a az egyesekre normalizált mantissza törtrésze,

p a karakterisztika(hatványkitevő) eredeti értéke,

e az eltolás (a többlet) értéke, amelynek nagysága 2m-1-1, ahol m=8, 11, 15.

(25.ábra)

Lehetséges méretek:

= egyszeres pontosságú(single precision, 32 bit),
= dupla pontosságú(double precision, 64 bit),
= kiterjesztett pontosságú(extended precision, 80 bit),
= négyszeres pontosságú(quadrupled precision, 128 bit).

Adatformátumok:

(26.ábra)
= normalizált adatformátum,
= denormalizált(denormalized) adatformátum,
= nulla számérték,
= végtelen értékének adatformátuma,
= nem meghatározott számérték, 'nem-szám' (NaN=Not a Number) adatformátum;

b2.)Numerikus adatok tizes számrendszer szerinti tárolása

Cél: nagyobb adatmennyiség és kevesebb aritme-tikai művelet esetén
===> kevesebb legyen az adatkonverzió.

Formái: BCD(binary coded decimal), Aiken, Stibitz, Gray-kód, stb.
Leggyakoribb a BCD-kód alkalmazása:
a számokat számjegyenként kettes számrendszerbe konvertálva és 4 helyiértékre(tetrád-dá) kiegészítve.

0	0000
1	0001
2	0010
3	0011
4	0100
5	0101
6	0110
7	0111
8	1000
9	1001

Előjel részére:

1100	+	előjel,
1101	-	előjel.

(27.ábra)

b3.)Alfanumerikus adatok tárolási formája

Cél: az adatok tárolása, aritmetikai műveletvégzés igénye nélkül.

Karakterenként egy-egy, többnyire 8-bites jelkombináció (kódszó) hozzárendelése:

ASCII kód(American Standard Code for Information Interchange): mikroszámítógépeknél általánosan használt.

például:	A	41h	0100 0001
	B	42h	0100 0010
	0	30h	0011 0000
	1	31h	0011 0001
	9	39h	0011 1001
	+	2Bh	0010 1011

EBCDI kód(Extended Binary Coded Decimal Interchange Code): IBM nagygépeknél használt kódrendszer.

c.)Egyéb adattárolási módszerek

= jelölt(tagged) adattárolás,
= deszkriptoros tárolási forma,
= összetett strukturális forma.

3.1.2.Utasítások tárolási formái

Az utasítások szerkezete és a rendelkezésre álló elemi utasítások köre meghatározó a processzor struktúrájának kialakításában.

Az elemi 'gépi kódú' utasítás részei:

(28.ábra)

= **műveleti jelrész**(operation code, opcode): mi a teendő?
= **címrész**(address field): az operandusok tárolóbeli helyének a kijelölése.
= **kiegészítő, módosító rész**: a címzési előírás módosításához, pontossá tételéhez.

Az utasítások hossza:

= azonos mindig(pl. RISC processzoroknál 4 byte),
= változó(pl. CISC processzoroknál 1-17 byte).

Problémák:

= utasításhossz 2 utasításátvitel sebessége,
utasításfeldolgozás sebessége,
= utasításkészlet 2 processzor struktúrája,
feldolgozás sebessége.

a.)Utasításszerkezet

Előírja, hogy az utasítás melyik részét hogyan kell értelmezni.

Legáltalánosabban, egyidőben a következő címekre van szüksége a gépnek:

= első operandus címe,
= második operandus címe,
= eredmény címe,
= következő utasítás címe.

(29.ábra)

b.)Utasítástípusok, utasításkészlet

b1.)Utasítástípusok

= átviteli utasítások:

- tárolóhivatkozású utasítások,
- veremkezelő utasítások,
- periféria utasítások;

= műveleti utasítások:

- aritmetikai műveleti utasítások(+,-,*,/),
- logikai műveleti utasítások,(and, or, not),
- léptető/forgató(shift/rotate) utasítások,
- bitműveleti utasítások,
- karakterlánc(string) műveleti utasítások;

= vezérlő utasítások:

- feltétel nélküli ugrató utasítás,
- feltételes ugró utasítások,
- alprogram(szubrutin)hívó utasítás,
- 'return' utasítás,

- leállító utasítás,
- ciklusutasítás,

- megszakítást tiltó és engedélyező utasítások,
- 'halt' utasítás.

b2.)Utasításkészlet

Azon elemi (gépi kódú) utasítások összessége, amelyeket a gép a legalsó, hardver szinten értelmezni és végrehajtani tud.

