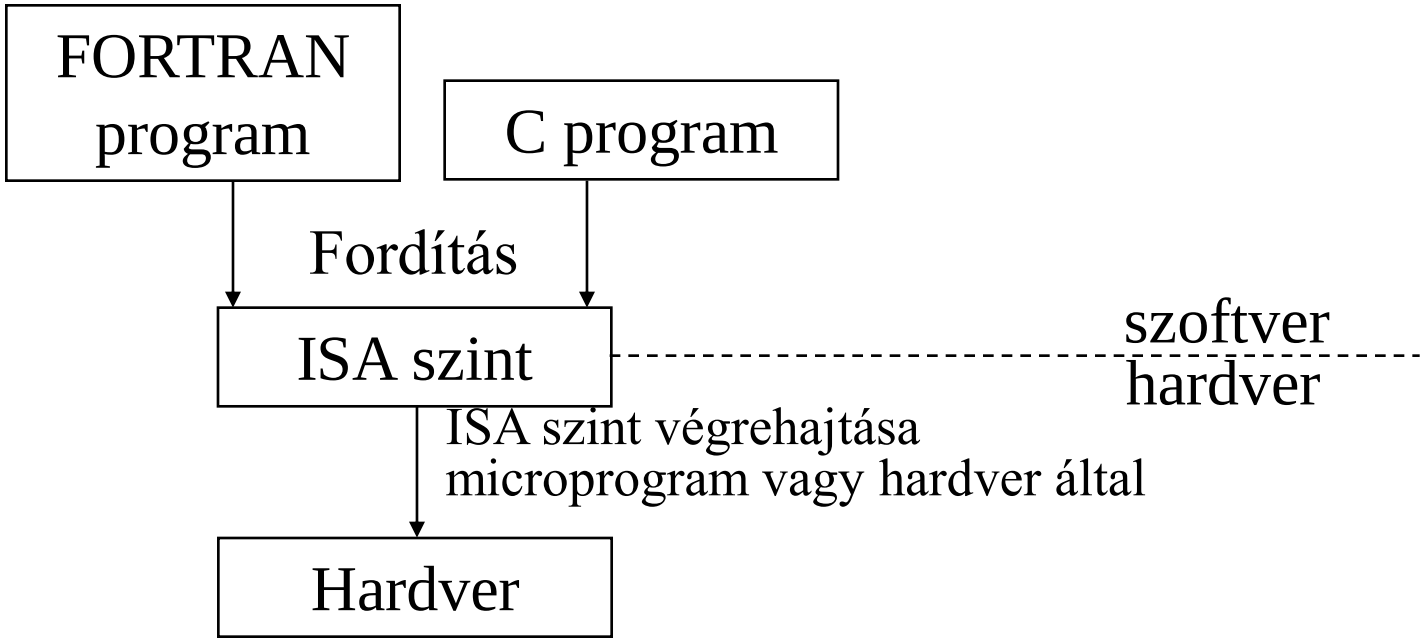
Számítógép architektúrák

- **Számítógépek felépítése**
- **Digitális adatábrázolás**
- **Digitális logikai szint**
- **Mikroarchitektúra szint**
- **Gépi utasítás szint**
- **Operációs rendszer szint**
- **Assembly nyelvi szint**
- **Probléma orientált (magas szintű) nyelvi szint**
- **Perifériák**

Utasításrendszer-architektúra szintje (ISA)

Amit a fordító program készítőjének tudnia kell:
memóriamodell, regiszterek, adattípusok, utasítások.

A hardver és szoftver között helyezkedik el, **5.1 ábra**.



Általában a mikroarchitektúra nem tartozik hozzá.

Utasítások szintje (ISA)

A jóság két kritériuma:
- hatékony hardver megvalósítási lehetőség,
- jó médium a fordítóknak.

Továbbfejlesztéseknél ügyelni kell a kompatibilitásra!

Nyilvános definíció:
van: **SPARC**, **JVM** (tervezők);
nincs: **Pentium 4** (gyártók).

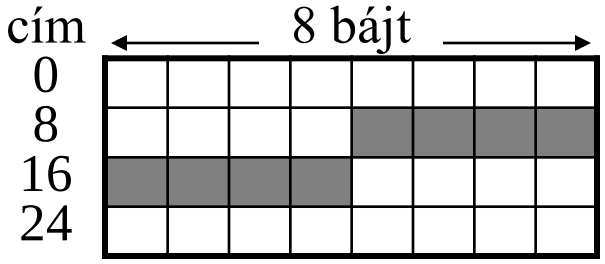
kernelmód <----> **(user) felhasználói mód**

Memória modellek

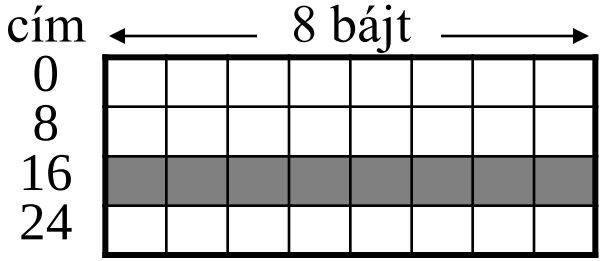
ASCII kód 7 bit + paritás ---> **Byte** (**bájt**)

Szó: 4 vagy 8 byte.

Igazítás (alignment), **5.2. ábra**: hatékonyabb, de probléma a kompatibilitás (a **Pentium 4**-nek két ciklusra is szüksége lehet egy szó beolvasásához).



Nem igazított 8 bájtos szó
a 12-es címtől



8 bájtos szó
8 határra igazítva

Néha (pl. **8051**) külön memória az adatoknak és az utasításoknak (nem ugyanaz, mint az osztott gyorsítótár!).

Memória modellek

Memória szemantika: STORE *A* -t közvetlenül követő **LOAD *A*** mit ad vissza?

A memória műveletek végrehajtása:

- kötött sorrendben,
- definiálatlan sorrendben (ez a trend, mert hardver szinten egyszerűbb és gyorsabb). A hardver segítséget nyújthat:
  - **SYNC** utasítás: befejeztet minden megkezdett memória műveletet,
  - függőség esetén a hardver vár.

Regiszterek

ISA-szinten a mikroszint nem minden regisztere látszik (**TOS**, **MAR**), de van közös is (**PC**, **SP**).

Speciális regiszterek: PC, **SP**, …

Általános célú regiszterek: a gyakran használt adatok gyors elérésére.
Jó, ha szimmetrikusak: fordítók, konvenciók.

RISC gépen általában legalább 32 általános célú.

Kernelmódban továbbiak: gyorsítótár vezérlés, memória védelem, …

PSW (Program Status Word): az eredmény negatív, nulla, ... mód, prioritásszint, megszakítás-állapot, ...

Utasításkészlet

LOAD, **STORE**,
MOVE, aritmetikai, logikai,
feltétlen, feltételes elágazó utasítások,
…

Pentium 4

Nagyon sok előd (kompatibilitás!), a fontosabbak:

- **4004**: 4 bites,
- **8086**, **8088**: 16 bites, 8 bites adat sín.
- **80286**: 24 bites (nem lineáris) címtartomány (16 K darab 64 KB-os szegmens).
- **80386**: **IA-32** architektúra, az Intel első 32 bites gépe, lényegében az összes későbbi is ezt használja.
- **Pentium II** –től **MMX** utasítások.

A Pentium 4 üzemmódjai

real (valós): az összes **8088** utáni fejlesztést kikapcsolja (valódi **8088-**ként viselkedik). Hibánál a gép egyszerűen összeomlik, lefagy.

virtuális 8086: a **8088-**as programok védett módban futnak (ha **WINDOWS-**ból indítjuk az **MS-DOS-**t, és abban hiba történik, akkor nem fagy le, hanem visszaadja a vezérlést a **WINDOWS-**nak).

védett: valódi **Pentium 4**. 4 védelmi szint (**PSW**):

