- eltolt ábrázolás jobb, mert a bitenkénti összehasonlítás hatékonyan végbemehet.

- Másik lebegőpontos ábrázolás (Knuth)

  0.1111010 0011 ⇒ $\frac{1}{2} \leq x < 1$

  ↳ ha ezt eltoljuk ⇒ 10^{1011}-gyel van eltolva

A két ábrázolásnak nincs köze egymáshoz.

BCD szám:

(Binárisan kódolt decimális számjegyek)

- történelmi időktől kezdve tud ilyen kódolást a processzor.
- Számrendszer-átváltás osztással:

  1111'0100011 : 1010 = 1
  0101

- 1) Az osztót fellépteljük a legnagyobb bit helyére ⇒ ha sikerül: a bit 1-es, ha nem ⇒ 0, és eggyel lejjebb lépünk.

Igazából 1010000000-et kellene levonni, mert ez egy konkrét szám. Mi mégis 1010-t vonunk ki, írjuk alá a maradékot.

- Alakítsuk direkt módon 4-es csoportokra a bináris számokat, hogy a 4 bit adja a számjegyeket.

BCD aritmetika:

I. jól tudja a processzor

A + B → BCD számokat adjuk össze
K OR Z → korigáló

II.

AC

C

Közbes átvitel
↓
mondja, ha átvitel történt a BCD kódolású számok határán.

C	AC
0	0
0	1
1	0
1	1

1)
  7 7
+ 6 6

$99_{(16)} + 66_{(16)} = FF_{(16)}$

```
 1 1      1 7      7 1
 6 6      6 6      6 6
 ---      ---      ---
 7 7      7 D      D 7
```

```
 7 7
 6 6
 ---
 D D
```

2)
```
  D D        7 7        7 D        D 7
+ 4 4 11   + 4 4 0    + 4 4 01     4 4 10
  ---        ---        ---        ---
  2 1        B B        C 1        1 B
```
↓
volt közbes átvitel

3)
```
  D D        B B        C 1
  4 4      + 9 A        ---
  ---        ---        6 1
  2 1        5 5
  0 0
```

h h
1B → FA
15

összeadtunk 2 számot össze kell rakni az őket

többről később: RISC / CISC processzorok

CISC → intel processzorai voltak tervezve

Egyre több műveletet vezettek

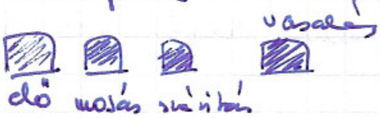
MOV ES: ~~ES~~ : [SI + BX + ~~ta~~] 127B+1

a proc. belsejében egy mikroprogram kezdi → a belső utasításokat hajtja végre.

Legeredményesebb felvezető üteme megegyezik az órajel ütemével. Gyorsítható, ha az ~~Faxit~~ ① órajelet gyorsítjuk.

① utasítás végrehajtásához kevesebb belső ciklus kell.

„Mosodai példa":



Tfh: minden művelet 1 órán át tart ⇒ 2 adag ruhát lehet kimosni.

a fázisok között átmenet kihasználásával 4X-es teljesítménye van → ne csináljunk többet.

↑
PIPELINE nevezés. proc-ban alkalmazás

Mi. feltételekkel a művelet fázisokra bontható?

Felt: 1) nagyjából ugyanannyi legyen időben a művelet-fázisok.

2) olyan eszközöket használunk, melyet már nem igényel. egymástól függetlenek.

...SC-ben a bonyolult ... ...

[3 v J, 127 MB

Reduction Instruction
(csökkentett utasításkészletű proc.)

CISC	RISC
...ett utasítások	• egyszerű utasítások
...rehajtási idő változó	• FIX végrehajtási idő
...asításhoz változó	• fix utasításvégrehajtás.
...ó kombinált és utasítás ...e lehetett	
...adott a mem-val ...bl.-beli utasítás ...hatott a tárolóból, u. ...van, regiszter.	
...ítás	• csak LOAD és STORE ut. van benne → mem. olvasás, mem. írás

...oad és store következménye:

... ... progr.-nál lokális vált.-kat használunk. ⇒ ...lett volna LOAD-dal betölteni a tárolóba és store-ral ...

