

ЛЕКЦИЯ 3 ПРЕДСТАВЯНЕ НА ДАННИТЕ В ОП

- ⌚ Определения
- ⌚ Цели числа
- ⌚ Дробни числа
- ⌚ Десетични числа
- ⌚ Знаци (знаков код)
- ⌚ Аналогови данни
- ⌚ Машинна програма

КА-03

1/56

ОПРЕДЕЛЕНИЯ

Поредица от битове, които се **обработват едновременно**, се нарича **дума**.

Размер на думата е броя на битовете в нея.

От IBM 360 (1966) ЦП може да обработва **думи с различен размер**.

Размерът на думите е кратен на размера на клетката на ОП.

Думите се съхраняват във фиксиран брой **клетки с последователни адреси**.

Така за да посочим **дума в ОП** е достатъчно да посочим **адреса на първата клетка**.

КА-03

2/56

ЗАБЕЛЕЖКИ

❶ Фирмите ползват **различни термини**:

♣ IBM: байт, полудума = 2 байта (16 бита), дума = 4 байта (32), двойна дума = 8 байта (64).

♣ Интел: байт, дума = 2 байта (16), двойна дума = 4 байта (32), четворна дума = 8 байта (64).

❷ Към **началния адрес** на думите може да се поставят **изисквания за кратност**, известни като **интегрални граници**:

♣ IBM 360: полудума от адрес, кратен на 2; дума от кратен на 4; двойна дума от кратен на 8.

♣ IBM 370 и Интел само препоръчват такива граници за по-бърз достъп до ОП.

КА-03

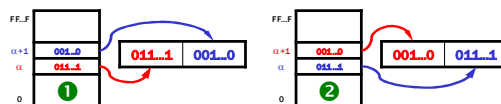
3/56

ЗАБЕЛЕЖКИ (прод.)

Разполагането на **дума в две клетки** може да стане **по два начина**:

❶ **Левите** (старшите) битове на думата в клетката с **по-малък адрес**: **голямокраен компютър** (big endian) – IBM, Моторола;

❷ **Левите** (старшите) битове на думата в клетката с **по-голям адрес**: **малокраен компютър** (little endian) – DEC, Интел.

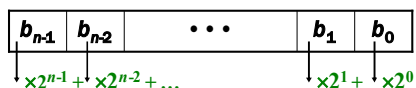


КА-03

4/56

ЦЕЛИ ЧИСЛА БЕЗ ЗНАК

Най-естествената интерпретация на n -те бита в една дума е следната:



Диапазон: $[0, 2^n - 1]$ или $[-(2^n - 1), 0]$.

Такова тълкувание на думата

пренос се нарича **цели числа без знак**. **заем**

1010	10	0010	2	1010	10	1010	10	10010	2		
+	0011	3	+	0011	3	+	0111	7	-	0011	3
1101	13	0101	5	10001	01	0111	7	1111	15		

КА-03

5/56

ЦЕЛИ ЧИСЛА СЪС ЗНАК

За да представяме **едновременно** както **положителни**, така и **отрицателни числа** трябва да обмислим и **други решения за тълкуване на битовете в думата**.

Представяните по този начин числа се наричат **числа със знак**.

Възможни са **две решения**:

❶ **абсолютна стойност** и **знаците + или -**;

❷ **допълване** на отрицателните числа с предварително определено **число**.

КА-03

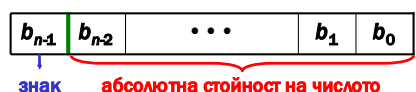
6/56

ПРАВ КОД

В **10-ична** ПБС имаме **2 знака** и **10 цифри**.

В **2-ична** ПБС имаме **2 знака**, но и **2 цифри**.

Следователно, можем да отделим първия **бит за представяне на знака** ($0 = +, 1 = -$).



Диапазон: $[-(2^{n-1} - 1), +(2^{n-1} - 1)]$.

Такова тълкуване на думите се нарича **прав код** на числата със знак.

КА-03

7/56

ПРИМЕРИ: ПРАВ КОД

Представяне:

1111 \rightarrow -7 (min) 0111 \rightarrow +7 (max)

1000 \rightarrow -0 0000 \rightarrow +0

1010 \rightarrow -2 0101 \rightarrow +5

Събиране:

+	0010	+2	010	+	1010	-2	0010	+2	011	+	1010	-2		
+	0011	+3	011	+	1011	-3	+	1011	-3	-	010	+	0011	+3
	0101	+5	101		1101	-5		1001	-1	001	0001	+1		

Изваждане:

$$1010 - 0011 = 1010 + 1011$$

$$(-2 - +3 = -2 + -3)$$

КА-03

8/56

ДОПЪЛНЕНИЯ

В p -ична ПБС с n цифри можем да представим p^n числа без знак.