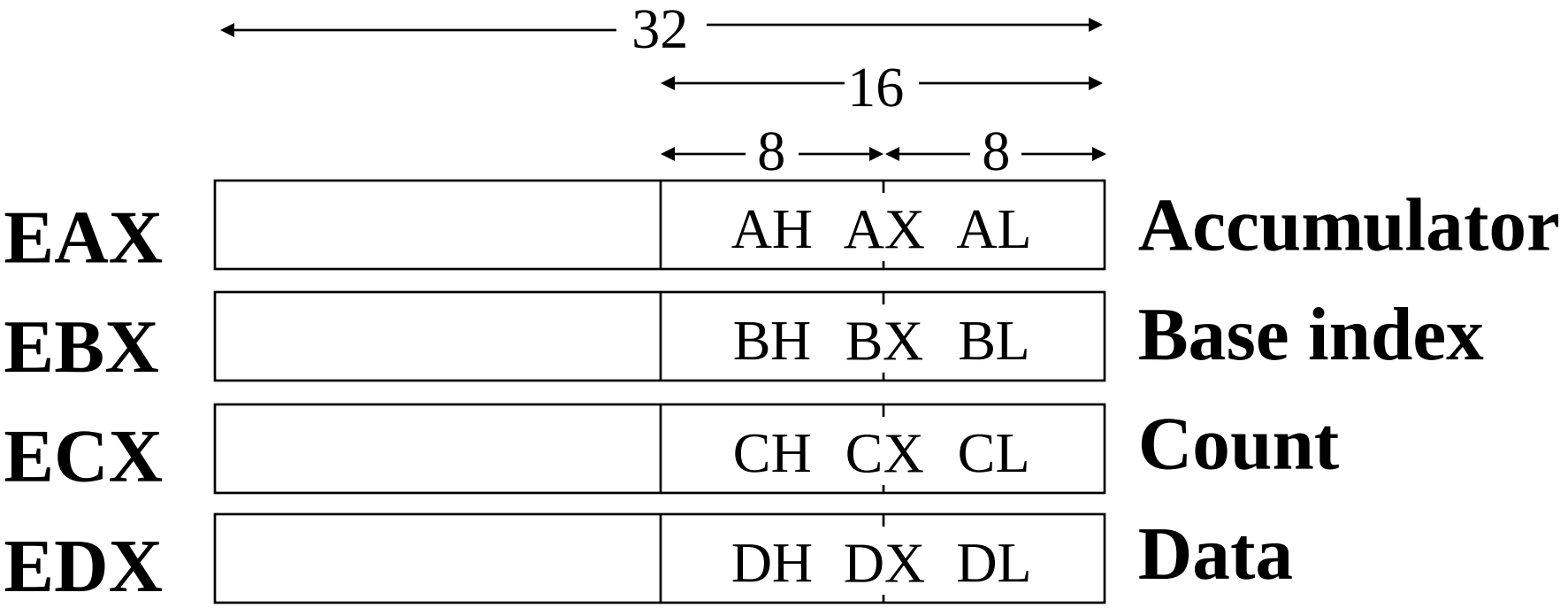
0: kernelmód (operációs r.), **1**, **2**: ritkán használt, **3**: felhasználói mód.

Memóriaszervezés:

- **16 K darab szegmens** lehetséges, de a **WINDOWS**-ok és **UNIX/LINUX** is csak **1** szegmenst támogatnak, és ennek is egy részét az operációs rendszer foglalja el,
- minden szegmensen belül a címtartomány: **0 - 2^{32}-1**
- **Little endian** tárolási mód: az alacsonyabb címen van az alacsonyabb helyértékű bájt.

Regiszterek (5.3. ábra):

- (majdnem) általános regiszterek:



Ezek 8 és 16 bites részei önálló regiszterként használhatók.

Regiszterek (5.3. ábra):

- **ESI**, **EDI** (mutatók tárolására, szöveg kezelésre),
- **EBP** (keretmutató, verem kezelésre),
- **ESP** (verem mutató),

Utasítás szerkezet szerint az eddigi 4 is általános célú

- **EIP** (utasítás számláló),
- **EFLAGS** (**PSW**),
- **CS**, **SS**, **DS**, **ES**, **FS**, **GS** (16 bites regiszterek. A kompatibilitást biztosítják a régebbi gépekkel. Mivel a **Windows**, **Unix** csak egy címtartományt használ, ezekre csak a visszafelé kompatibilitás miatt van szükség).

UltraSPARC III

SPARC 1987 még **32**, a **Version 9** már **64 bites architektúra**, az **UltraSPARC** ezen alapul.

Memóriaszervezés: **64 bites** (lineáris) címtartomány (jelenleg maximum **44 bit** használható).
Big endian, de **little endian** is beállítható.

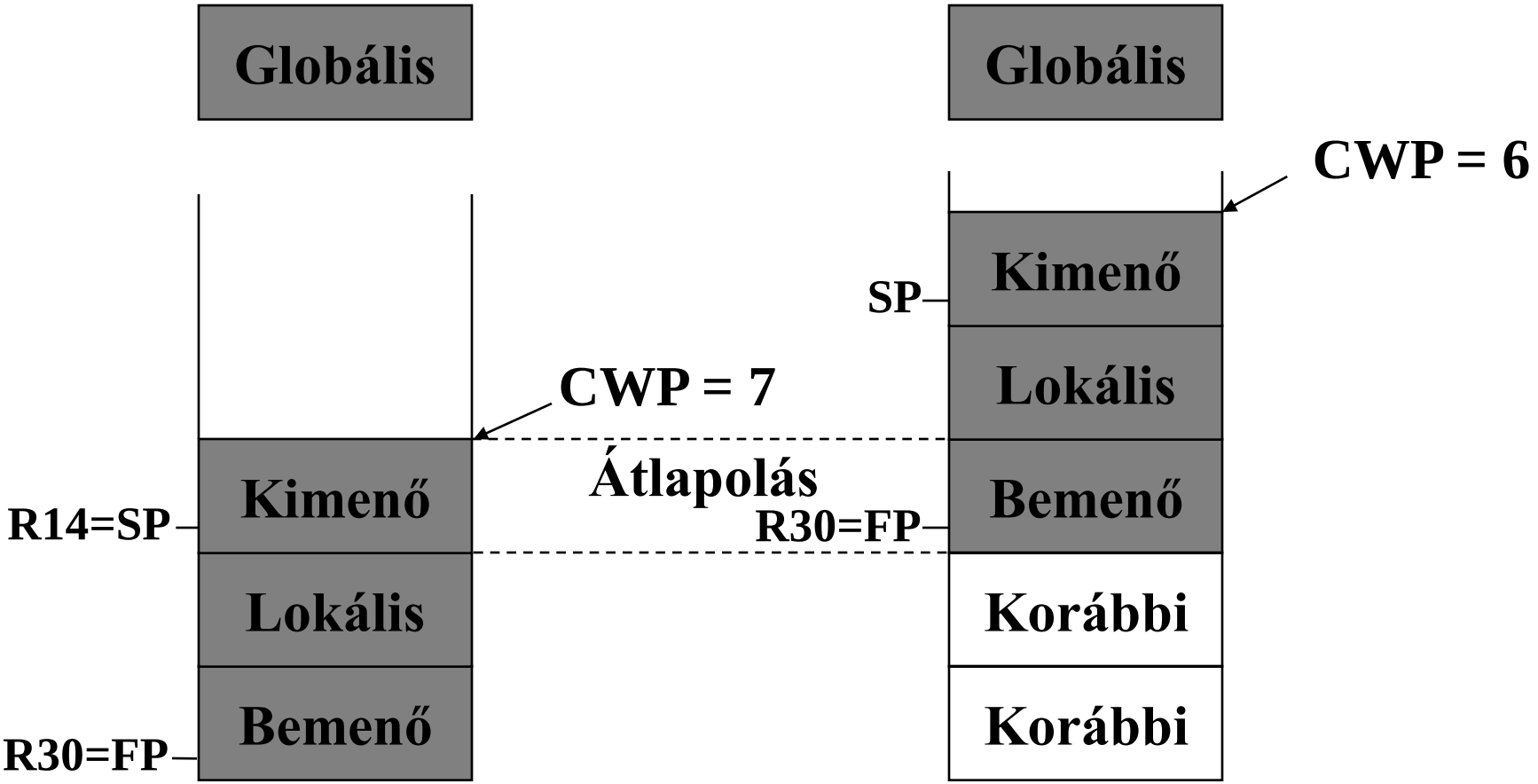
Regiszterek:

- **32 általános** (**5.4. ábra**) **64 bites**, a használatuk részben konvención, részben a hardveren alapul),
- **32 lebegőpontos** (**32** vagy **64 bites**, de lehetséges két regiszterben egy **128 bites** számot tárolni).

Általános regiszterek

- **R0-R7** (**G0-G7**) Globális változók: minden eljárás használhatja,
  G0 huzalozott **0**, minden tárolás eredménytelen.
- **R8-R15** (**O0-O7**,): Kimenő paraméterek,
  de **R14** (**O6**) = **SP**: verem mutató
  O7 csak ideiglenes tárolásra használható.
- **R16-R23** (**L0-L7**) Lokális regiszterek
- **R24-R31** (**I0-I7**) Bejövő paraméterek, de
  R30 (**I6**) = **FP** az aktuális veremkeret mutatója,
  R31: visszatérési cím.

