektorok címzése
- Járműgondolkodás
- Fájlrendszer alkalmazása
- Tegmozgásütemezés
  - FCFS
  - SSTF
  - SCAN
  - N-SCAN
  - C-SCAN

KLIENS-SZERVER FELÉPÍTÉS:



A kapcsolódó egységek változatos lehetnek kliens-szerver szerepűek

Banker - algoritmus

Erőforrások: E(10,5,7)

Folyamatok: P1, P2, P3, P4, P5

	Max igény			Foglal			Igény			
	E1	E2	E3	E1	E2	E3	E1	E2	E3	
P1	7	5	3	0	1	0	7	4	3	
P2	3	2	2	3	0	2	0	2	0	P2: igen
P3	9	0	2	3	0	2	6	0	0	
P4	2	2	2	2	1	1	0	1	1	
P5	6	3	3	0	0	2	4	3	1	P5 igen
				8	2	7				
KÉSZLET:				2 szabad	3	0				

1. Igény = max. igény - foglal.

2. Szabad erőforrások meghatározása (az összes: 10, 5, 7)

3. Készletből kielégíthető-e valamelyik folyamat?

4. Új készlet = Készlet + P2 (foglal) = (5, 3, 2)

5. Megnézzük, hogy az új készlettel kielégíthető e valamelyik folyamat?

⋮

- Egy rendszerben az alábbi erőforrások vannak:

E1: 10 darab E2: 5 darab E3: 7 darab

- A rendszerben 5 folyamat van: P1, P2, P3, P4, P5
- Biztonságos-e holtpontmentesség szempontjából a következő állapot?

	MAX. IGÉNY:			FOGLAL:			IGÉNY:			
	E1	E2	E3	E1	E2	E3				
P1	7	5	3	0	1	0 ✓	7	4	3	✓
P2	3	2	2	3	0	2	0	2	0	✓
P3	9	0	2	3	0	2	6	0	0	✓
P4	2	2	2	2	1	1	0	1	1	✓
P5	4	3	3	0	0	2	4	3	1	✓
FOG:				8	2	7				

SZAB: 2 3 0

5 3 2

7 4 3

7 5 3

HA VÉGÜL MINDEN FOLYAMAT ELFOGY, AKKOR BIZTONSÁGOS ÁLLAPOTBAN VAN A RENDSZER!

CPU ütemezések.

P: preemptív
NP: nem preemptív

FCFS: first come first served NP

érkezési sorrendben szolgálja ki a folyamatokat

SJF: shortest job first NP

ebből után a legrövidebb jön. Ha több ilyen van FCFS-el választ közülük.

RR(x): round robin P

körbemegy a folyamatokon és mindegyikből max. x-et végez el.

SRTF: shortest remaining time first P

mert tudja szakítani (preemptív)

PP: preemptív prioritás P

aminek nagyobb a prioritása, megszakíthatja

NPP: non-preemptív prioritás NP

LAPCSERE STRATÉGIÁK:

FIFO:

(4)

5	4	3	2	4	5	1	7	4	5	4	3	6	7	3	4	5	4	3	7
5	5	5	5	✓	✓	1	1	1	1	✓	3	3	3	✓	3	5	✓	5	✓
	4	4	4			4	7	7	7		7	6	6		6	6		3	
		3	3			3	3	4	4		4	4	7		7	7		7	
			2			2	2	2	5		5	5	5		4	4		4	
						X	X	X	X		X	X	X		X	X	✓	X	

~~5~~ ~~4~~ ~~3~~ ~~2~~ 1 7 4 5 3

10→4

OPT: (4)

								1	1		1		1	1	1	↓			
5	4	3	2	4	5	1	7	4	5	4	3	6	7	3	4	5	4	3	7
5	5	5	5	✓	✓	5	5	✓	✓	✓	✓	6	✓	✓	✓	5	✓	✓	
	4	4	4			4	4					4				4			
		3	3			3	3					3				3			
			2			1	7					7				7			
						X	X					X				X			

4+4

LRU (last recently used)

5	4	3	2	4	5	1	7	4	5	4	3	6	7	3	4	5	4	3	7
5	5	5	5	✓	✓	5	5	✓	✓	✓	5	5	✓	3	✓	3	✓	✓	✓
	4	4	4			4	4				4	4	4		4				
		3	3			1	1				3	6	6		5				
			2			2	7				7	7	7		7				

SC (second chance)

5	4	3	2	4	5	1	7	3	4	5
5+	5+	5+	5+	✓	✓	1+ 5−	1+	1+	1+	1−
	4+	4+	4+			4−	7+	7+	7+	7−
		3+	3+			3−	3−	3+	3−	5+
			2+			2−	2−	2−	4+	5+

8 4 8 2 8 4 3 2 5 7 1 7 3 4 1 7 5

MULTIPROG. RENDSZEREKNÉL PROBLÉMÁK:

- ① ídleapcsoláshoz egyszerre több prog-nak kell a tárban lennie: tárgazdálkodás
- ② egyy időben több futásra kész prog-lehet: CPU ütemezés
- ③ gépi erőforrás-felh.-t koordinálni kell: holtpont-kezelés
- ④ prog-ok ne zavarják egymást: védelmi mechanizmusok