Половината от тях (от 0 до $p^n/2$) тълкуваме чрез стойността им като положителни числа.

Другата половина използваме за представяне на отрицателните числа, които допълваме с определено число D : p^n - пълно допълнение (до p), или $p^n - 1$ - непълно допълнение (до $p-1$).

КА-03 9/56

ОБРАТЕН КОД

Непълното допълнение $+(p^n - 1)$ се нарича **обратен код** или допълнение до $p-1$.

Нека $p^n/2 > A \geq 0$. Обратният код на A е \bar{A} .

Нека $A \leq 0$. Тогава $A + (p^n - 1) = (p^n - 1) - |A|$.

Сега обратният код се получава като **цифрите на $|A|$ се изваждат от цифрата $(p-1)$!**

При $p=2$ такова изваждане е равносилно на **инвертиране на цифрите на $|A|$** .

Така при $p=2$:

- Обратният код на $a \geq 0$ съвпада с **правия**.
- Обратният код на $a < 0$ **инвертира** $-a > 0$.

КА-03 10/56

ДОПЪЛНЕНИЕ ДО 1

Нека разгледаме една операция, която **инвертира всички битове** на дадена дума.

Тази операция се нарича **допълнение до 1**.

Тя ни служи за три цели:

- смяна на знака** при числата, записани в обратен код: $0101 (+5) \leftrightarrow 1010 (-5)$.
- намиране на обратния код** на отрицателни числа: $|-5| = +5 = 0101 \rightarrow 1010 (-5)$.
- разчитане на отрицателните числа**, които са записани в обратен код: $1011 = ? \rightarrow 0100 = 4 \rightarrow -4 (1011)$.

КА-03 11/56

ПРИМЕРИ: ОБРАТЕН КОД

Диапазон: $[-(2^{n-1}-1), +(2^{n-1}-1)]$.

$1000 \rightarrow -7$ (min) $0111 \rightarrow +7$ (max)
 $1111 \rightarrow -0$ $0000 \rightarrow +0$

Събиране:

$$\begin{array}{r} + \quad 0010 \quad +2 \quad + \quad 1101 \quad -2 \quad + \quad 0010 \quad +2 \quad + \quad 1101 \quad -2 \quad + \quad 1101 \quad -2 \\ + \quad 0011 \quad +3 \quad + \quad 1100 \quad -3 \quad + \quad 1100 \quad -3 \quad + \quad 0011 \quad +3 \quad + \quad 0010 \quad +2 \\ 0 \quad 0101 \quad +5 \quad 1 \quad 1001 \quad -6 \quad 0 \quad 1110 \quad -1 \quad 1 \quad 0000 \quad +0 \quad 0 \quad 1111 \quad -0 \\ + \quad \underline{\quad 0 \quad} \quad + \quad \underline{\quad 1 \quad} \quad + \quad \underline{\quad 0 \quad} \quad + \quad \underline{\quad 1 \quad} \quad + \quad \underline{\quad 1 \quad} \quad + \quad \underline{\quad 0 \quad} \\ \hline 0101 \quad +5 \quad 1010 \quad -5 \quad 1110 \quad -1 \quad 0001 \quad +1 \quad 1111 \quad -0 \end{array}$$

Изваждане:

$1101 - 0011 = 1101 + 1100$
 $(-2 - +3 = -2 + -3)$

КА-03 12/56

ДОПЪЛНИТЕЛЕН КОД

Пълното допълнение $(+p^n)$ се нарича **допълнителен код** или допълнение до p .

Нека $p^n/2 > A \geq 0$. Допълнителният код на A е \bar{A} .

Нека $A \leq 0$. Тогава $A + p^n = ((p^n - 1) - |A|) + 1$.

Т. е. допълнителният код сега се получава като **към обратния код се добави 1!**

При $p=2$: **инвертираме $|A|$ и добавяме 1**.

Така при $p=2$:

- Допълнителният код на $a \geq 0$ съвпада с **правия**.
- Допълнителният код на $a < 0$ се получава като **добавим 1 към обратния код на a** .