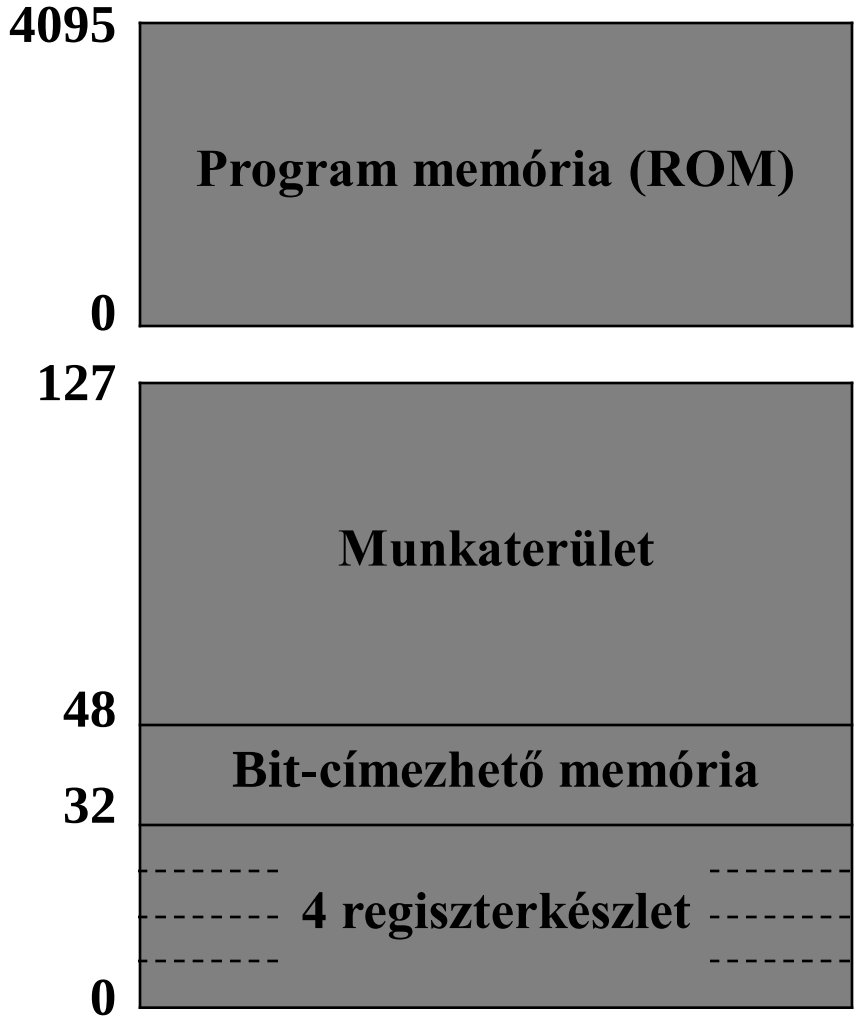
CWP (Current Window Pointer, **5.5. ábra**) mutatja az aktuális **regiszter ablak**ot (több regiszter készlet létezik, de mindig csak egy látszik). Ha kifogy a regiszter készlet, memóriába mentés, …



Load/store architektúra: csak ezek az utasítások fordulhatnak a memóriához. A többi utasítás operandusa regiszterben vagy az utasításban van. Az eredmény is regiszterbe kerül.

5.6. ábra. A **8051** memória szervezése, fő regiszterei

Külön címtartományú program és adat memória.



Vannak lapkán kívüli bővítési lehetőségek. Van nagyobb (**8052**) és programozható (**8751** és **8752**) „rokona” (**ROM** helyett **EPROM**).

C	A		RS		O		P	PSW
EA		E2	ES	E1	X1	E0	X0	IE
		E2	ES	E1	X1	E0	X0	IP
O1	R1	O0	R0	E1	X1	E0	X0	TCON
1. időzítő				2. időzítő				TMOD
<--- 8 bit --->								

8 regiszter: **R0**, … , **R7**. A regiszterek a memóriában vannak.

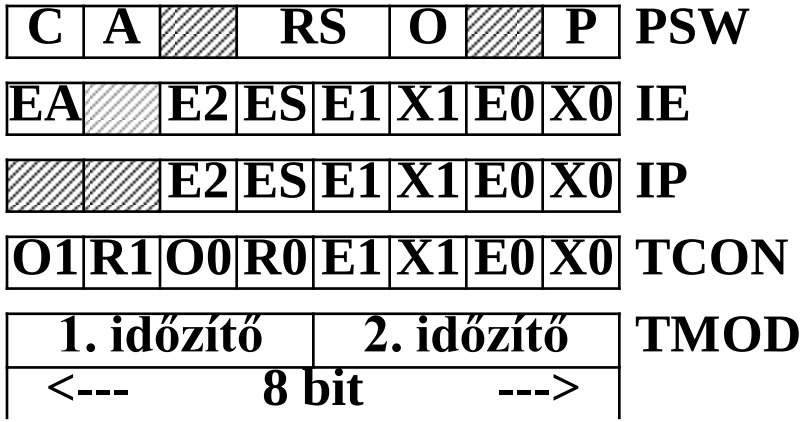
4 regiszter készlet, de egyszerre csak egy használható.

PSW RS mezeje mondja meg, hogy melyik az aktuális.

Bit-címezhető memória (32-47. bájt): címzésük: 0-127

Bit utasítások: beállítás, törlés, ÉS, VAGY, tesztelés.

PSW: **C**arry, **A**uxiliary carry, **R**egister**S**, **O**verflow, **P**arity



Az eddig említett és még néhány speciális regiszter (**ACC**, **B/K** portok, …) a 128-255 címtartományban vannak.
Pl. **ACC** a 240-en.

A 8052 valódi memóriát tartalmaz a 128-255 tartományban, a speciális regiszterek átfednek a memóriával.
– Direkt címzéssel a speciális regisztereket,
– Indirekt címzéssel a **RAM**-ot érhetjük el.

Címzési módszerek

3, 2, 1, 0 címes utasítások.

Címzési módok:

- közvetlen operandus,
- direkt címzés,
- regiszter címzés
- regiszter-indirekt címzés,
- indexelt címzés,
- bázisindex címzés,
- verem címzés.

Verem címzés
Fordított Lengyel Jelölés
(Postfix Polish Notation - Lukasiewicz)

Postfix jelölés: a kifejezéseket olyan formában adjuk meg, hogy az első operandus után a másodikat, majd ezután adjuk meg a műveleti jelet:

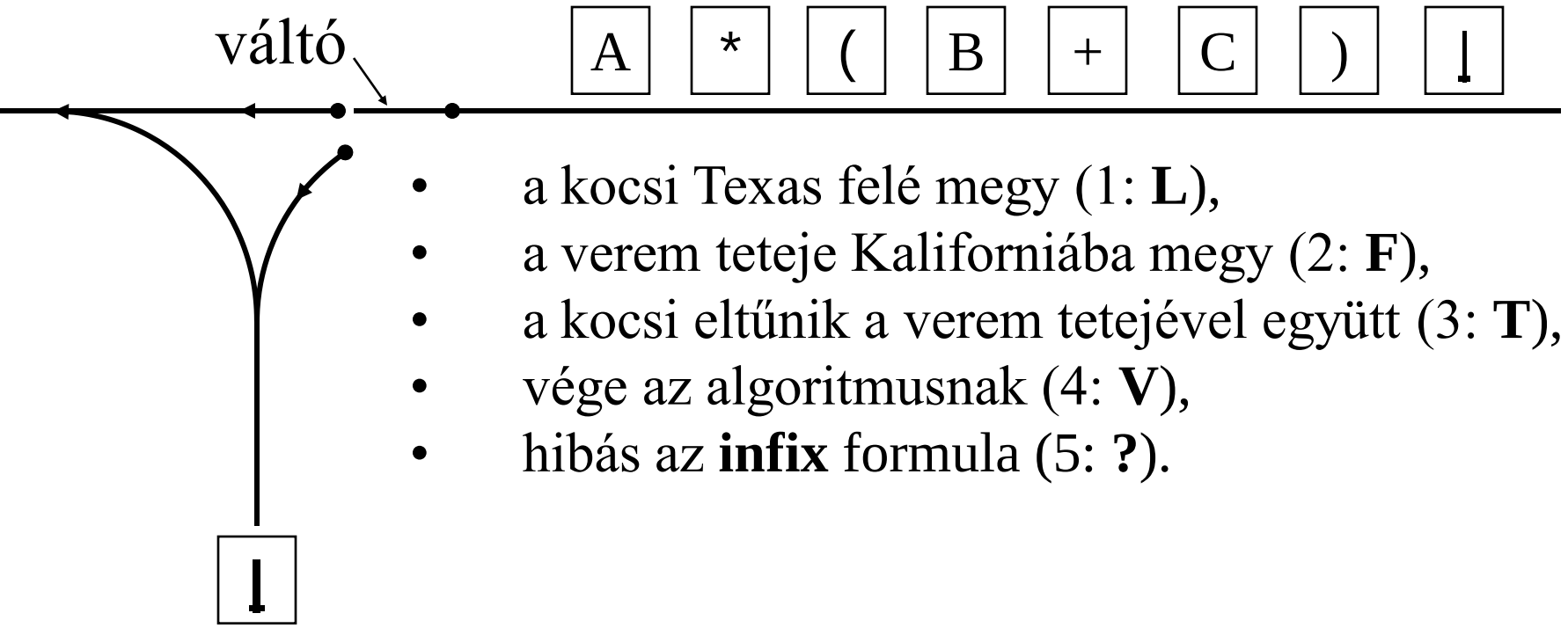
infix: $x + y$, **postfix**: $x\ y\ +$.