...giszterkész	• nagy regiszterkész
...e egyszerű fordítóprogramok	• bonyolult fordítóprogr. (egy RISC annyira jó, amennyire a hozzá adott fordító)

a CISC-nek a bonyolult belső ciklo. orientációja

MOV [BV], 127 MB

(RISC) Reduction Instruction
(csökkentett utasításkészletű proc.)

•

CISC	RISC
• összetett utasítások	• egyszerű utasítások
• végrehajtási időváltozik	• FIX végrehajtási idő
• utasításhossz változik	• fix utasításvégrehajtás.
• tároló használat is utasítás közbe felvett	
átmeneti a mem-val a tábl.-beli utasítás módosított a tárolóból, u. baj van, regiszter.	
• utasítás	• csak LOAD és STORE ut. vannak benne. → mem. olvasás, mem. írás

csak Load és store következménye:

nagyon ritka progr.-nál, lokális vált.-kat használunk. ⇒
ezért kell volna LOAD-dal betölteni a tárolóba és Store-ral kiírni.

• kis regiszterkészlet	• nagy regiszterkészlet
• a relatív egyszerű fordítóprogramok	• bonyolult fordítóprogram. (egy RISC annyira jó, amennyire a hozzá adott fordító)

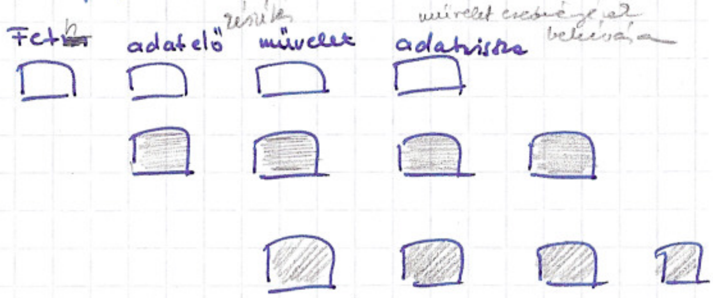
~~leegyszerűsített belső mikroprogram~~
~~(horizontális jellegű)~~

• ~~huzalozott műveleti vezérlés~~

• mikroprogram vertikális

• ~~huzalo~~ horizontális mikroprogram
(HUZALOZOTT VEZÉRLÉS)

ugyanolyan órajelek mellett
pipeline vezérlés diadala kezdődött



PROBLÉMA

1) UGRÁS   ADD R2, R4   adjuk össze R2-t R4-gyel, helyezzük R2-be
MOV R1, R2   tegyük R1-be R2 másolatát

R2 értéke még nincs visszaírva, mikor már a progr. használni akarja ⇒

2) ⇒ ADATÜTKÖZÉS

Megoldási módok: 1) NOP : ne csináljon semmit a processzor ⇒ húzza az időt.

a proc. ismeri az adatfüggőséget ⇒ nem hagyhatja az adatütközést ⇒ beírat 1 NOP utasítást. (kihagy 1 ciklust, de nem lesz számítási hibás).

Ez a fordító intelligenciáján múlik.

2) DATA FORWARDING

- a pipeline-ok kapcsolatban maradnak egymással
- már beolvasódott az utasítás ⇒ kell neki az adat ⇒ az adatot előre megadják.

3) SCORE BOARDING (értéktáblatartás)

- figyeli, hogy mit használunk, mely fejeztük be műveletet
- mindegyik pipeline-nak ad engedélyt, vagy tilt ⇒ mint a NOP utasítással egyenlő, de nincs benne a NOP, hanem 1 fázissal eltolja a pipeline-t.

ADD R4, R5

1) utasítás csere, utasítás átrendezés

adatfüggőség értéke a csere után megtörténik a végrehajtás, nem okoz függőséget.

UGRÁSJELZÉSEK:

- kitalálás: jósoljuk meg az ugrás várható kimenetelét
- kezdetben statisztika volt a futó programnál: a progr. ált. ugranak v. nem ugranak
  minden ugrás mellé tegyél 1 bitet, h. a pipeline nézője, h. ugrik, v. nem ugrik.

0+2 → nincs átv.
1+2 → ..
⋮
254+2 → v.
255+2 → van

} nem van, h. v. átvitel, de a fordító úgy fordít, h. valószínűleg, h. nem lesz átvitel

CASH
ismétlés