КА-03 13/56

ДОПЪЛНЕНИЕ ДО 2

Да разгледаме операцията, която **инвертира битовете** на една дума и добавя 1.

Тази операция се нарича **допълнение до 2**.

Тя ни служи за три цели:

- смяна на знака** при числа, записани в допълнителен код: $0101 (+5) \leftrightarrow 1011 (-5)$.
- намиране на допълнителния код** на отрицателните: $|-5| = +5 = 0101 \rightarrow 1011$.
- разчитане на отрицателните числа**, които са записани в допълнителен код: $1010 = ? \rightarrow 0110 = 6 \rightarrow -6 (1010)$.

КА-03 14/56

ПРИМЕРИ: ДОПЪЛНИТЕЛЕН КОД

Диапазон: $[-2^{n-1}, +(2^{n-1}-1)]$.

$1000 \rightarrow -8$ (min) $0111 \rightarrow +7$ (max)
 $0000 \rightarrow 0$ -8 няма противоположно: $+8$

b_{n-1}	b_{n-2}	...	b_1	b_0
\downarrow	\downarrow		\downarrow	\downarrow
$x \cdot 2^{n-1} +$	$x \cdot 2^{n-2} +$		$x \cdot 2^1 +$	$x \cdot 2^0$

Събиране:

$$\begin{array}{r} + \quad 0010 \quad +2 \quad + \quad 1110 \quad -2 \quad + \quad 0010 \quad +2 \quad + \quad 1110 \quad -2 \quad + \quad 1110 \quad -2 \\ + \quad 0011 \quad +3 \quad + \quad 1101 \quad -3 \quad + \quad 1101 \quad -3 \quad + \quad 0011 \quad +3 \quad + \quad 0010 \quad +2 \\ 0 \quad 0101 \quad +5 \quad 1 \quad 1011 \quad -5 \quad 0 \quad 1111 \quad -1 \quad 1 \quad 0001 \quad +1 \quad 1 \quad 0000 \quad 0 \end{array}$$

Изваждане:

$1110 - 0011 = 1110 + 1101$
 $(-2 - +3 = -2 + -3)$

КА-03 15/56

ПРЕПЪЛВАНЕ ПРИ ЧИСЛА СЪС ЗНАК

- При **сумиране на числа с различни знаци не може** да се получи **препъване**:
 $-2^{n-1} \leq a < 0, 0 \leq b < 2^{n-1} \Rightarrow -2^{n-1} \leq a + b < 2^{n-1}$
- Наличието на **пренос не е препъване**:

$$\begin{array}{r} + \quad 1110 \quad -2 \quad + \quad 1110 \quad -2 \quad + \quad 0101 \quad +5 \quad + \quad 1100 \quad -4 \\ + \quad 1101 \quad -3 \quad + \quad 0010 \quad +2 \quad + \quad 0110 \quad +6 \quad + \quad 1010 \quad -6 \\ 1 \quad 1011 \quad -5 \quad 1 \quad 0000 \quad 0 \quad 0 \quad 1011 \quad -5 \quad 1 \quad 0110 \quad +6 \end{array}$$
- Сигнал за препъване е получаването на **сума с различен от събираемите знак**:

$$\text{прен} = (\bar{a}_{n-1} \wedge \bar{b}_{n-1} \wedge s_{n-1}) \vee (a_{n-1} \wedge b_{n-1} \wedge \bar{s}_{n-1})$$

КА-03 16/56

МОДИФИЦИРАНИ КОДОВЕ

Формулата за преплъване е доста сложна. За да се опрости се използват **модифицирани обратен и допълнителен код**, при които **знаковите разреди са два**.

Дублирането на знака се реализира **преди сумиране**, а сигнализация за преплъване е **различieto на двата знакови бита на сумата**.

$+ \begin{array}{r} 11110 \\ 11101 \\ 11011 \end{array} \begin{array}{l} -2 \\ -3 \\ -5 \end{array}$	$+ \begin{array}{r} 11110 \\ 00010 \\ 00000 \end{array} \begin{array}{l} -2 \\ +2 \\ 0 \end{array}$	$+ \begin{array}{r} 00101 \\ 00110 \\ 01011 \end{array} \begin{array}{l} +5 \\ +6 \\ -5 \end{array}$	$+ \begin{array}{r} 11100 \\ 11010 \\ 10110 \end{array} \begin{array}{l} -4 \\ -6 \\ +6 \end{array}$
--	---	--	--

$npren = s_n \oplus s_{n-1}$

КА-03 17/66

КОД С ИЗМЕСТВАНЕ

- 1 Сетунъ има троична симетрична: **-1, 0, 1**.
- 2 В CDC е бил използван **обратен код**.
- 3 Днес се използва **само допълнителен код**.
- 4 Често е **полезен и код с изместване**: към всички числа от $[-2^{n-1}, 2^{n-1}-1]$ се добавя **изместване 2^{n-1}** за преход към $[0, 2^n-1]$.
- 5 При изместване знаковият бит е **обърнат**: $0 \rightarrow -, 1 \rightarrow +$, а събирането - затруднено.