Előnyei: nem kell zárójel, sem precedencia szabályok, jól alkalmazható veremcímzés esetén.

Dijkstra algoritmusa

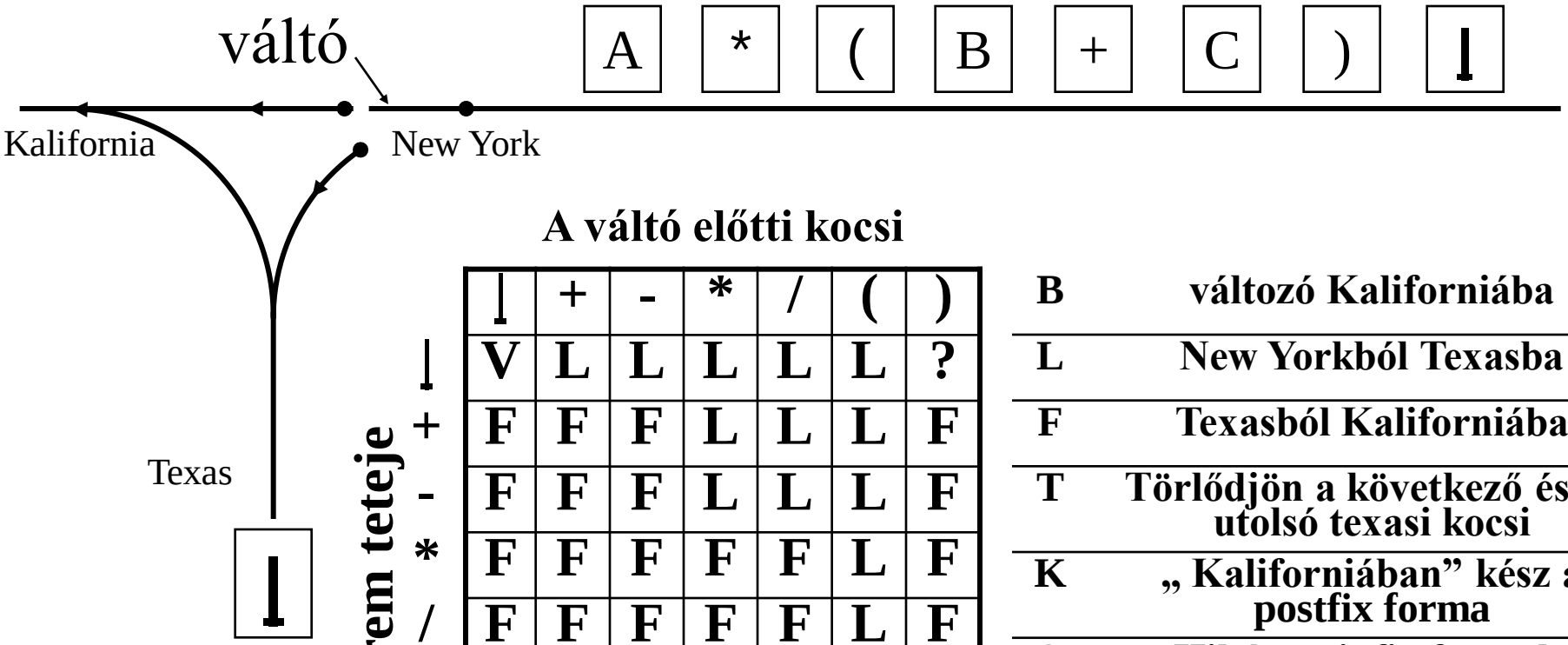
Infix jelölés konvertálása postfix-re (5.21, 22. ábra):

- az **infix** elemek egy váltóhoz (switch) érkeznek - a változók és konstansok Kaliforniába mennek (←),
- a többi esetben a verem tetejétől függően (**5.22. ábra**):



- a kocsi Texas felé megy (1: **L**),
- a verem teteje Kaliforniába megy (2: **F**),
- a kocsi eltűnik a verem tetejével együtt (3: **T**),
- vége az algoritmusnak (4: **V**),
- hibás az **infix** formula (5: **?**).

Minden változó és konstansok menjen Kaliforniába (←),
a többi esetben a döntési tábla szerint járjunk el (**5.21. ábra**):



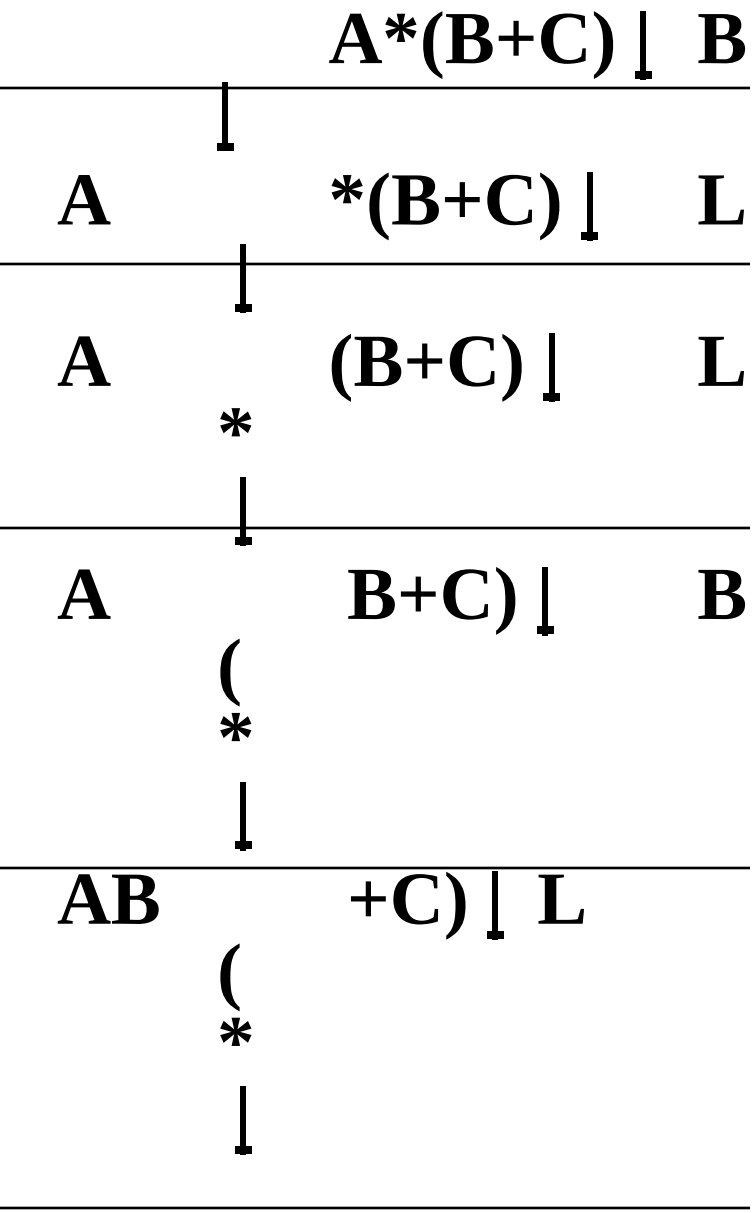
A váltó előtti kocsi

A verem teteje

	|	+	-	*	/	(	)
|	V	L	L	L	L	L	?
+	F	F	F	L	L	L	F
-	F	F	F	L	L	L	F
*	F	F	F	F	F	L	F
/	F	F	F	F	F	L	F
(	?	L	L	L	L	L	T

B	**változó Kaliforniába**
L	**New Yorkból Texasba**
F	**Texasból Kaliforniába**
T	**Törlődjön a következő és az utolsó texasi kocsi**
K	**„ Kaliforniában” kész a postfix forma**
?	**Hibás az infix formula**

