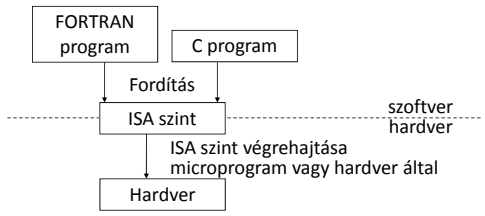
Utasításrendszer-architektúra szintje (ISA)

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

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



Általában a mikroarchitektúra nem tartozik hozzá.

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.
- **…**

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

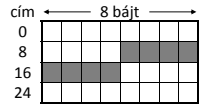
Néha (pl. **I-8051**) külön memória van az adatoknak és az utasításoknak (Harvard-architektúra, nem ugyanaz, mint az osztott gyorsítótár!).

Memória modellek

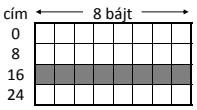
ASCII kód 7 bit + paritás → **Byte** (**bájt**)

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

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 elől (kompatibilitás!), a fontosabbak:

- **4004**: 4 bites,
- **8080**: 8 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 ha abban hiba történik, akkor nem fagy le, hanem visszaadja a vezérlést a **WINDOWS**-nak).

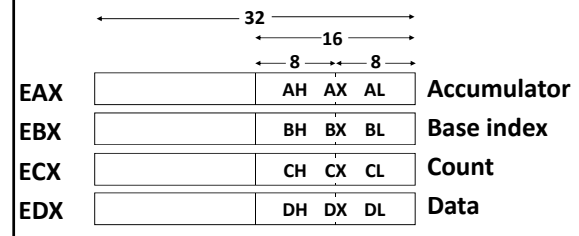
védett: valódi **Pentium 4**. 4 védelmi szint (**PSW**):
0: kernelmód (operációs r.), **1**, **2**: ritkán használt,
3: felhasználói mód.

Memóriaszervezés:

- **16 K db szegmens** lehetséges, de a **WINDOWS**-ok és **UNIX** 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),
- **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.

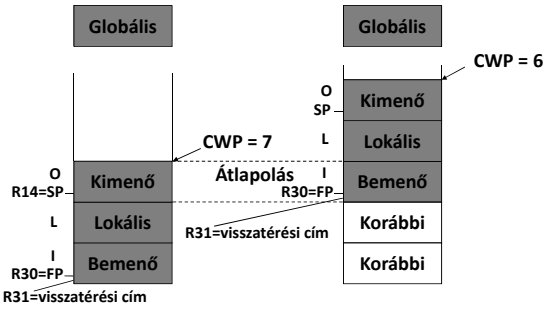
Regiszterek:

- **32 általános** (**5.4. ábra**) **64 bites**, a használatuk részben konvenció, 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**) Bemenő paraméterek, de
  R30 (**I6**) **= FP** az aktuális veremkeret mutatója,
  R31: visszatérési cím.

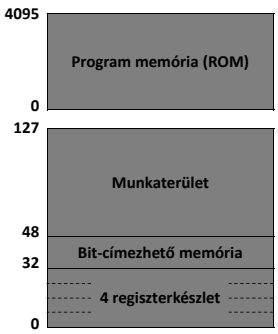
CWP (Current Window Pointer, **5.5. ábra**) mutatja az aktuális **regiszter ablak**ot (több regiszter készlet létezik, de mindig csak egy látszik). Ha kifogy a regiszter készlet, memóriába mentés, …



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

Külön címtartományú program és adat memória.



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

8 regiszter: **R0**, … , **R7**. A regiszterek a memóriában vannak. 4 regiszter készlete van, de egyszerre csak egy használható.

Az ábra ajánlott

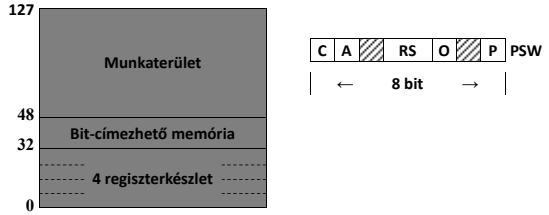
5.6. ábra. A **8051** memória szervezése, fő regiszterei

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

A **PSW** regiszter **RS** mezeje mondja meg, hogy melyik regiszterkészlet az aktuális.