КА-03 18/66

ТЪЛКУВАНЕ

Четирибитова дума ще се тълкува като:

дума	тълкуване на думата				
	без знак	прав	обратен	допълнителен	изместване
0000	0	+0	+0	+0	-8
0001	1	+1	+1	+1	-7
0010	2	+2	+2	+2	-6
0011	3	+3	+3	+3	-5
0100	4	+4	+4	+4	-4
0101	5	+5	+5	+5	-3
0110	6	+6	+6	+6	-2
0111	7	+7	+7	+7	-1
1000	8	-0	-7	-8	+0
1001	9	-1	-6	-7	+1
1010	10	-2	-5	-6	+2
1011	11	-3	-4	-5	+3
1100	12	-4	-3	-4	+4
1101	13	-5	-2	-3	+5
1110	14	-6	-1	-2	+6
1111	15	-7	-0	-1	+7

КА-03 19/66

ДРОБНИ ЧИСЛА

Използват се **три варианта** за представяне на **дробни** (част от рационалните) числа:

- 1 с фиксирана запетая;
- 2 с естествена запетая;
- 3 с плаваща запетая.

Фиксирана запетая означава, че **мястото на двоичната запетая никога не се мени**. То е определено предварително, т. е. **част от битовете на думата са цифрите в цялата част на числото, а останалите - в дробната**.

КА-03 20/66

ФИКСИРАНА ЗАПЕТАЯ

- 1 числата са (почти) **симетрични спрямо 0**.
- 2 запетаята определя **най-малкото $\neq 0$** .
- 3 **диапазонът не е особено голям**.
- 4 **покритието (на оста) е равномерно**.
- 5 **събирането не зависи от запетаята**.
- 6 **умножението зависи от запетаята**.
- 7 **опасна операция е деленето**.
- 8 **смяната на запетаята** е умножение с фиксирано число - **мощабен множител**.
- 9 **най-популярни** са два избора на място:

ЗН	цели числа ($f=0$);
ЗН	правилни дроби ($i=0$) $\in (-1,1)$.

КА-03 21/66

ЕСТЕСТВЕНА ЗАПЕТАЯ

- 1 Записаното число се дели на две части:
 - $\rightarrow k$ цифри за запис на цифрите на числото;
 - $\rightarrow \lceil \log_p(k+1) \rceil$ цифри посочващи колко цифри има дробната част (изгодно е $k=p^m-1 \rightarrow m$ цифри).
- 2 Така запетаята е между цифрите на числото.
- 3 Събирането е затруднено.
- 4 Диапазонът на числата е по-голям.
- 5 Използва се главно в калкулаторите.

0025 1 \rightarrow 2,5 0025 3 \rightarrow 0,025 0025 0 \rightarrow 25
 0001 4 \rightarrow 0,0001 ($\min \neq 0$)
 9999 0 \rightarrow 9999, (\max)

КА-03 22/66

ПЛАВАЩА ЗАПЕТАЯ

За да увеличим диапазона на представимите числа трябва да **пожертваме броя на верните цифри в записа** ($ctg x \rightarrow 0, x \rightarrow \infty$).

Нека $x > 0$ и $p \geq 2$. Тогава $x = m \times p^e$ за някои m и e - цяло число. Такъв **запис** на числата се нарича **експоненциален**, m - **мантиса**, e - **порядък**, p - **основа**.

Цифрите в записа на мантисата определят **точността** (брой верни цифри), а броят на **цифрите за запис на порядъка - диапазона**.

Представянето чрез m и e се нарича **плаваща запетая** (p се фиксира и не се мени).

КА-03 23/66

ЗАБЕЛЕЖКИ

Представянето чрез m и e не е еднозначно:
 $11 \rightarrow 0,1100 + 2 ; 0,0110 + 3 ; 0,0011 + 4$

Преместването на цифрите на мантисата в дясно изисква увеличаване на порядъка.