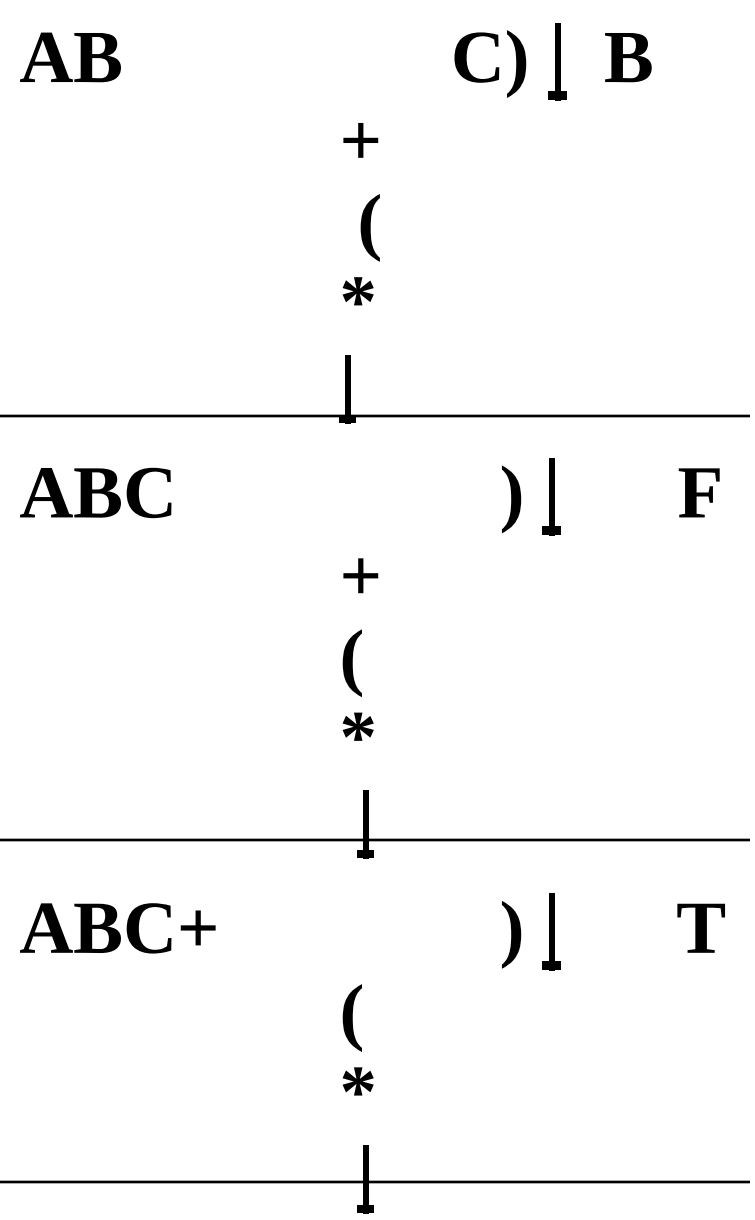
A döntési tábla tartalmazza a prioritási szabályokat.



A váltó előtti kocsi

A verem teteje

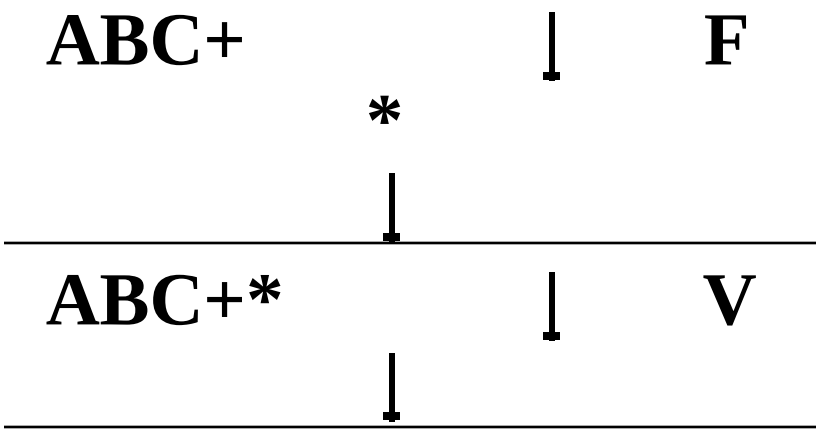
	⊥	+	-	*	/	(	)
⊥	V	L	L	L	L	L	?
+	F	F	F	L	L	L	F
-	F	F	F	L	L	L	F
*	F	F	F	F	F	L	F
/	F	F	F	F	F	L	F
(	?	L	L	L	L	L	T



A váltó előtti kocsi

A verem teteje

	+	-	*	/	(	)
V	L	L	L	L	L	?
+	F	F	F	L	L	L	F
-	F	F	F	L	L	L	F
*	F	F	F	F	F	L	F
/	F	F	F	F	F	L	F
(	?	L	L	L	L	L	T



Fordított lengyel jelölésű formulák kiértékelése

Pl. (**5.24. ábra**):

(8 + 2 * 5)/(1 + 3 * 2 – 4) // infix
8 2 5 * + 1 3 2 * + 4 – / // postfix

Olvassuk a formulát balról jobbra!

Ha a következő jel

- **operandus**: rakjuk a verembe,
- **műveleti jel**: hajtsuk végre a műveletet (a verem tetején van a jobb, alatta a bal operandus!).

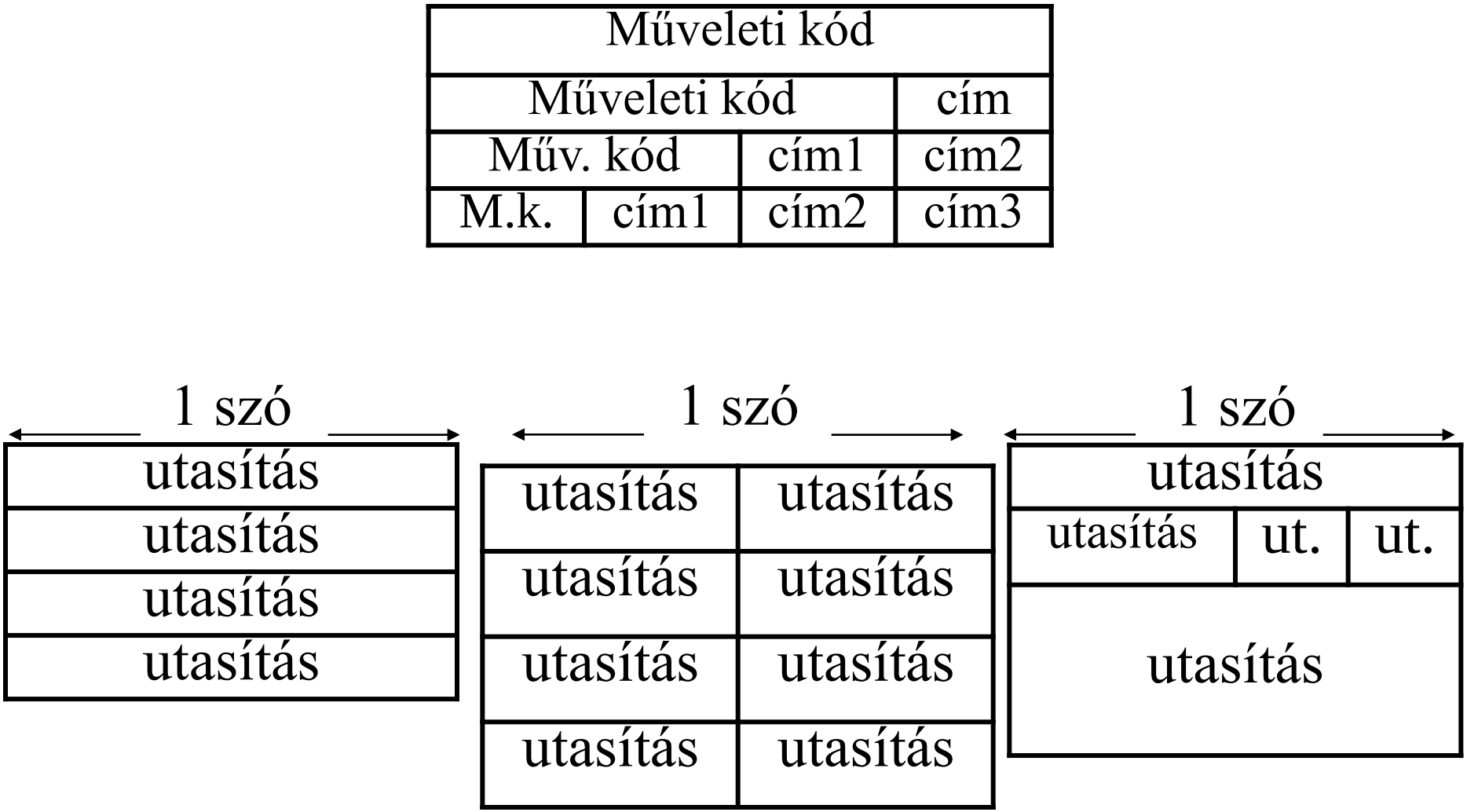
(8 + 2 * 5)/(1 + 3 * 2 – 4) // infix

Lépés	Maradék formula	Utasítás	Verem
1	8 2 5 * + 1 3 2 * + 4 - /	BIPUSH 8	8
2	2 5 * + 1 3 2 * + 4 - /	BIPUSH 2	8, 2
3	5 * + 1 3 2 * + 4 - /	BIPUSH 5	8, 2, 5
4	* + 1 3 2 * + 4 - /	IMUL	8, 10
5	+ 1 3 2 * + 4 - /	IADD	18
6	1 3 2 * + 4 - /	BIPUSH 1	18, 1
7	3 2 * + 4 - /	BIPUSH 3	18, 1, 3
8	2 * + 4 - /	BIPUSH 2	18, 1, 3, 2
9	* + 4 - /	IMUL	18, 1, 6
10	+ 4 - /	IADD	18, 7
11	4 - /	BIPUSH 4	18, 7, 4
12	- /	ISUB	18, 3
13	/	IDIV	6

