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

Három címes elképzelés (5.25. ábra):

	8	1	5	5	5	8
1	Műv.kód	0	cél	forrás1	forrás2	Műv.kód
2	Műv.kód	1	cél	forrás1	eltolás	
3	Műv.kód	eltolás				

1. típus: aritmetikai utasítások.
2. típus: aritmetikai utasítások 13 bites közvetlen adattal, index módú **LOAD** és **STORE** utasítás 13 bites eltolással.
3. típus: PC relatív ($\pm 2^{23}$ szó) elágazó, eljárás hívó utasítások, **LOAD** és **STORE**, ezek **R0**-t használnák.

Két címes elképzelés (5.26. ábra).

8	3	5	4	3	5	4
Műv.kód	mód	reg	eltolás	mód	reg	eltolás
Feltételesen: 32 bites direkt operandus vagy eltolás						
Feltételesen: 32 bites direkt operandus vagy eltolás						

A mód 3 bitje lehetővé teszi a
közvetlen operandus,
direkt,
regiszter,
regiszter indirekt,
index és
verem címzési módokat

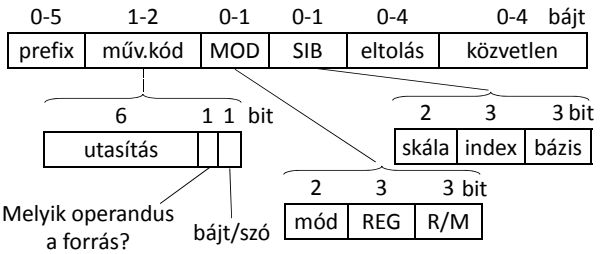
Két további mód bevezetésére is lehetőség van.

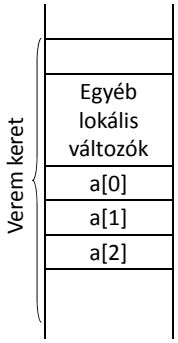
Pentium 4 utasításformái (5.14. ábra)

Több generáción keresztül kialakult architektúra.

Csak egy operandus lehet memória cím.

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





2	3	3 bit
skála	index	bázis

Legyen *i* az EAX regiszterben

SIB módú hivatkozás: M[4*EAX+EBP+8]

`SIB` (**5.28. ábra**): jó, de megéri?

Címzési módok (**5.27. ábra**): nagyon szabálytalan.

Bajok:

- 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.

Forma	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.

Forma	2	5	3	22	
2	m.k.	cél	m.k.	közvetlen konstans	**SETHI**

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

Forma	2	1	4	3	3	22 (19)	
3	m.k.	A	felt	m.k.		PC relatív cím	**UGRÁS**

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

Forma	2	30	
4	m.k.	PC relatív cím	**CALL**

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

UltraSPARC címzési módjai

Memóriára hivatkozó utasítások:
betöltő, tároló, multiprocesszor szinkronizáló
bázis-index (1a),
index + 13 bit eltolás (1b).

A többi utasítás általában 5 bites regiszter címzést használ

Az **I-8051** utasításformátumai

1 | **Műv.kód** |

Implicit regiszter általában **ACC**, …pl. **ACC** növelő

2 | **Műv.kód** | **R** | **R 3 bites regisztercím**

Regiszter és **ACC** tartalmán végzett művelet, mozgatás, …

3 | **Műv.kód** | **Operandus** |

Operandus: közvetlen, eltolás, bitsorszám

4 | **Műv.kód** | **11 bites cím** |

Ugró és szubrutin (eljárás) hívó utasítások (**cím < 2048**)

5 | **Műv.kód** | **16 bites cím** |

Ugró és szubrutin (eljárás) hívó utasítások

6 | **Műv.kód** | **Operandus1** | **Operandus2** |

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 vagy **PC**-vel
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	X
Index	X	X	
Bázis-index	X	X	
Verem			

Nem kell

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).

Utasítástípusok

- Adatmozgató (másoló) utasítások.
- Diadikus: **+**, **−**, *****, **/**, **AND**, **OR**, **NOT**, **XOR**, …
- Monadikus: léptetés, forgatás, **CLR**, **INC**, **NEG**, …
- Összehasonlítás, feltételes elágazás: **Z**, **O**, **C**, …
- Eljáráshívás. Visszatérési cím:
  - rögzített helyre (rossz),
  - az eljárás első szavába (jobb),
  - verembe (rekurzív eljárásokhoz is jó).
- Ciklusszervezés (**5.30. ábra**): számláló
- Input/output (**5.31-33. ábra**):
  - programozott **I/O**: tevékeny várakozás, **5.32. ábra**,
  - megszakítás vezérelt **I/O**,
  - **DMA I/O** (**5.33. ábra**): ciklusolás.