Нулева мантиса определя **еднозначно $x = 0$** .

Еднозначност при $x \neq 0$: $1/p \leq |m| < 1$ - **нормализирано представяне** (първата цифра на мантисата да бъде $\neq 0$).

При $p = 2$ първата цифра на нормализирана мантиса винаги е 1! При $p = 16$ двоичната мантиса може да започва с **до 3 нули**.

КА-03 24/66

СЪБИРАНЕ

1 при равни порядъци можем да съберем мантисите.

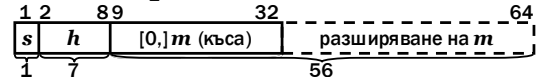
2 при различни порядъци трябва да денормализираме едната мантиса.

$$\begin{array}{r}
 +1201 +5 : 12\ 010 \\
 +2051 +5 : 20\ 510 \\
 +3252 +5 : 32\ 520 \\
 \hline
 +6201 +5 \\
 +4051 +5 \\
 +13252 +5 \\
 +1325 +6 \\
 \hline
 +1201 +5 \\
 -1202 +5 \\
 -0001 +5 \\
 -1000 +2 \\
 \hline
 +2010 +4 \\
 +2051 +4 \\
 +4061 +4 \\
 \hline
 +1201 +5 \\
 +1202 -1 \\
 +1201 +5 \\
 a + b = a \\
 \hline
 +1201 +5 \\
 +0000 +5 \\
 +1202 -1 \\
 +0000 +5 \\
 +0000 -9 \\
 \hline
 \end{array}$$

КА-03 25/56

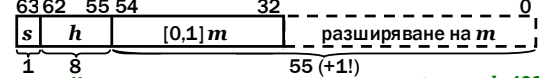
ПРИМЕРИ: IBM И VAX

1 IBM 360: $p=16, |m| < 1, h = e + 64$.



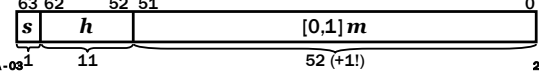
Стойността на числото е $(-1)^s \times 0, m \times 16^{h-64}$.

2 VAX F и D: $p=2, |m| < 1, 255 \geq h = e + 128 > 0$.



Стойността на числото е $(-1)^s \times 0, 1 m \times 2^{h-128}$.

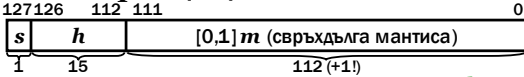
3 VAX G: $p=2, |m| < 1, 2047 \geq h = e + 1024 > 0$.



КА-03 26/56

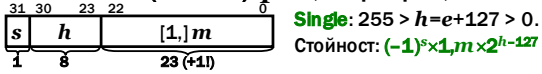
ПРИМЕРИ: VAX И IEEE

4 VAX H: $p=2, |m| < 1, 32767 \geq h = e + 16384 > 0$.

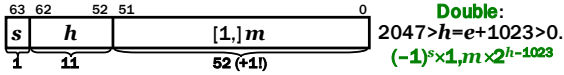


Стойността на числото е $(-1)^s \times 0, 1 m \times 2^{h-16384}$.

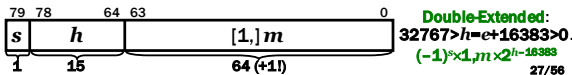
5 IEEE 754 (1985 г.): $p=2, 1 \leq |m| < 2$,



Single: $255 > h = e + 127 > 0$.
Стойност: $(-1)^s \times 1, m \times 2^{h-127}$



Double: $2047 > h = e + 1023 > 0$.
 $(-1)^s \times 1, m \times 2^{h-1023}$



Double-Extended: $32767 > h = e + 16383 > 0$.
 $(-1)^s \times 1, m \times 2^{h-16383}$

КА-03 27/56

ОСОБЕНИ СТОЙНОСТИ

1 VAX, $h=0$:

- $s = 0$ - представя числото 0,0 (плаваща 0);
- $s = 1$ - неизползвани стойности (запасени).

2 IEEE 754 (ANSI):

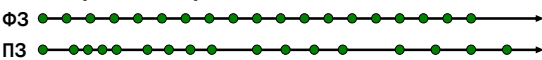
- $h = 111...1$ [255, 2047, 32767], $m \neq 0$: NAN (Not A Number);
- $h = 111...1$ [255, 2047, 32767], $m = 0$: $(-1)^s \times [\pm\infty]$: $a/0$;
- $h = 0, m \neq 0$: денормализирана мантиса (без скритото 1);
- $h = 0, m = 0$: $(-1)^s \times 0$ [± 0].