Az ISA szint tervezési szempontjai

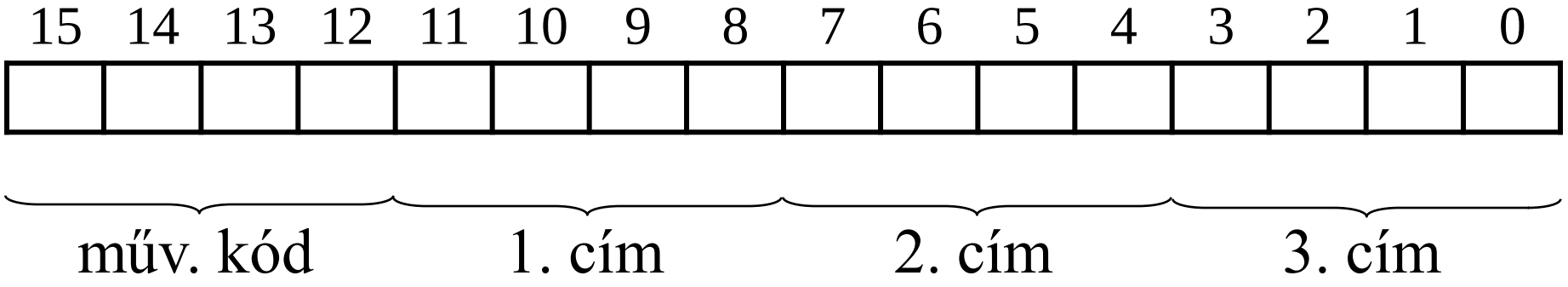
- **Hosszú távú**: később is jó legyen az architektúra,
  Rövid távú: addig is piacon kell maradni.
- **Rövidebb utasítások**: kevesebb helyet foglalnak el, gyorsabban betölthetők.
  Hosszabb utasítások: több lehetséges műveleti kód, nagyobb memória címezhető.
- **Bájt címzés**: hatékonyabb szöveg feldolgozásnál,
  Szó címzés: nagyobb memória címezhető.
- …

Utasításformák, utasításhossz (**5.10-11. ábra**).



A műveleti kód kiterjesztése

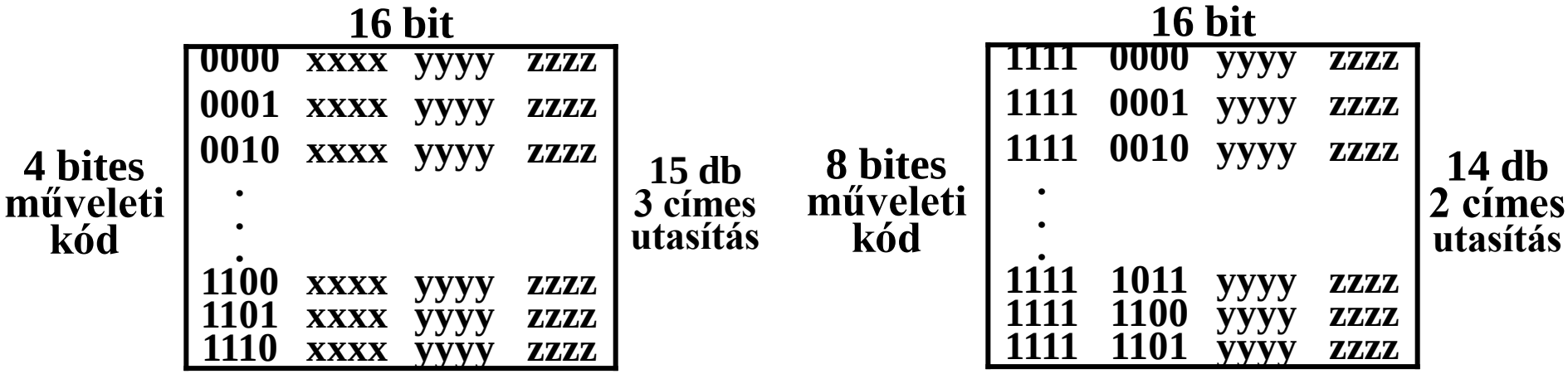
k bites műveleti kód esetén 2^k különböző utasítás lehet, ***n*** bites címrésznél 2^n memória címezhető, fix utasítás hossz esetén egyik csak a másik rovására növelhető (**5.12. ábra**).



Lehetőségek:

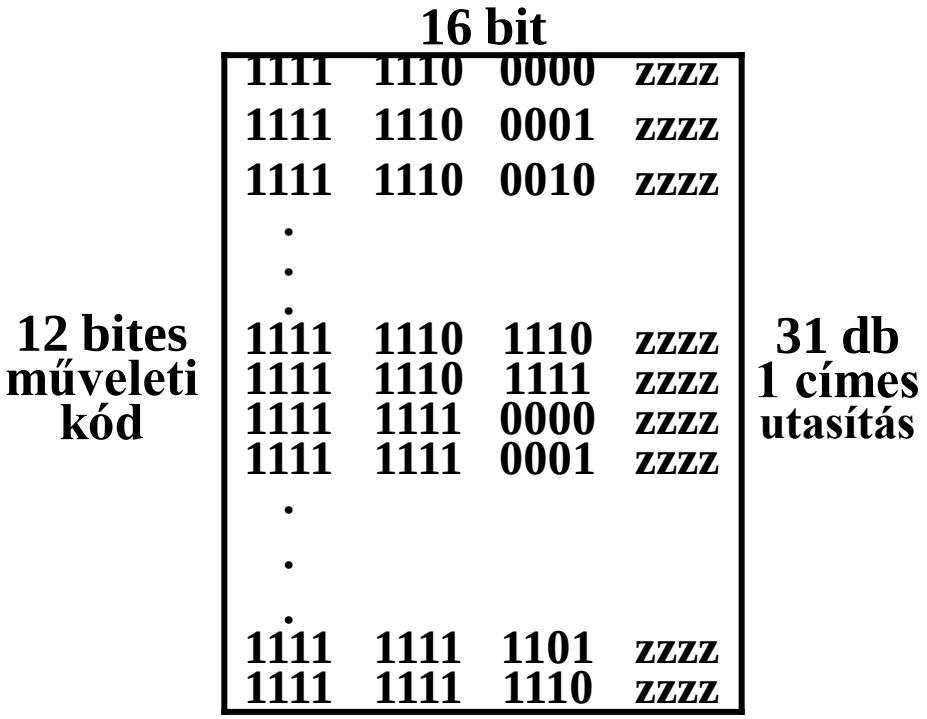
- fix utasításhossz: rövidebb kód mellett hosszabb operandus rész,
- minimális átlagos utasításhossz: a gyakori kódok rövidek, a ritkán használtak hosszabbak.

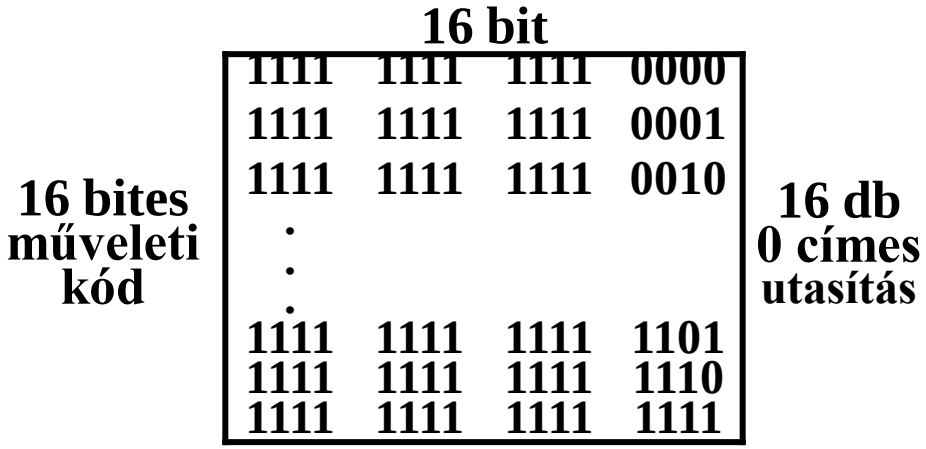
A műveleti kód kiterjesztése (5.13. ábra)



Az **1111** kódot nem használtuk ki 3 címes utasításnak **(menekülő kód**), és ez lehetővé teszi, hogy további – igaz, nem 3 címes – utasításokat adjunk meg.
1111 1110 és **1111 1111** is menekülő kód.