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

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



Máté: Architektúrák **Az ábra ajánlott** 10. előadás 19

IE (Interrupt Enable):

EA= 1: nincs tiltva a megszakítás, 0: mind tiltva van,

ES=1: megszakítás engedélyezve a soros vonalon, 0: tiltva

E0-**2**=1: a 0-2 időzítő csatorna engedélyezve, 0: tiltva.

Az engedélyezett számlálók egyszerre futhatnak, és ezek megszakítást válthatnak ki.

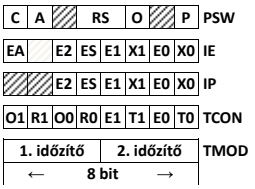
X0-**1**=1: külső eszköz megszakítás engedélyezve, 0: tiltva

IP (Interrupt Priority):

0 (alacsony),

1 (magas).

Az alacsonyabb szintű megszakítást megszakíthatja egy magasabb szintű.



Máté: Architektúrák **Az ábra ajánlott** 10. előadás 20

TCON: a 0. és 1. időzítőt vezérli (ezek a fő időzítők).

O0-**1**: beáll az időzítő túlcsordulásakor.

R0-**1**: ezzel ki- és bekapcsolható az időzítő futása.

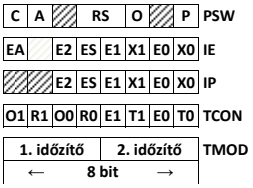
A többi bit az időzítő él- vagy szintvezérlésével kapcsolatos.

TMOD: a fő időzítők üzemmódját határozza meg

8, 13 vagy 16 bites,

valódi időzítő vagy számláló,

hardver jelek szintje.



Máté: Architektúrák **Az ábra ajánlott** 10. előadás 21

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

Pl. **ACC** a 240-en.

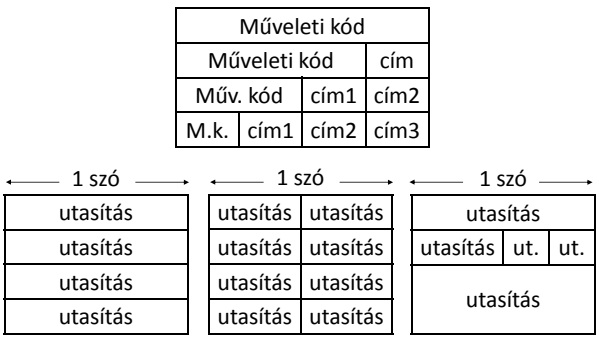
A 8052 valódi memóriát tartalmaz a 128-255 tartományban, a speciális regiszterek címe átfed a memóriával.

– Direkt címzéssel a speciális regisztereket,

– Indirekt címzéssel a **RAM**-ot érhetjük el.

Máté: Architektúrák 10. előadás 22

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



Máté: Architektúrák 10. előadás 23

Az utasítás hossza

Lehetőségek:

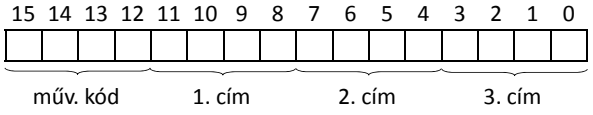
- fix utasításhossz:
  rövidebb kód mellett hosszabb operandus rész,
- minimális átlagos utasításhossz:
  a gyakori kódok rövidek, a ritkán használtak hosszabbak.

Máté: Architektúrák 10. előadás 24

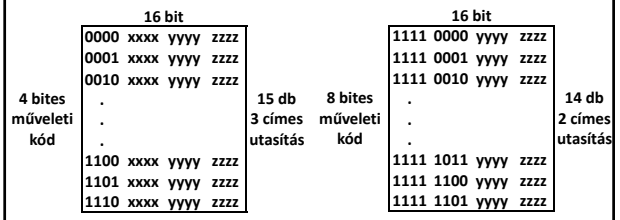
A műveleti kód kiterjesztése

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**).



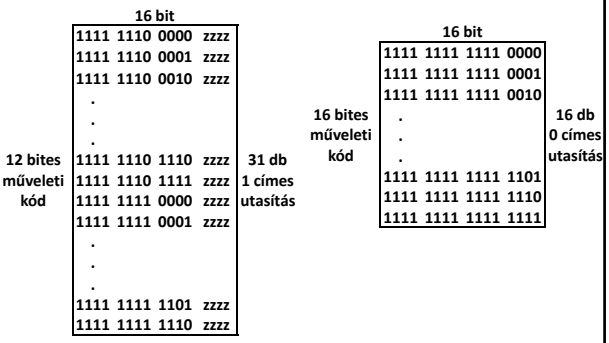
A műveleti kód kiterjesztése (5.13. ábra)



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.