3 Не числата на IEEE (NAN) са предвидени като сигнал за грешка. Те биват два вида: тихи (quiet) QNAN [$m_{ст} = 1$] - неопределена операция ($\sqrt{-1}$) и значими (signaling) SNAN [$m_{ст} = 0$] - невалидна операция (неинициализирана променлива).

КА-03 28/56

СРАВНЯВАНЕ НА ФЗ И ПЗ

1 Покритие на реалната ос.



2 Оценка на допустимата грешка:
ФЗ: фиксирана абсолютна грешка;
ПЗ: фиксирана относителна грешка.

3 Опасна операция:
ФЗ: делене (загуба на дробната част);
ПЗ: събиране на близки по абсолютна стойност числа с различни знаци.
 $(+1201 +5) + (-1202 +5) = -1000 +2 !!!$

КА-03 29/56

ДЕСЕТИЧНИ ЧИСЛА

- Необходимост: $0, 1_{(10)} = 0, 0(0011)_{(2)}$.
- Цифри: 1 десетична = 4 двоични цифри.
- Нелегални комбинации (6): A, B, C, D, E, F.
- Едноцифрено събиране:

$$\begin{array}{r}
 + 0011\ 3 \\
 + 0100\ 4 \\
 \hline
 0111\ 7 \odot \\
 + 0000\ 0 \\
 \hline
 0111\ 7 \odot
 \end{array}
 \quad
 \begin{array}{r}
 + 0101\ 5 \\
 + 0110\ 6 \\
 \hline
 1011\ ? \oplus \\
 + 0110\ 6 \\
 \hline
 0001\ 1 \odot
 \end{array}
 \quad
 \begin{array}{r}
 + 1001\ 9 \\
 + 1000\ 8 \\
 \hline
 0001\ 1 \oplus \\
 + 0110\ 6 \\
 \hline
 0111\ 7 \odot
 \end{array}$$

След първото двоично събиране определяме десетична корекция 6/0 за второ събиране.

КА-03 30/56

ВИДОВЕ ДКД ЧИСЛА

Поради представянето на всяка десетична цифра чрез четири двоични десетичните числа се наричат още и Двоично Кодирани Десетични (ДКД) числа - Binary Coded Decimal (BCD) numbers.

8-битова клетка (байт) дава 2 възможности:

- 1 неопаковани ДКД: фиксирана стойност в левия (старшия) полубайт - $3_{(16)}$ или $F_{(16)}$ и 1 цифра;
- 2 опаковани ДКД: цифра във всеки полубайт (2) зонов полубайт

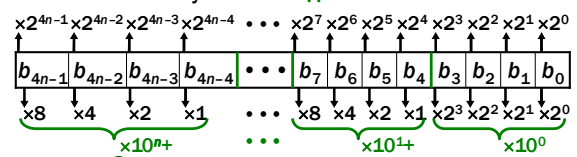


КА-03 31/56

ДВОИЧНИ И ДКД ЧИСЛА

Една и съща поредица от битове може да бъде тълкувана както като запис на двоично число, така и като запис на опаковано двоично кодирано десетично число.

1 Тълкуване като двоично число.



2 Тълкуване като десетично число.

1 133 = 1000101 = -85 2

КА-03 32/56

ЗНАЦИ

Неотрицателните цели числа са **универсална азбука** за кодиране. Те могат да представят и **знаците**, с които си пишем човешката реч.

Най-удобно е да се използва **равномерно кодиране** – всяко число с равен брой цифри.

Разпределението на кодиращите числа по кодираните знакове **се нарича знаков код**.

Остава да уточним само **основата на кода**, т. е. броя на цифрите, с които ще записваме кодиращите числа. **Тя определя и броя на знаците**, които ще се кодират (представят).

КА-03

33/66

ЗНАКОВИ КОДОВЕ

Принципи на знаковите кодове:

- 1 Десетичните **цифри** се кодират с **последователни цели числа**;
- 2 Буквите се кодират **съгласно азбучния им ред** (но, за коя азбука говорим?).

Примери за знакови кодове са:

- 1 Код на **Бодо**, изобретил буквопечатания телеграф: **5 битов код (32 знака)**.
- 2 При 12 битовата си клетка **DEC (PDP-8)** използва **6 битов код (64 знака)**.