A műveleti kód kiterjesztése

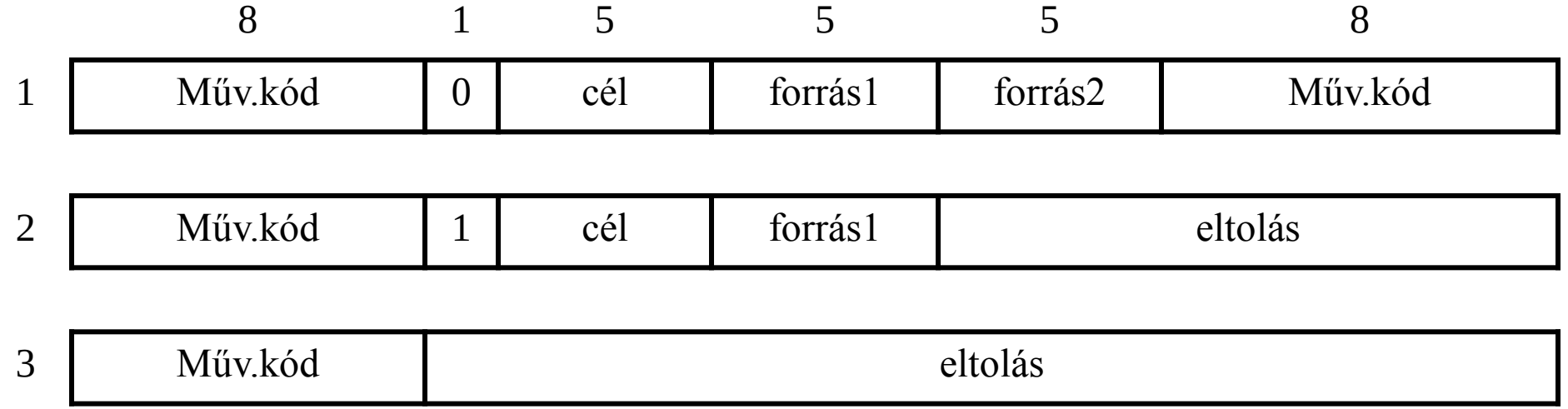




1111 1111 1111 is menekülő kód.

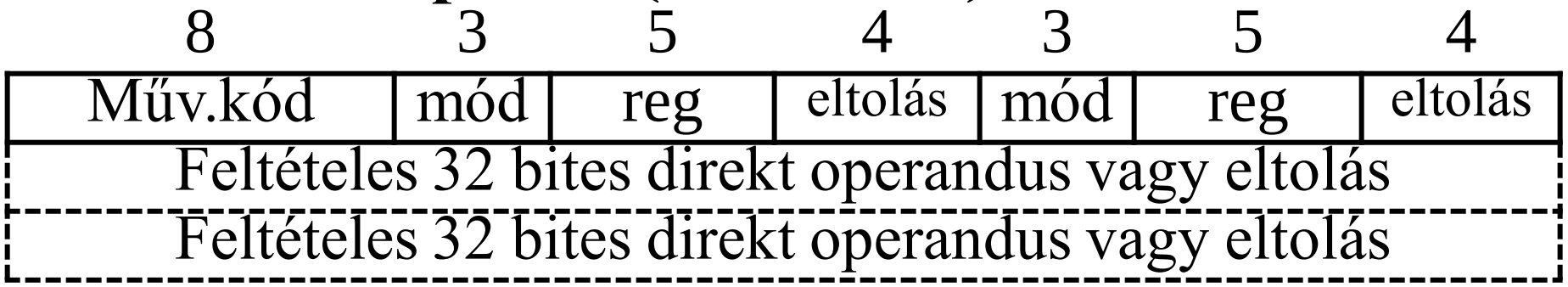
Ortogonalitási elv: Jó architektúrában a műveleti kódok és a címzési módszerek (majdnem) szabadon párosíthatók.

Három címes elképzelés (5.25. ábra):



1. típus: aritmetikai utasítások.
2. típus: közvetlen adat megadás,
   index módú **`LOAD`** és **`STORE`** utasítás.
3. típus: elágazó, eljárás hívó utasítások,
   `LOAD` és **`STORE`**, ezek **`R0`**-t használnák.

Két címes elképzelés (5.26. ábra).



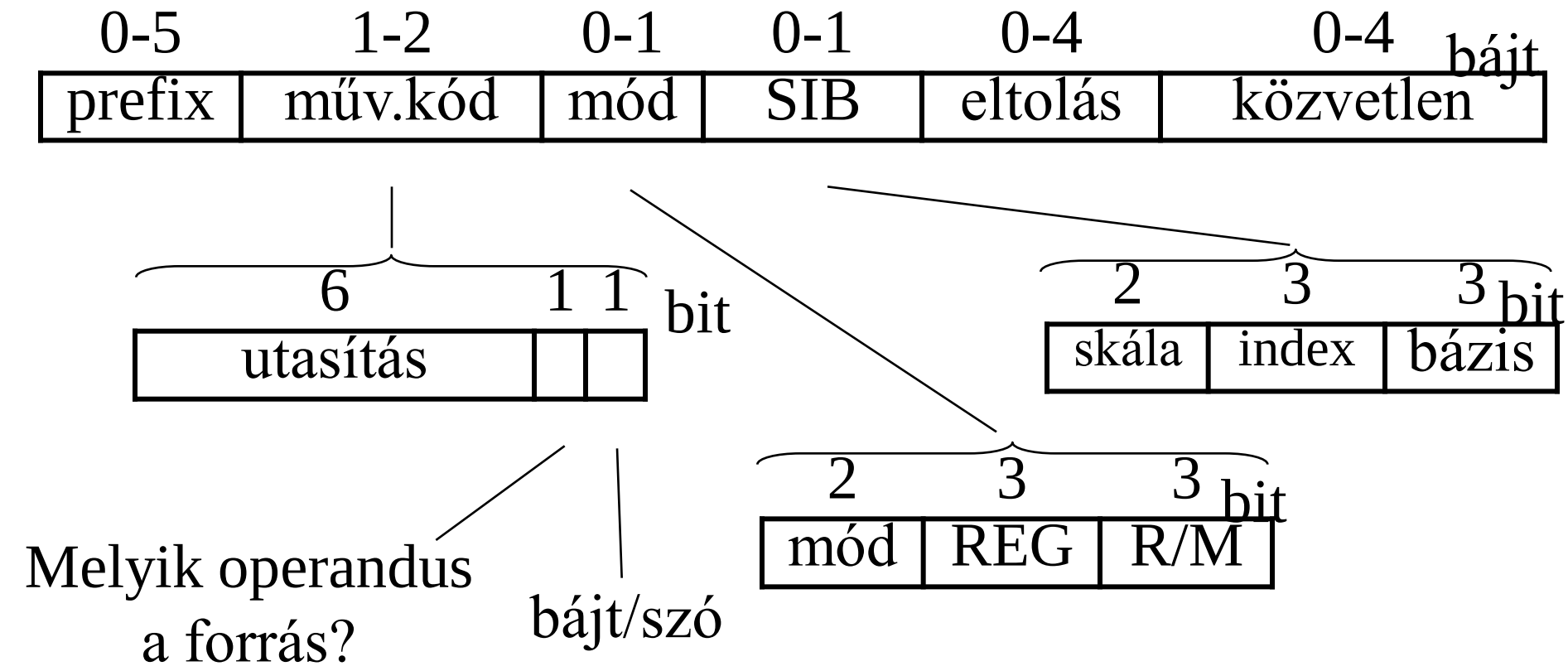
A mód 3 bitje lehetővé teszi a
közvetlen operandus,
direkt,
regiszter,
regiszter indirekt,
index és
verem címzési módokat
Két további mód bevezetésére is lehetőség van.

Pentium 4 utasításformái (5.14. ábra)

Több generáción keresztül kialakult architektúra.

Csak egy operandus lehet memória cím.

Prefix, **escape** (bővítésre), **`MOD`**, **`SIB`** (Scale Index Base)



Címzési módok (5.27. ábra): nagyon szabálytalan.