Ciklusszervezés (5.30. ábra)

```
      i=1;                      i=1;
L1:   első utasítás       L1:   if(i > n) goto L2;
      .                         első utasítás
      .                         .
      .                         .
      utolsó utasítás           .
      i = i + 1;                utolsó utasítás
      if(i ≤ n) goto L1;        i = i + 1;
                                goto L1;
                          L2:   …
```

Végfeltételes ismétlés ***Kezdő feltételes ismétlés***

Input, output (I/O) utasítások

A külvilággal történő információ csere **port**-okon (kapukon) keresztül zajlik. A kapu egy memória cím, az információ csere erre a címre történő írással, vagy erről a címről való olvasással történik. Egy-egy cím vagy cím csoport egy-egy perifériához kötődik.

Input/output

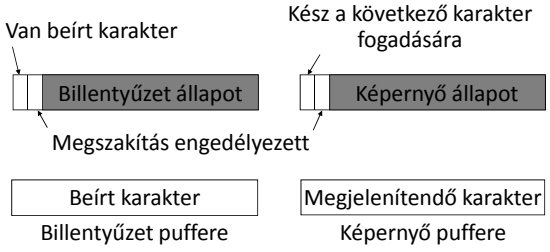
Az **I/O** végrehajtása lassú ↔ a **CPU** gyors, a **CPU** várakozni kényszerül

I/O regiszter (**port**): a **port** és a központi egység közötti információ átadás gyors, a periféria autonóm módon elvégzi a feladatát. Újabb perifériához fordulás esetén a **CPU** várakozni kényszerülhet.

- **Pollozásos technika** (~tevékeny várakozás)**:** a futó program időről időre megkérdezi a periféria állapotát, és csak akkor ad ki újabb **I/O** utasítást, amikor a periféria már fogadni tudja. A hatékonyság az éppen futó programtól függ.

Input/output (5.31-33. ábra)

Az **I/O** vezérlő regiszterei (**5.31. ábra**).
Terminál: külön regiszterek az inputra és outputra.



- **Pollozásos technika** (tevékeny várakozás)**:**

```
public static void output_buffer(int buf[], int count) {
 // count számú bájt kiírása buf-ból az eszközre
int status, i, ready;
 for(i=0; i < count; i++) {
   do {
     status = in(display_status_reg); // az állapot lekérdezése
     ready = (status >> 7) & 0x01;   // a kész bit elkülönítése
   } while(ready != 1);
   out(display_buffer_reg, buf[i]);
 }
}
```

Programozott **B/K** (**5.32. ábra**)

Megszakítás

A (program) megszakítás azt jelenti, hogy az éppen futó program végrehajtása átmenetileg megszakad – a processzor állapota megőrződik, hogy a program egy későbbi időpontban folytatódhassék – és a processzor egy másik program, az úgynevezett **megszakítás kezelő** végrehajtását kezdi meg.

Miután a megszakítás kezelő elvégezte munkáját, gondoskodik a processzor megszakításkori állapotának visszaállításáról, és visszaadja a vezérlést a megszakított programnak.

- **Megszakítás vezérelt I/O**
  Bizonyos előkészületek után az eszköz megkapja a feladatát (pl. az első karakter kiírását), és a program futása folytatódik.
  Ha az eszköz elkészült a feladatával, akkor beállítja a „Megszakítás engedélyezett” bitet. Ez megszakítás kérést eredményez. A megszakítás bekövetkezésekor ez a bit törlődik.
  A megszakítás kezelő újabb feladatot ad az eszköznek (a következő karakter kiírását) és visszatér a megszakításból. Ez mindaddig folytatódik, amíg a teljes feladat el nem készült.

Az **I/O** művelet végrehajtása közben a központi egység más feladatot végezhet.

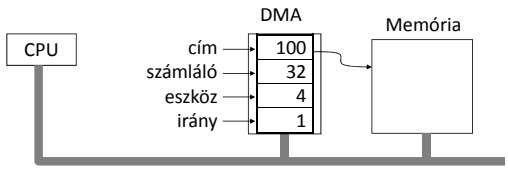
Pl.: Egy sornyi karakter képernyőre írása a terminálon:

Előkészítés: Egy rendszerprogram összegyűjti a kiírandó karaktereket egy pufferben, beállít egy globális változót, hogy mutasson a puffer elejére, egy másik globális változóban megadja a karakterek számát. Megnézi, hogy a terminál tud-e adatot fogadni (**5.31. ábra**), és ha igen, akkor elindítja az első karakter kiíratását. Ekkor a **CPU** fölszabadul egy másik program futtatására.

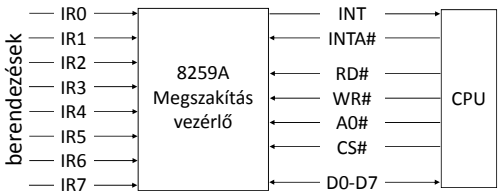
A terminál a képernyőre írja a karaktert, és **megszakítást kezdeményez**. A megszakítás kezelő újabb karakter kiírását kezdeményezi . . .