Ez a programozó által használható legalsó szint.

Jellemzők:

= elemi utasítások száma,
= az elemi utasítások tartalma, az elvégzendő feladatok összetettsége,
= a kezelhető feltételek száma,
= az egyes jellemzők következetes használati lehetősége,
= az utasítások által nyújtott támogatás:
- a programíráshoz,
- a program fordításához,
- az ellenőrízhetőséghez.

Az utasításkészlet kialakítását befolyásolja, hogy mit bízunk a hardverre és mit bízunk a szoftverre:

===> szoftver-hardver közötti távolság (semantic gap)

Összetett utasításkészlet 2 CISC processzorok;

Egyszerűsített utasításkészlet 2 RISC processzorok.

3.1.3.Műveletek végrehajtása

Elvégezhető műveletek: = aritmetikai műveletek,
= logikai műveletek.

a.)Aritmetikai műveletek

- kettes számrendszer alapján:
  fixpontos számok körében,
  lebegőpontos számok körében;
- tízes számrendszer alapján:

25

BCD kód alapján.

Bináris jelrendszer alkalmazása révén, az aritmetikai műveletek visszavezethetők a logikai műveletekre.

a1.)Műveletek fixpontos számokkal

Összeadás:

Szokásos helyiértékes ábrázolás szerint végezhető el.

Például:
```
A=22        0 001 0110
B=75        0 100 1011
  ----   ---------------
  97        0 110 0001
```

Átvitelképzés: 10 és
11 esetén.

Kivonás:

Komplemenskód használatával; oka, hogy így:
= a kivonás visszavezethető az összeadásra,
= az előjel automatikusan adódik.

Például:
```
          A= 97       0 110 0001
        B=-62         1 011  1110

   B komplemense:     1 100 0010
                 -----  ---------------
   A+Bk          35  1 0 010 0011

          A= 45       0 010 1101
        B=-52         1 011 0100
          -----  ---------------
B komplemense:      1 100 1100
           -----  ---------------
A+Bk       -7           1 111 1001
```

az eredmény negatív ===> komplementálás

A+Bk -7 1 000 0111

Szorzás, osztás:

Az alkalmazott algorimus bonyolultabb, de visszavezethető sorozatos összeadásokra és eltolásokra (léptetésekre).

Fixpontos számok összeadásának és kivonásának közös algoritmusa:

(30.ábra)

A fixpontos számok szorzásának algoritmusa:

(31.ábra)

a2.)Műveletek lebegőpontos számokkal

Összeadás, kivonás:

Ha **a karakterisztikák megegyeznek**, azaz $k_a = k_b$, akkor

$$A + B = m_a \cdot 2^{k_a} + m_b \cdot 2^{k_b} = \left(m_a + m_b\right) 2^{k_a} = m_{(a+b)} \cdot 2^{k_{(a+b)}}$$

ahol $m_{(a+b)} = m_a + m_b$ és

Ha **a karakterisztikák nem egyeznek meg**, azaz $k_a \neq k_b$, akkor az eredmény karakterisztikája:

$$k_{(a+b)} = \max\left(k_a, k_b\right)$$

Ha $k_a > k_b$, és így $n = k_a - k_b$, akkor a B operandus mantisszáját osztani kell a $2^n = 2^{k_a - k_b}$ értékkel és a karakterisztikát ennek megfelelően növelni kell, azaz

$$m_{bu} = \frac{m_b}{2^n} = m \cdot 2^{-(k_a - k_b)}$$

$$A + B = m_a \cdot 2^{k_a} + m_b \cdot 2^{k_b} = m_a \cdot 2^{k_a} + m_{bu} \cdot 2^{k_a} = \left(m_a + m_{bu}\right) 2^{k_a} = m_{(a+b)} \cdot 2^{k_{(a+b)}}$$

ahol $m_{(a+b)} = m_a + m_{bu}$ és

Szorzás, osztás:

mantisszák esetében ===> fixpontos szorzás, osztás
karakterisztikák esetében ===> összeadás, kivonás

Lebegőpontos számok összeadásának, kivonásának algoritmusa: (32.ábra)

a3.)Műveletek tízes számrendszer alapján

BCD kódban ábrázolt számokkal; átvitelképzés:
- tetrádon belül kettes számrendszer alapján,
- tetrádok között tízes számrendszer szerint.