Baj: nem minden utasításban használható minden mód,
nem minden regiszter használható minden módban (nincs **`EBP`** indirekt, **`ESP`** relatív címzés). 32 bites címzési módok:

	MÓD			
R/M	00	01	10	11
000	M[EAX]	M[EAX+offset8]	M[EAX+offset32]	EAX v. AL
001	M[ECX]	M[ECX+offset8]	M[ECX+offset32]	ECX v. CL
010	M[EDX]	M[EDX+offset8]	M[EDX+offset32]	EDX v. DL
011	M[EBX]	M[EBX+offset8]	M[EBX+offset32]	EBX v. BL
100	SIB	SIB offset8-cal	SIB offset32-vel	ESP v. AH
101	direkt	M[EBP+offset8]	M[EBP+offset32]	EBP v. CH
110	M[ESI]	M[ESI+offset8]	M[ESI+offset32]	ESI v. DH
111	M[EDI]	M[EDI+offset8]	M[EDI+offset32]	EDI v. BH

UltraSPARC utasításformái (5.15. ábra)

32 bites egyszerű utasítások.

Form.	2	5	6	5	1	8	5	
1a	m.k.	cél	m.k.	forrás1	0	FP-m.k.	forrás2	3 címes
1b	m.k.	cél	m.k.	forrás1	1	közvetlen konst.		2 címes

Aritmetikai utasítások:

1 cél és 2 forrás regiszter vagy

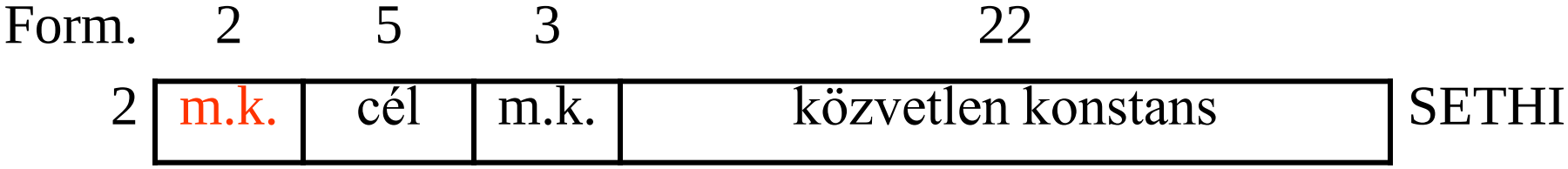
1 cél, 1 forrás regiszter és 1 közvetlen konstans.

`LOAD`, **`STORE`** (csak ezek használják a memóriát):

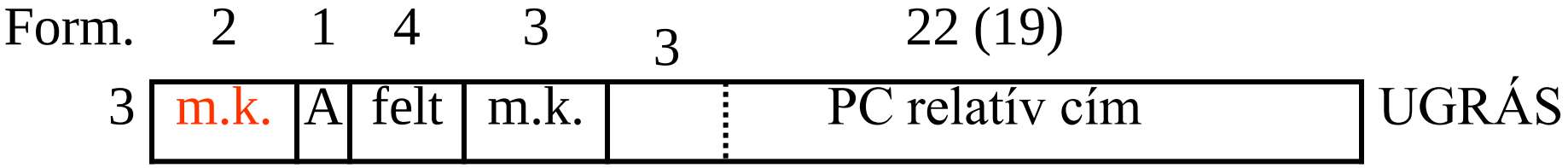
a cím két regiszter összege vagy

egy regiszter + 13 bites eltolás.

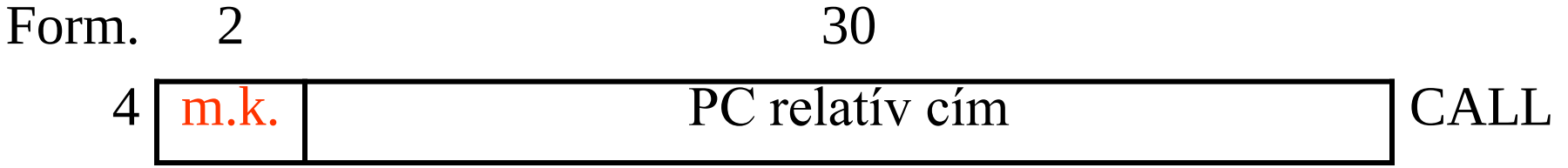
Processzorokat szinkronizáló utasítás.



32 bites közvetlen adat megadása: **SETHI** – megad 22 bitet, a következő utasítás a maradék 10 bitet.



Az ugrások **PC**-relatívok, szót (4-gyel osztható bájt címet) címeznek. Jósláshoz 3 bitet elcsíptek. Az **A** bit az eltolás rést akadályozza meg bizonyos feltételek esetén.

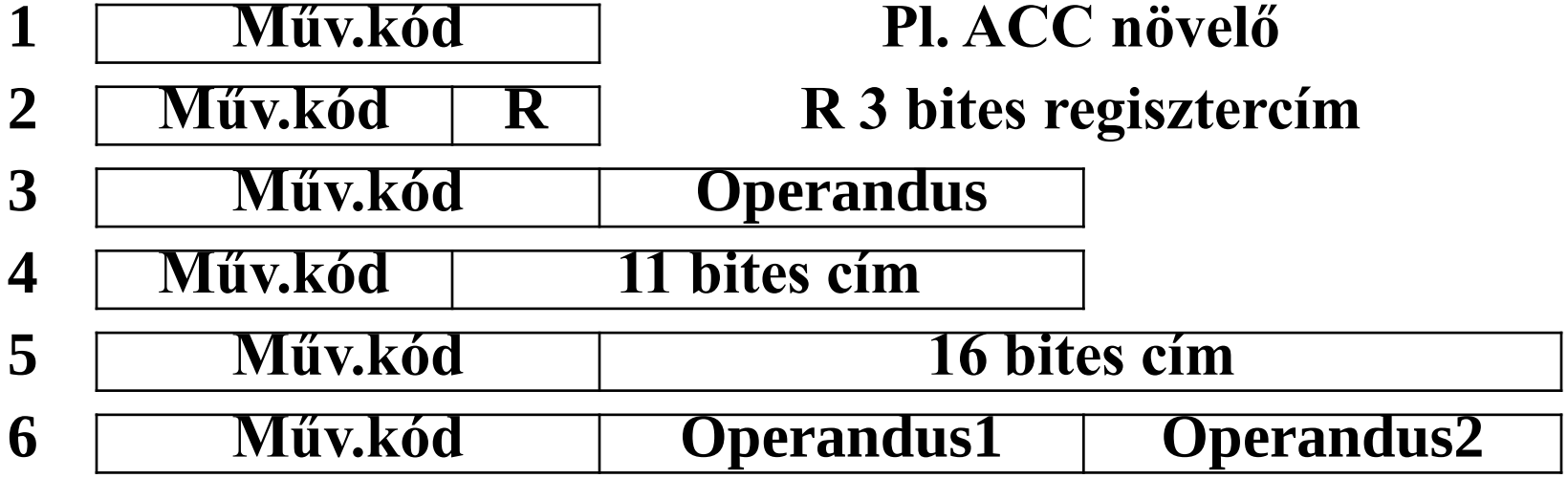


Eljárás hívás: 30 bites **PC**-relatív (szó) cím

UltraSPARC címzési módjai

Memóriára hivatkozó utasítások:
betöltő, tároló, multiprocesszor szinkronizáló
index + 13 bit eltolás, bázis-index
A többi utasítás általában 5 bites regiszter címzést használ

A **8051** utasításformátumai



1. Implicit regiszter általában **ACC**, …
2. Regiszter és **ACC** tartalmán végzett művelet, mozgatás, …
3. Operandus: közvetlen, eltolás, bitsorszám
4. Ugrás, szubrutin hívás
5. Ugrás, szubrutin hívás
6. Pl. közvetlen operandus memóriába töltése, …

A 8051 címzési módjai

Implicit: **ACC**

Regiszter: akár forrás, akár cél operandus lehet

Direkt: 8 bites memóriacím

Regiszter-indirekt: 8 bites memóriacím

Indirekt címzés a 16 bites **DPTR**-rel

Közvetlen operandus: általában 8 bites, de 11 ill. 16 bites abszolút cím ugráshoz, eljárás híváshoz

Összefoglaló: 5.29. ábra.

Címzési mód	Pentium 4	UltraSPARC III	8051
Akkumulátor			X
Közvetlen	X	X	X
Direkt	X		X
Regiszter	X	X	X
Regiszter indirekt	X		X
Index	X	X	
Bázis-index	X	X	
Verem			

A bonyolult címzési módok tömörebb programokat tesznek lehetővé, de nehezítik a párhuzamosítást. Ha a párosítás nem történhet szabadon, akkor jobb, ha csak egy választási lehetőség van (egyszerűbb hatékony fordítóprogramot írni).