КА-03

34/66

ЗНАКОВИ КОДОВЕ (прод.)

- 3 Американски Стандартен Код за Обмен на Информация (**АСКОИ**) – American Standard Code for Information Interchange (**ASCII**): 7-битов код (128 знака), но в 8 бита.
- 4 Разширен Двоично-Кодиран Десетичен Код (**РАКДК**) – Extended Binary Coded Decimal Information Code (**EBCDIC**): 8 бита (256 зн.).
- 5 **Разширен АСКОИ**: 8 бита (256 знака).
- 6 Код на Американския Национален Стандарт Институт (**АНСИ**) – American National Standard Institute (**ANSI**): 8 бита.

КА-03

35/66

ЗАБЕЛЕЖКИ

- 1 **Макар и създадени в САЩ** повечето кодове са стандартизирани и в много други страни (**съгласно БДС КОИ 7 и 8 = ASCII**, ДКОИ = **EBCDIC**) и дори от **Международната организация по стандартите (ISO – International Standard Organization)**: ISO xxxx.
- 2 Днес ISO предлага нови 16- и 32-битови кодове, наречени **UNICODE**, кодиращи съответно 65 536 и над 4 милиарда знака.
- 3 В **UNICODE има място за всички азбуки**: латинската, българската, гръцката, арабската, еврейската, арменската и дори (!) китайската.

КА-03

36/66

РАЗШИРЕН АСКОИ (EXTENDED ASCII)

Някои групи от знакове и техните кодове са:

- 1 управляващи: 0÷31 (00÷1F) и 127 (7F), вкл.:
 - ♣ акустичен сигнал (BEL): 7 (07);
 - ♣ хоризонтална табулация (HT): 9 (09);
 - ♣ придвижване на хартията на нов ред (LF): 10 (0A);
 - ♣ придвижване в началото на реда (CR): 13 (0D);
 - ♣ изтриване (DEL): 127 (7F).
- 2 цифри (0÷9): 47÷57 (**30÷39**) [F0÷F9 в EBCDIC!].
- 3 главни букви латиница (A÷Z): 65÷90 (41÷5A).
- 4 редовни букви латиница (a÷z): 97÷122 (61÷7A).
- 5 главни букви кирилица (A÷Я): 128÷159 (80÷9F).
- 6 редовни букви кирилица (a÷я): 160÷191 (A0÷BF).

КА-03

37/66

АНАЛОГОВИ ДАННИ

Цифровите компютри не тълкуват и не обработват аналогови данни (образ, звук) без преобразуването им в поредица от цифри.

Някои периферни **устройства** (скенер, микрофон) могат да **преобразуват изображения и звук в специфична поредица от нули и единици**, които записват в **ОП**.

Други периферни **устройства** (екран, печат, тон колона) при получаване на определени поредици **от нули и единици** могат да покажат съответната **картина или** да произведат **звук**.

Обработката се реализира по програмен път.

КА-03

38/66

ЦИФРОВО ПРЕДСТАВЯНЕ НА ИЗОБРАЖЕНИЯ

Използват се **два варианта** за цифрово представяне на изображения (образи и чертежи): **растерен** и **векторен**.

Растерният вариант експлоатира **недостатъците на човешкото око**.

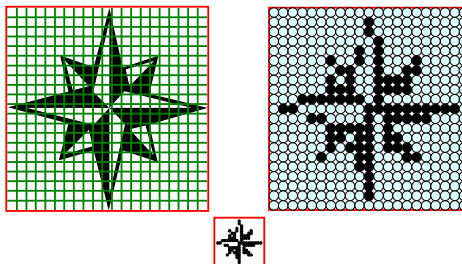
Векторният експлоатира механизма на **рисуване от страна на художника**.

КА-03

39/66

ПРИМЕР: РАСТЕРЕН ФОРМАТ

Експлоатира недостатъка на човешкото око да вижда **слято близки точки**.



КА-03

40/66

ПРЕДСТАВЯНЕ НА ТОЧКА

I. Монохромен формат (черно-бял):

- 1) с 1 бит (има – няма, черно-бяла графика);
- 2) с 256 отпънка (0 – бяло, 255 – черно и сиво).

II. Представяне на цветове – чрез смесване на базови цветове:

- 1) при излъчване на светлина (телевизия):
Червено (Red)-Зелено (Green)-Синьо (Blue) – RGB или Нюанс(Hue)-Наситеност (Saturation)-Осветеност(Luminosity) – **HSL**.
- 2) при отразяване на светлина (печат): **Небесно синьо (Cyan)-Светлорозово (Magenta)-Жълто (Yellow)-Черно (black) – CMYK**.