Például:

```
245             0010 0100 0101
137             0001 0011 0111
-----           ----------------------
382             0011 0111 1100
korrekció:                0110
                ----------------------
                    0011 1000 0010
```

b.)Logikai műveletek

b1.)Alapműveletek

- logikai VAGY-művelet(OR),
- logikai ÉS-művelet(AND),
- logikai NEM-művelet(NOT).

	NEM-művelet	VAGY-művelet	ÉS-művelet
műszaki írásmód	$C = \overline{A}$	$C = A+B$	$C = AB$
matematikai írásmód	$C = \neg A$	$C = A \vee B$	$C = A \wedge B$

A műveletek igazságtáblázata:

28

operan-dusok		NEM művelet		VAGY művelet	ÉS művelet
A	B	$C=\overline{A}$	$C=\overline{B}$	$C=A+B$	$C=AB$
0	0	1	1	0	0
0	1	1	0	1	0
1	0	0	1	1	0
1	1	0	0	1	1

b2.)Összetett műveletek

= ekvivalencia(azonosság) művelete(jele: 🗁, illetve 1),

= antivalencia művelete(jele: 1, illetve □),

= NEM-VAGY(NOR) művelet(Pierce-művelet),

= NEM-ÉS(NAND) művelet(Sheffer-művelet).

A műveletek igazságtáblázata:

A	B	ekviva-lencia	antiva-lencia	NEM-VAGY	NEM-ÉS
		A🗁B	A1B	$\overline{A+B}$	$\overline{AB}$
0	0	1	0	1	1
0	1	0	1	0	1
1	0	0	1	0	1
1	1	1	0	0	0

A táblázat összetett kifejezései az alapműveletek segítségével:

= ekvivalencia: 

= antivalencia: 

= NEM-VAGY: 

= NEM-ÉS: 

b3.)Logikai függvények

Jelentőségük: különösen a gépek tervezésekor

29

'Teljes összeadó' (két bináris számjegy és az átvitel összeadására szolgáló egység) összeg(S) és átvitel(C) kimenete:

A	B	D	S	C
0	0	0	0	0
0	0	1	1	0
0	1	0	1	0
0	1	1	0	1
1	0	0	1	0
1	0	1	0	1
1	1	0	0	1
1	1	1	1	1

$$S=\overline{A}\cdot\overline{B}\cdot D+\overline{A}\cdot B\cdot\overline{D}+A\cdot\overline{B}\cdot\overline{D}+A\cdot B\cdot D=(A\otimes B)\cdot D+(A\oplus B)\cdot\overline{D}=(A\oplus B)\oplus D$$

$$S=\overline{A}\cdot B\cdot D+A\cdot\overline{B}\cdot D+A\cdot B\cdot\overline{D}+A\cdot B\cdot D=A\cdot B+(A\oplus B)\cdot D$$

c.)Aritmetikai-logikai műveletvégző egység(ALU)

Részei:
= teljes-összeadó egység,
= léptető áramkörök,
= logikai műveletvégző,
= adatregiszterek(AC - akkumulátor).

Állapotjelző regiszter(flag regiszter):

= átvitel(carry),
= nulla(zero),
= túlcsordulás(overflow),
= előjel(sign).

Félösszeadó és teljes összeadó vázlata:

(33.ábra)

ALU vázlata:

(34.ábra)

3.1.4.Utasítások végrehajtása

a.)Utasitásvégrehajtás lépései

= utasításelőkészítés(fetching),
= utasításszámláló regiszter(PC) tartalmának növelése,
= műveleti kód értelmezése(decoding), operandusok címének meghatározása,
= operandusok előkészítése,
= művelet végrehajtása(executing),
= eredmény elhelyezése.

Neumann-elvű, hagyományos struktúrájú számítógépek utasításfeldolgozása soros rendszerű:

(35.ábra)

b.)Műveleti vezérlés

Feladat: az utasításokban meghatározott műveletek elemi lépéseinek vezérlése.

Lehetőségek: = huzalozott módon,
= mikroprogramozott módon.

(36.ábra)

Példa: LDA X [X] 2 AC

elemi lépések	érintett vezérlési pontok
[PC] □ R-sín □ S-sín □ MAR	R2, S2, M2
címdekódolás	
[mem(MAR)] □ MBR	
[MBR] □ R-sín □ IR	R1, C1
[IR(op.kód)] □ □MAR	C2
[IR(címrész)] □ S-sín □ MAR	S1, M2
címdekódolás	
[mem(MAR)] □ MBR	
[MBR] □ R-sín □AC	R1, A1

c.)Mikroprogramozott műveleti vezérlés