- **DMA** (Direct Memory Access, **5.33. ábra**)
  Egy tömb kiírása/beolvasása során nagyon sokszor következne be megszakítás, és mindannyiszor le kellene fusson a megszakítás kezelő eljárás. Jobb megoldást kínál a **CPU**-nál lényegesen egyszerűbb **DMA**.
  A **DMA** önállóan végzi az eszköz figyelését és az adatok mozgatását.

Cikluslopás. Ritkábban következik be megszakítás.



Megszakítás kezelés (3.43. ábra)



IRi →, **INT** →, ha **CPU** tudja fogadni, akkor **INTA#** →, **i** → **D0-D7**, a **CPU** megszakításvektor táblázat **i** -edik eleméből tudja a megszakítást kiszolgáló eljárás kezdőcímét, létrejön a megszakítás …

Nyolcnál több eszköz kiszolgálásához több megszakítás vezérlő kapcsolható össze.

Megszakítás

Hardver tevékenységek (3.42. ábra):

1. Az eszköz vezérlő megszakítás jelet tesz a sínre,
2. ha a **CPU** fogadni tudja a megszakítást, nyugtázza,
3. az eszköz vezérlője az eszköz azonosítószámát (megszakítás-vektor) elküldi a sínen,
4. ezt a **CPU** átmenetileg tárolja,
5. a **CPU** a verembe teszi az utasításszámláló aktuális értékét és a **PSW**-t,
6. a **CPU** az azonosító indexű megszakítás kezelő címét teszi az utasításszámlálóba és gyakran betölti vagy módosítja **PSW**-t.

Szoftver tevékenységek (terminálra íráskor)**:**

7. menti a használni kívánt regisztereket,
8. kiolvassa egy eszközregiszterből a terminál számát,
9. beolvassa az állapotkódot,
10. ha **B/K** hiba történt, itt lehet kezelni,
11. aktualizálja a mutatót és a számlálót, a kimenő pufferbe írja a következő karaktert, ha van,
12. visszajelez az eszköz vezérlőnek, hogy készen van,
13. visszaállítja a mentett regisztereket,
14. visszatér a megszakításból, itt történik a **PSW** eredeti értékének visszaállítása is.

Átlátszóság: Amikor bekövetkezik egy megszakítás, akkor bizonyos utasítások végrehajtódnak, de amikor ennek vége, a **CPU** ugyanolyan állapotba kerül, mint amilyenben a megszakítás bekövetkezése előtt volt.

Csapda és megszakítás

Csapda (trap): A program által előidézett feltétel (pl. túlcsordulás, csapdát eredményező utasítás) hatására automatikus eljárás hívás. **Csapda kezelő**.

Megszakítás (interrupt): Olyan automatikus eljárás hívás, amit általában nem a futó program, hanem valamilyen **B/K** eszköz idéz elő, pl. a program utasítja a lemezegységet, hogy kezdje el az adatátvitelt, és annak végeztével megszakítást küldjön. **Megszakítás kezelő**.

A csapda a **programmal szinkronizált**,
a megszakítás nem.

A megszakító rutin megszakítható-e? Gyors periféria kiszolgálása közben megszakítás kérés, …

„Alap” állapot – „megszakítási” állapot, megszakítási állapotban nem lehet újabb megszakítás.

Hierarchia: megszakítási állapotban csak magasabb szintű ok eredményezhet megszakítást.

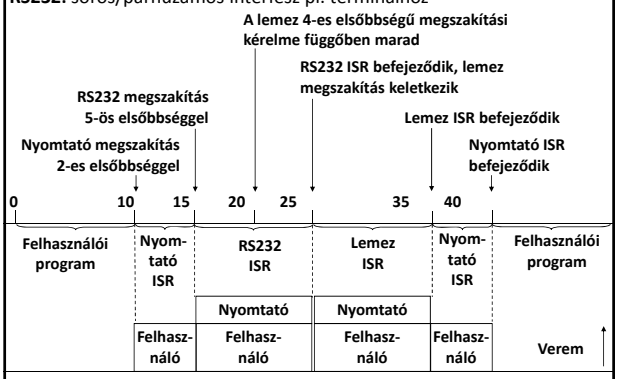
Bizonyos utasítások csak a központi egység bizonyos kitüntetett állapotában hajthatók végre, alap állapotban nem → csapda, szoftver megszakítás.

Megoldható az operációs rendszer védelme, a tár védelem stb.

A megoldás nem tökéletes: **vírus**.

ISR: Interrupt Service Routine **5.46. ábra**

RS232: soros/párhuzamos interfész pl. terminálhoz



Nem kell

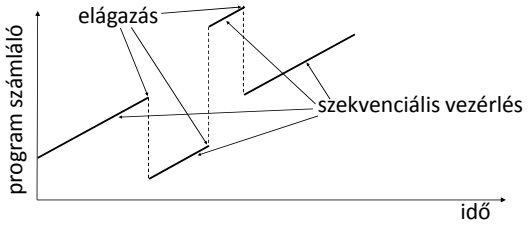
Sok esetben a programnak nincs mit tennie, amíg az I/O be nem fejeződik, pl. a klaviatúráról vár adatot. A várakozás helyett jobb megoldás, ha az operációs rendszer átmenetileg fölfüggeszti a program működését, és elindítja egy másik program futását.