Minden utasítás tartalmaz műveleti kódot.
Ezen kívül tartalmazhat az operandusokra, eredményre vonatkozó információt.

Utasítás típusok:

- regiszter-memória utasítások: a regiszterek és a memória közötti adatforgalom (betöltés, tárolás). Ilyenkor egy regiszter és egy memória cím megadása szükséges a címrészen.
- regiszter-regiszter utasítások: összeadás, kivonás, … Az eredmény is regiszterben keletkezik. Ilyenkor három regiszter megadása szükséges a címrészen.
- …

Címzési módszerek

Három cím: *cél = forrás1 + forrás2*.

A memória sok rekeszt tartalmaz, de csak kevés regiszter van. Egy regiszter néhány bittel címezhető.

Regiszterek használata rövidíti a címeket, de nyújtja a programot, ha az operandus csak egyszer kell.

A legtöbb operandust többször használjuk.

Implicit operandusok:

- **Két cím**: ***regiszter2*** = *regiszter2 + forrás1*.
- **Egy cím**: ***akkumulátor*** = ***akkumulátor*** + *forrás1*.
- **Nulla cím**: verem, pl. az **IJVM IADD** utasítása.

$e = a_1 + a_2 + \ldots + a_n$ kiszámítása

3, 2 és 1 címes gépen

lépés	3 címes	2 címes	1 címes
1.	$e = a_1 + a_2$	$e = a_1$	$A = a_1$
2.	$e = e + a_3$	$e = e + a_2$	$A = A + a_2$
…	…	…	…
n-2.	$e = e + a_{n-1}$	$e = e + a_{n-2}$	$A = A + a_{n-2}$
n-1.	$e = e + a_n$	$e = e + a_{n-1}$	$A = A + a_{n-1}$
n.	kész	$e = e + a_n$	$A = A + a_n$
n+1.		kész	$e = A$
			kész

Operandus megadás

- **Közvetlen operandus** (immediate operand): Az operandus megadása az utasításban (**5.17. ábra**)

MOV	R1	#4

- **Direkt címzés** (direct addressing): A memóriacím megadása a címrészen. Az utasítás mindig ugyanazt a címet használja. Az operandus értéke változhat, de a címe nem (fordításkor ismert kell legyen!).
- **Regiszter címzés** (register addressing): Mint a direkt címzés, csak nem memóriát, hanem regisztert címez.

Egy tömb elemeinek összeadásához $\mathbf{e = e + a_k}$ vagy $\mathbf{A = A + a_k}$ alakú utasításokra van szükség.

- Vagy mindegyik utasítást szerepeltetjük a programban,
- vagy minden elem hozzáadása után úgy **módosítjuk az összeadó utasítást**, hogy legközelebb a következő elem hozzáadását végezze …

Önmódosító program (Neumann János ötlete).
Ma már kerülendő (cache problémák!),
pl. regiszter-indirekt címzéssel kikerülhetjük.

- **Regiszter-indirekt címzés** (register indirect addressing):
  A címrészen valamelyik regisztert adjuk meg, de a megadott regiszter nem az operandust tartalmazza, hanem azt a **memóriacímet**, ahol az operandus található (**mutató - pointer**).

  Rövidebb, és a regiszter értékének módosításával a cím változtatható.

Pl.: A 100 szóból álló **A** tömb elemeinek összeadása két címes gépen (egy elem 4 bájt), ~ **5.18. ábra.**

```
        MOV  R1, #0        ; gyűjtsük az eredményt R1-ben,
                           ; kezdetben ez legyen 0.
        MOV  R2, #A        ; R2-be töltjük az A tömb címét
        MOV  R3, #A + 400; a tömb utáni első cím
C:      ADD  R1, (R2)      ; regiszter-indirekt címzés a tömb
                           ; aktuális elemének elérésére
        ADD  R2, #4        ; R2 tartalmát növeljük 4-gyel
        CMP  R2, R3        ; végeztünk?
        BLT  C             ; ha nem, ugrás a C címkéhez
        . . .              ; kész az összegzés
```