Ez vezetett a **multiprogramozás** kialakulásához.

Vezérlési folyamat

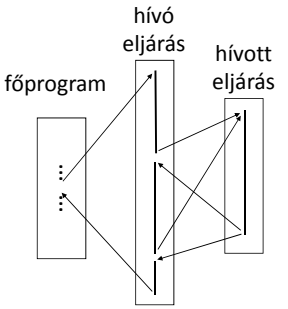
Szekvenciális vezérlés: Az utasítások abban a sorrendben kerülnek végrehajtásra, ahogy a memóriában elhelyezkednek.

Elágazás: **5.39. ábra**.



Eljárás (**5.44. ábra**):
Az eljáráshívás úgy tekinthető, mint egy magasabb szinten definiált utasítás végrehajtása: sokszor elég, ha azt tudjuk, mit csinál az eljárás, nem lényeges, hogy hogyan.

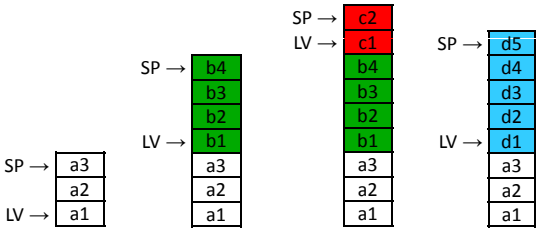
Rekurzív eljárás: önmagát közvetlenül vagy közvetve hívó eljárás.



Eljárás: paraméterek, munka terület.

A hívó és hívott eljárás paraméterei, változói nem lehetnek azonos területen: **lokális változók**.

Verem (stack): **LV** (Local Variable),
SP (Stack Pointer) verem mutató (**4.8. ábra**).



Rekurzív és re-entrant eljárások

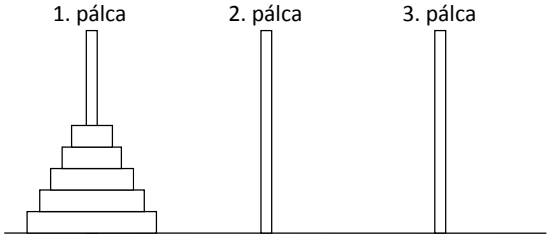
Egy eljárás **rekurzív**, ha önmagát hívja közvetlenül, vagy más eljárásokon keresztül.

Egy eljárás **re-entrant**, ha többszöri belépést tesz lehetővé, ami azt jelenti, hogy az eljárás még nem fejeződött be, amikor újra felhívható. A rekurzív eljárással szemben a különbség az, hogy a rekurzív eljárásban „programozott”, hogy mikor történik az eljárás újra hívása, re-entrant eljárás esetén az esetleges újra hívás ideje a véletlentől függ. Ez utóbbi esetben az biztosítja, hogy a munkaterületek ne keveredjenek össze, hogy újabb belépés csak másik processzusból képzelhető el, és minden processzus saját vermet használ.

Rekurzív és re-entrant eljárások

Ha egy eljárásunk készítésekor betartjuk, hogy az eljárás a paramétereit a vermen keresztül kapja, kilépéskor visszaállítja a belépéskori regiszter tartalmakat – az esetleg eredményt tartalmazó regiszterek kivételével –, továbbá a fenti módon kialakított munkaterületet használ, akkor az eljárásunk rekurzív is lehet, és a többszöri belépést is lehetővé teszi (re-entrant).

Hanoi tornyai (5.40-42. ábra)



Feladat: tegyük át az összes korongot az 1. pálcáról a 2-ra úgy, hogy egyszerre csak egy korongot mozgathatunk, és kisebb korongra nem tehetünk nagyobbat

Hanoi tornyai (5.40-42. ábra)

Rekurzív eljárás, amely **n** korongot mozgat **i**-ről **j**-re:

```
public void towers (int n, int i, int j) {
  int k;

  if(n==1)
     System.out.println(
          "Korong mozgatása:"+i+"-ről"+j"-re");
  else {
    k=6-i-j;
    towers(n-1, i, k);
    towers(1, i, j);
    towers(n-1, k, j);
  }
}
```