- **Indexelt címzés** (indexed addressing): Egy eltolási érték (offset) és egy (index) regiszter tartalmának összege lesz az operandus címe, **~5.19-20. ábra**.

```
        MOV  R1, #0        ; gyűjtsük az eredményt R1-ben,
                           ; kezdetben ez legyen 0.
        MOV  R2, #0        ; az index kezdő értéke
        MOV  R3, #400      ; a tömb mögé mutató index
C:      ADD  R1, A(R2)     ; indexelt címzés a tömb
                           ; aktuális elemének elérésére
        ADD  R2, #4        ; R2 tartalmát növeljük 4-gyel
        CMP  R2, R3        ; végeztünk?
        BLT  C             ; ha nem, ugrás a C címkéhez
        . . .              ; kész az összegzés
```


- **Bázisindex címzés** (based-indexed addressing): Egy eltolási érték (offset) és két (egy bázis és egy index) regiszter tartalmának összege lesz az operandus címe. Ha **R5** **A** címét tartalmazza, akkor

```
C:      ADD  R1, A(R2)
```

helyett a

```
C:      ADD  R1, (R2+R5)
```

utasítás is írható.

Ez a módszer előnyös, ha nem csak az **A** tömb elemeit szeretnénk összegezni.

- **Verem címzés** (stack addressing): Az operandus a verem tetején van. Nem kell operandust megadni az utasításban.

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:

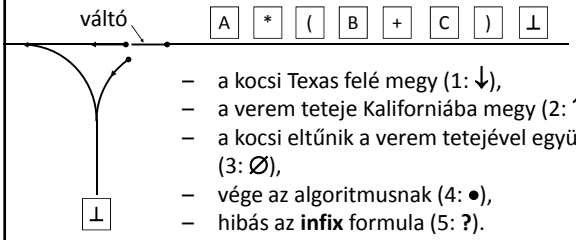
infix: *x* + *y*, **postfix**: *x y* +.

Előnyei: nem kell zárójel, sem precedencia szabályok. Jól alkalmazható verem címzés 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: ↓),
- a verem teteje Kaliforniába megy (2: ↑),
- a kocsi eltűnik a verem tetejével együtt (3: ∅),
- vége az algoritmusnak (4: •),
- hibás az **infix** formula (5: **?**).

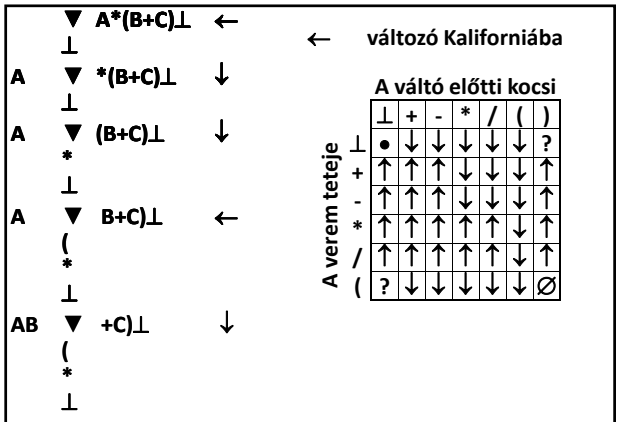
Minden változó és konstans menjen Kaliforniába (←), a többi esetben a döntési tábla szerint járjunk el (**5.21. ábra**):

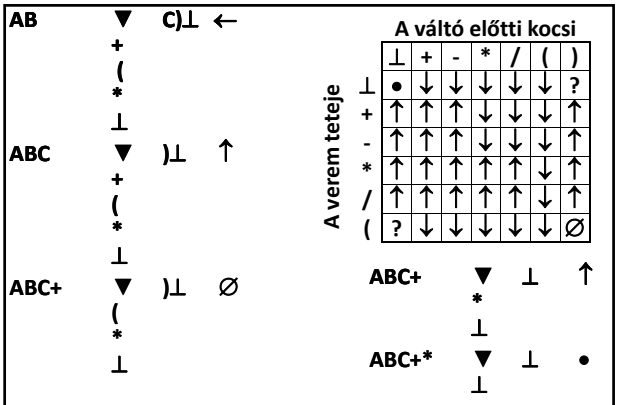
A váltó előtti kocsi

A verem teteje	⊥	+	-	*	/	(	)
⊥	•	↓	↓	↓	↓	↓	?
+	↑	↑	↑	↓	↓	↓	↑
-	↑	↑	↑	↓	↓	↓	↑
*	↑	↑	↑	↑	↑	↓	↑
/	↑	↑	↑	↑	↑	↓	↑
(	?	↓	↓	↓	↓	↓	∅