~ 5.43. ábra A verem tartalma az eljárásban:

towers(n, i, j) ; c konvenció V: visszatérési cím, F: régi FP

Hívások:		j i n k	j i n k	j i n k
towers(3,1,3)		**3, 1, 3, V, F, k**		
towers(2,1,2)		**3, 1, 3, V, F, 2,**	**2, 1, 2, V, F, k**	
towers(1,1,3)	**1 → 3**	3, 1, 3, V, F, 2,	2, 1, 2, V, F, 3,	3, 1, 1, V, F, k
visszatérés után		3, 1, 3, V, F, 2,	2, 1, 2, V, F, 3	
towers(1,1,2)	**1 → 2**	3, 1, 3, V, F, 2,	2, 1, 2, V, F, 3,	2, 1, 1, V, F, k
towers(1,3,2)	**3 → 2**	3, 1, 3, V, F, 2,	2, 1, 2, V, F, 3,	2, 3, 1, V, F, k
towers(1,1,3)	**1 → 3**	**3, 1, 3, V, F, 2,**	**3, 1, 1, V, F, k**	
towers(2,2,3)		**3, 1, 3, V, F, 2,**	**3, 2, 2, V, F, k**	
towers(1,2,1)	**2 → 1**	3, 1, 3, V, F, 2,	3, 2, 2, V, F, 1,	1, 2, 1, V, F, k
towers(1,2,3)	**2 → 3**	3, 1, 3, V, F, 2,	3, 2, 2, V, F, 1,	3, 2, 1, V, F, k
towers(1,1,3)	**1 → 3**	3, 1, 3, V, F, 2,	3, 2, 2, V, F, 1,	3, 1, 1, V, F, k

~ 5.43. ábra A verem tartalma az eljárásban:

V: visszatérési cím, F: régi FP

Hívások:	j i n k	j i n k	j i n k
towers(3,1,3)	3, 1, 3, V, F, k		

~ 5.43. ábra A verem tartalma az eljárásban:

V: visszatérési cím, F: régi FP

Hívások:	j i n k	j i n k	j i n k
towers(3,1,3)	3, 1, 3, V, F, k		
towers(2,1,2)	3, 1, 3, V, F, 2,	2, 1, 2, V, F, k	

~ 5.43. ábra A verem tartalma az eljárásban:

V: visszatérési cím, F: régi FP

Hívások:	j i n k	j i n k	j i n k
towers(3,1,3)	3, 1, 3, V, F, k		
towers(2,1,2)	3, 1, 3, V, F, 2,	2, 1, 2, V, F, k	
towers(1,1,3) **1 → 3**	3, 1, 3, V, F, 2,	2, 1, 2, V, F, 3,	3, 1, 1, V, F, k

~ 5.43. ábra A verem tartalma az eljárásban:

V: visszatérési cím, F: régi FP

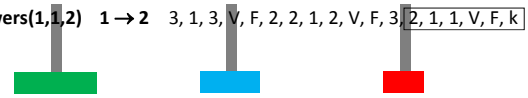
Hívások:	j i n	k | j i n	k | j i n	k
towers(3,1,3)	3, 1, 3, V, F, k			
towers(2,1,2)	3, 1, 3, V, F, 2, 2, 1, 2, V, F, k			
towers(1,1,3) 1 → 3	3, 1, 3, V, F, 2, 2, 1, 2, V, F, 3, 3, 1, 1, V, F, k			
visszatérés után	3, 1, 3, V, F, 2, 2, 1, 2, V, F, 3			

Máté: Architektúrák **Gyakorlásra** 11. előadás 43

~ 5.43. ábra A verem tartalma az eljárásban:

V: visszatérési cím, F: régi FP

Hívások:	j i n	k | j i n	k | j i n	k
towers(3,1,3)	3, 1, 3, V, F, k			
towers(2,1,2)	3, 1, 3, V, F, 2, 2, 1, 2, V, F, k			
towers(1,1,3) 1 → 3	3, 1, 3, V, F, 2, 2, 1, 2, V, F, 3, 3, 1, 1, V, F, k			
visszatérés után	3, 1, 3, V, F, 2, 2, 1, 2, V, F, 3			
towers(1,1,2) 1 → 2	3, 1, 3, V, F, 2, 2, 1, 2, V, F, 3, 2, 1, 1, V, F, k			

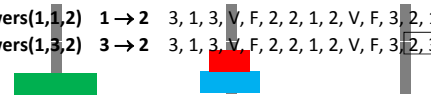


Máté: Architektúrák **Gyakorlásra** 11. előadás 44

~ 5.43. ábra A verem tartalma az eljárásban:

V: visszatérési cím, F: régi FP

Hívások:	j i n	k | j i n	k | j i n	k
towers(3,1,3)	3, 1, 3, V, F, k			
towers(2,1,2)	3, 1, 3, V, F, 2, 2, 1, 2, V, F, k			
towers(1,1,3) 1 → 3	3, 1, 3, V, F, 2, 2, 1, 2, V, F, 3, 3, 1, 1, V, F, k			
visszatérés után	3, 1, 3, V, F, 2, 2, 1, 2, V, F, 3			
towers(1,1,2) 1 → 2	3, 1, 3, V, F, 2, 2, 1, 2, V, F, 3, 2, 1, 1, V, F, k			
towers(1,3,2) 3 → 2	3, 1, 3, V, F, 2, 2, 1, 2, V, F, 3, 2, 3, 1, V, F, k			

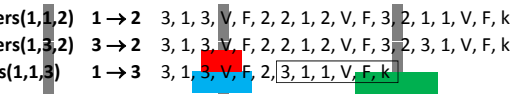


Máté: Architektúrák **Gyakorlásra** 11. előadás 45

~ 5.43. ábra A verem tartalma az eljárásban:

V: visszatérési cím, F: régi FP

Hívások:	j i n	k | j i n	k | j i n	k
towers(3,1,3)	3, 1, 3, V, F, k			
towers(2,1,2)	3, 1, 3, V, F, 2, 2, 1, 2, V, F, k			
towers(1,1,3) 1 → 3	3, 1, 3, V, F, 2, 2, 1, 2, V, F, 3, 3, 1, 1, V, F, k			
visszatérés után	3, 1, 3, V, F, 2, 2, 1, 2, V, F, 3			
towers(1,1,2) 1 → 2	3, 1, 3, V, F, 2, 2, 1, 2, V, F, 3, 2, 1, 1, V, F, k			
towers(1,3,2) 3 → 2	3, 1, 3, V, F, 2, 2, 1, 2, V, F, 3, 2, 3, 1, V, F, k			
towers(1,1,3) 1 → 3	3, 1, 3, V, F, 2, 3, 1, 1, V, F, k			

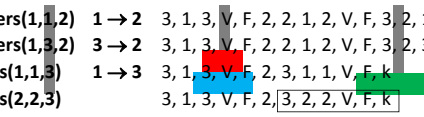


Máté: Architektúrák **Gyakorlásra** 11. előadás 46

~ 5.43. ábra A verem tartalma az eljárásban:

V: visszatérési cím, F: régi FP

Hívások:	j i n	k | j i n	k | j i n	k
towers(3,1,3)	3, 1, 3, V, F, k			
towers(2,1,2)	3, 1, 3, V, F, 2, 2, 1, 2, V, F, k			
towers(1,1,3) 1 → 3	3, 1, 3, V, F, 2, 2, 1, 2, V, F, 3, 3, 1, 1, V, F, k			
visszatérés után	3, 1, 3, V, F, 2, 2, 1, 2, V, F, 3			
towers(1,1,2) 1 → 2	3, 1, 3, V, F, 2, 2, 1, 2, V, F, 3, 2, 1, 1, V, F, k			
towers(1,3,2) 3 → 2	3, 1, 3, V, F, 2, 2, 1, 2, V, F, 3, 2, 3, 1, V, F, k			
towers(1,1,3) 1 → 3	3, 1, 3, V, F, 2, 3, 1, 1, V, F, k			
towers(2,2,3)	3, 1, 3, V, F, 2, 3, 2, 2, V, F, k			

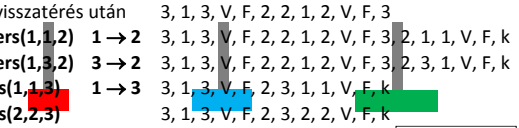


Máté: Architektúrák **Gyakorlásra** 11. előadás 47

~ 5.43. ábra A verem tartalma az eljárásban:

V: visszatérési cím, F: régi FP

Hívások:	j i n	k | j i n	k | j i n	k
towers(3,1,3)	3, 1, 3, V, F, k			
towers(2,1,2)	3, 1, 3, V, F, 2, 2, 1, 2, V, F, k			
towers(1,1,3) 1 → 3	3, 1, 3, V, F, 2, 2, 1, 2, V, F, 3, 3, 1, 1, V, F, k			
visszatérés után	3, 1, 3, V, F, 2, 2, 1, 2, V, F, 3			
towers(1,1,2) 1 → 2	3, 1, 3, V, F, 2, 2, 1, 2, V, F, 3, 2, 1, 1, V, F, k			
towers(1,3,2) 3 → 2	3, 1, 3, V, F, 2, 2, 1, 2, V, F, 3, 2, 3, 1, V, F, k			
towers(1,1,3) 1 → 3	3, 1, 3, V, F, 2, 3, 1, 1, V, F, k			
towers(2,2,3)	3, 1, 3, V, F, 2, 3, 2, 2, V, F, k			
towers(1,2,1) 2 → 1	3, 1, 3, V, F, 2, 3, 2, 2, V, F, 1, 1, 2, 1, V, F, k			



Máté: Architektúrák **Gyakorlásra** 11. előadás 48

~ 5.43. ábra A verem tartalma az eljárásban:

V: visszatérési cím, F: régi FP

Hívások:	j i n	k | j i n	k | j i n	k
towers(3,1,3)	3, 1, 3, V, F, k			
towers(2,1,2)	3, 1, 3, V, F, 2, 2, 1, 2, V, F, k			
towers(1,1,3) 1 → 3	3, 1, 3, V, F, 2, 2, 1, 2, V, F, 3, 3, 1, 1, V, F, k			
visszatérés után	3, 1, 3, V, F, 2, 2, 1, 2, V, F, 3			
towers(1,1,2) 1 → 2	3, 1, 3, V, F, 2, 2, 1, 2, V, F, 3, 2, 1, 1, V, F, k			
towers(1,3,2) 3 → 2	3, 1, 3, V, F, 2, 2, 1, 2, V, F, 3, 2, 3, 1, V, F, k			
towers(1,1,3) 1 → 3	3, 1, 3, V, F, 2, 3, 1, 1, V, F, k			
towers(2,2,3)	3, 1, 3, V, F, 2, 3, 2, 2, V, F, k			
towers(1,2,1) 2 → 1	3, 1, 3, V, F, 2, 3, 2, 2, V, F, 1, 1, 2, 1, V, F, k			
towers(1,2,3) 2 → 3	3, 1, 3, V, F, 2, 3, 2, 2, V, F, 1, 3, 2, 1, V, F, k			

~ 5.43. ábra A verem tartalma az eljárásban:

V: visszatérési cím, F: régi FP

Hívások:	j i n	k | j i n	k | j i n	k
towers(3,1,3)	3, 1, 3, V, F, k			
towers(2,1,2)	3, 1, 3, V, F, 2, 2, 1, 2, V, F, k			
towers(1,1,3) 1 → 3	3, 1, 3, V, F, 2, 2, 1, 2, V, F, 3, 3, 1, 1, V, F, k			
visszatérés után	3, 1, 3, V, F, 2, 2, 1, 2, V, F, 3			
towers(1,1,2) 1 → 2	3, 1, 3, V, F, 2, 2, 1, 2, V, F, 3, 2, 1, 1, V, F, k			
towers(1,3,2) 3 → 2	3, 1, 3, V, F, 2, 2, 1, 2, V, F, 3, 2, 3, 1, V, F, k			
towers(1,1,3) 1 → 3	3, 1, 3, V, F, 2, 3, 1, 1, V, F, k			
towers(2,2,3)	3, 1, 3, V, F, 2, 3, 2, 2, V, F, k			
towers(1,2,1) 2 → 1	3, 1, 3, V, F, 2, 3, 2, 2, V, F, 1, 1, 2, 1, V, F, k			
towers(1,2,3) 2 → 3	3, 1, 3, V, F, 2, 3, 2, 2, V, F, 1, 3, 2, 1, V, F, k			
towers(1,1,3) 1 → 3	3, 1, 3, V, F, 2, 3, 2, 2, V, F, 1, 3, 1, 1, V, F, k			

Visszatérések után:

~ 5.43. ábra A verem tartalma az eljárásban:

V: visszatérési cím, F: régi FP

Hívások:	j i n	k | j i n	k | j i n	k
towers(3,1,3)	3, 1, 3, V, F, k			
towers(2,1,2)	3, 1, 3, V, F, 2, 2, 1, 2, V, F, k			
towers(1,1,3) 1 → 3	3, 1, 3, V, F, 2, 2, 1, 2, V, F, 3, 3, 1, 1, V, F, k			
visszatérés után	3, 1, 3, V, F, 2, 2, 1, 2, V, F, 3			
towers(1,1,2) 1 → 2	3, 1, 3, V, F, 2, 2, 1, 2, V, F, 3, 2, 1, 1, V, F, k			
towers(1,3,2) 3 → 2	3, 1, 3, V, F, 2, 2, 1, 2, V, F, 3, 2, 3, 1, V, F, k			
towers(1,1,3) 1 → 3	3, 1, 3, V, F, 2, 3, 1, 1, V, F, k			
towers(2,2,3)	3, 1, 3, V, F, 2, 3, 2, 2, V, F, k			
towers(1,2,1) 2 → 1	3, 1, 3, V, F, 2, 3, 2, 2, V, F, 1, 1, 2, 1, V, F, k			
towers(1,2,3) 2 → 3	3, 1, 3, V, F, 2, 3, 2, 2, V, F, 1, 3, 2, 1, V, F, k			
towers(1,1,3) 1 → 3	3, 1, 3, V, F, 2, 3, 2, 2, V, F, 1, 3, 1, 1, V, F, k			

Visszatérések után:

Rekurzív eljárások megvalósításához veremre van szükség. Minden hívás esetén az eljárás paramétereit a verembe kell tenni, és ott kell elhelyezni a lokális változókat is!

Eljárás prológus: a régi verem keret mutató (**FP**) elmentése, új verem keret mutató megadása, és a verem mutató (**SP**) növelése, hogy legyen hely a veremben a lokális változók számára.

Eljárás epilógus: visszatéréskor a verem kitakarítása.

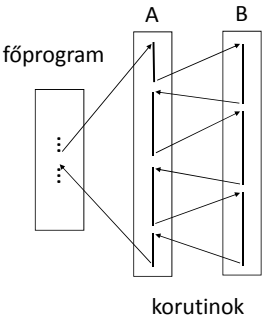
A Hanoi tornyai probléma megoldása

5.47. ábra: **Pentium 4** program.