←	változó Kaliforniába
↓	New Yorkból Texasba
↑	Texasból Kaliforniába
∅	Törlődjön a következő és az utolsó texasi kocsi
•	„ Kaliforniában" kész a postfix forma
?	Hibás az infix formula

A döntési tábla tartalmazza a prioritási szabályokat!





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

Feladatok

Miért kitüntetett szint a gépi utasítások szintje (**ISA**)?
Mikor jó egy gép **ISA** szintje?
Mi a különbség a felhasználói (user) és a kernel mód között?
Mit jelent a Harvard architektúra kifejezés?
Mondjon példát Harvard architektúrájú gépre!
Mit jelent az igazítás 4 bájtos szavak tárolásánál?
Mi az igazítás előnye?
Mit jelent a memória szemantika?
Milyen hardver megoldásokat ismer a memória műveletek végrehajtási sorrendjére vonatkozóan?

Feladatok

Mi a **SYNC** utasítás hatása?
Miért van szükség a **SYNC** utasításra?
Mondjon olyan regisztert, amely a mikroutasítások szintjén és **ISA** szinten is látszik!
Mondjon olyan regisztert, amely csak a mikroutasítások szintjén látszik!
Mondjon olyan regisztert, amely csak kernel módban érhető el!
Milyen utasítás típusokat ismer?

Feladatok

Melyek a **Pentium 4** processzor legfontosabb elődjei?
Milyen üzemmódjai vannak a **Pentium 4**-nek?
Milyen a **Pentium 4** memória szervezése?
Milyen regiszterei vannak a **Pentium 4**-nek?
Mit jelent a **Little endian** tárolási mód?
Mit jelent a Load/store architektúra?
Milyen az **UltraSPARC III** memória szervezése?
Milyen regiszterei vannak az **UltraSPARC III**-nak?
Mit tud az **UltraSPARC III G0** regiszteréről?
Mi a **CWP** (Current Window Pointer) szerepe?
Hogy működik az **UltraSPARC III** regiszter ablak technikája?

Feladatok

Hány regiszter készlete van a **8051**-nek?
Hol helyezkednek el a **8051** regiszterei?
Mire jó a bit-címezhető memória?
Írja le a **8051 RAM**-jának a szerkezetét!
Mire szolgál a **8051 IE**, **IP**, **TCON** és **TMOD** regisztere?
Milyen operandus megadási módokat ismer?
Mi a közvetlen operandus megadás?
Mi a direkt címzés?
Mi a regiszter címzés?
Mi a regiszter-indirekt címzés?
Mi az indexelt címzés?
Mi a bázisindex címzés?

Feladatok

Hány címes utasítások lehetségesek? Adjon mindegyikre példát!
Milyen címzési módokat ismer? Részletezze ezeket!
Mit jelent a fordított lengyel jelölés?
Milyen előnyei vannak a postfix jelölésnek?
Írja át postfix alakúra az alábbi formulákat!
A+B, **A+B+C**, **A+B*C**, **A*B+C**.
Írja át infix alakúra az alábbi postfix formulákat!
AB+, **AB–C/**, **AB*C+**, **AB*CD/E+–**.
Hogy működik Dijkstra algoritmusa?
Hogy értékelhetők ki a postfix alakú formulák?

Az előadáshoz kapcsolódó

Fontosabb témák

Gépi utasítás szint. Memória modellek, memória szemantika

A Pentium 4, az UltraSPARC III és az I-8051 regiszterei

Három, kettő egy és nulla címes utasítások

Operandus megadás módjai. Közvetlen operandus, direkt, regiszter, regiszter-indirekt, indexelt, bázis-index címzés, implicit operandus

Veremcímzés. Fordított lengyel (postfix) jelölés. Dijkstra algoritmusa. Postfix alakú formulák kiértékelése.

Máté: Architektúrák 10. előadás 49