5.48. ábra: **UltraSPARC III** program.
Eltolás rés!

Vezérlési folyamat

- **Szekvenciális vezérlés (5.39. ábra)**
- **Elágazás.**
- **Eljárás: 5.44. ábra.**
- **Megszakítások.**
- **Csapdák.**
- **Korutinok: 5.45. ábra.** Párhuzamos feldolgozás szimulálására alkalmas egy **CPU**-s gépen.



goto helyett jobb a ciklus vagy az eljárás alkalmazása!

A Pentium 4 utasításai

- Egész utasítások legnagyobb része: **5.34. ábra**.
- Egyéb utasítások (pl. lebegőpontosak).

Az UltraSPARC III utasításai

Összes egész utasítás: **5.35. ábra**.
A utasításnévben **CC**: beállítja a feltételkódot.
ADD, **ADDC**, **ADDCC**, **ADDCCC** utasítások.
Szimulált utasítások (**5.36. ábra**), pl.:

```
MOV SRC,DST  ≡  OR SRC,G0,DST
```

A 8051 utasításai (5.37. ábra)

Bit utasítások, pl. a 43. bit 1-re állítása:

```
SETB 43
```


Feladatok

Mi az ortogonalitási elv?
Milyen utasítás formájú 3 címes gépet tervezne?
Milyen utasítás formájú 2 címes gépet tervezne?
Teljesül-e az ortogonalitási elv a **Pentium 4**-en?
Milyen utasítás formái vannak a **Pentium 4**-nek?
Mire szolgál a prefix bájt a **Pentium 4**-en?
Mire szolgál a címzési mód bájt a **Pentium 4**-en?
Mire szolgál a **SIB** bájt a **Pentium 4**-en?

Feladatok

Teljesül-e az ortogonalitási elv az **UltraSPARC III**-on?
Milyen utasítás formái vannak az **UltraSPARC III**-nak?
Mi a különbség az **UltraSPARC III** **ADD**, **ADDC**, **ADDCC** és **ADDCCC** utasításai között?
Milyen formátumú **LOAD** utasításai vannak az **UltraSPARC III**-nak?
Hogy adható meg 32 bites közvetlen adat az **UltraSPARC III**-on?
Milyen formátumú **CALL** utasítása van az **UltraSPARC III**-nak?

Feladatok

Teljesül-e az ortogonalitási elv a **8051**-en?
Milyen utasítás formái vannak a **8051**-nek?
Milyen formátumú ugró utasításai vannak a **8051**-nek?
Hogy érhető el 256-nál magasabb memória cím a **8051**-en?
Milyen utasítás típusokat ismer?
Mondjon diadikus/monadikus utasításokat!
Hogy néz ki a vég-/kezdőfeltételes ciklus?

Feladatok

Milyen vezérlő regiszterei vannak egy terminálnak?
Mire szolgál a terminál billentyűzet puffer regisztere?
Hogyan történik a programozott **B/K**?
Mit nevezünk pollozásos technikának?
Mire használható a **DMA**?
Hogy működik a **DMA**?
Milyen regiszterei vannak a **DMA**-nak?

Feladatok

Mit nevezünk program megszakításnak?
Mi a megszakítás kezelő?
Hogyan történhet a nyomtatás szervezése megszakítás segítségével?
Megszakítható-e a megszakítás kezelő?
Mi a csapda?
Mi a különbség a csapda és a megszakítás között?
Hogy működik a 8259A megszakítás vezérlő lapka?
Milyen hardver és milyen szoftver tevékenységek tartoznak a megszakításhoz?
Mit jelent az átlátszóság megszakítás esetén?

Feladatok

Mi a szekvenciális vezérlés?

Mi az eljárás?

Mi a lokális adat terület?

Hogy alakíthatunk ki lokális adat területet?

Mi a rekurzív eljárás?

Mi a re-entrant eljárás?

Mondjon példát rekurzív eljárással megoldható problémára!

Hogy oldható meg a Hanoi tornyai probléma?

Feladatok

Mi az eljárás prológus?

Mi az eljárás epilógus?

Mit nevezünk korutinnak (társrutin, coroutine)?

Mi az eltolás rés?

Mi a különbség az **UltraSPARC III** `ADD`, `ADDC`, `ADDCC` és `ADDCCC` utasításai között?

Az előadáshoz kapcsolódó

Fontosabb témák

A Pentium 4, az UltraSPARC III, I-8051 utasítás formátumai, címzési módjai

Utasítás típusok.

Ciklus szervezés.

Programozott és megszakítás vezérelt I/O. DMA

Vezérlési folyamat. Szekvenciális vezérlés, elágazás, ciklus szervezés, eljárás, rekurzív és re-entrant eljárás, megszakítás, csapda. Korutinok.