КА-03

41/56

КОДИРАНЕ НА RGB ЦВЯТ

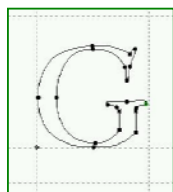
- 1) При 256 степенна скала (чисти цветове – true color) – **24** бита (16 777 216 цвята);
- 2) При 32 степенна скала – **15** бита (32 768);
- 3) При смесена скала (червено и синьо с 32, а зелено с 64 бита) – **16** бита (65 538);
- 4) Чрез избор на цвят от палитра, която може да бъде фиксирана предварително или добавена към изображението чрез 24-битов цвят: **4, 8, 15** или **16** бита (16, 256, 32 768 или 65 536 цвята в палитрата).

КА-03

42/56

ПРИМЕР: ВЕКТОРЕН ФОРМАТ

Проследява маниера на рисуване: с **четка** с определена дебелина взимаме желаня **цвят** и свързваме две точки с **крива линия**.



Точките и кривите се представят с **определен брой числа**.
Нарича се и **обектно-ориентирана графика**.

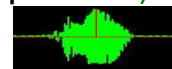
КА-03

43/56

ЦИФРОВО ПРЕДСТАВЯНЕ НА ЗВУК

Основава се на **недостатъците** на човешкото **ухо**, което регистрира звукови трептения с честота от **16 Hz** до **20 KHz** и различна амплитуда.

За да запазим сведения за различимите честоти е достатъчно да **измерваме амплитудата** през двойно по-малък интервал – **44,1 KHz**.



КА-03

44/56

ЦИФРОВА ОЦЕНКА НА АМПЛИТУДАТА

- 1) С 256 степени – **8-битов звук** (достатъчни са за човешки говор).
 - 2) С 65 536 степени – **16-битов звук** (необходими са за качествена музика).
- При **качествен 16-битов стерео-звук** (два звукови канала) за запис на **1 секунда музика** са необходими **около 150 000 бита** или за запис на **една минута музика – около 9 мегабайта**.

КА-03

45/56

МАШИННА ПРОГРАМА

За да върши полезна работа един **компютър** той трябва да **изпълнява** зададения му **алгоритъм**, явяващ се **последователност** от **елементарни действия**.

Кодираният вид на алгоритъма се нарича (машинна) **програма** и се съхранява в **ОП**.
Описанието на **едно елементарно действие** се нарича **машинна инструкция** (МИ).

В ОП, където живее програмата, има само 0 и 1, т. е. трябва да дадем ново **тълкувание на думата, но вече като машинна инструкция**.

КА-03

46/56

МАШИННИ ИНСТРУКЦИИ

Всяка **МИ** трябва да посочи две неща:

- 1) какво трябва да се свърши: +, – и др.;
- 2) с какво ще се извърши действието.

Действията се прилагат **към** междинни **данни**, които живеят в **ОП**, а **резултатът** от тях **също** се съхранява в **ОП** за следващия етап.

Т. е. **посочването** на операндите и резултата **става чрез адреси** от ОП.

Битовите на една МИ се **поделят на две части**:

КОП (какво?) | **Адресна част (със какво?)**

Заб. КОП = Код на ОПерация (вкл. размер на данните).

КА-03

47/56

ВИДОВЕ ИНСТРУКЦИИ

Машинните инструкции са **два вида**:

- 1) **обработващи** за **извършване** на определени **пресмятания**: събиране, изваждане и др.;
- 2) **управляващи** за **анализиране** на възникналите **обстоятелства** и вземане на решение.

Адресната част на всяка от тях се поделя на отделни части, наречени **адресни полета**.

Тъй като данните, които идентифицира едно адресно поле, са в ОП **най-естествено** (но не винаги и най-удобно) **е в него да бъде записан пълен адрес** от ОП.

КА-03

48/56

АДРЕСНИ ПОЛЕТА

📖 В **обработващите МИ** ни трябва:

- 1 адрес за четене на **I операнд**;
- 2 адрес за четене на **II операнд**;
- 3 адрес за запис на получения **резултат**;
- 4 адрес на **МИ**, която трябва да **бъде изпълнена след тази МИ**.

📖 В **управляващите МИ** ни трябва:

- 1 адрес на **проверявана стойност**;
- 2 адрес на **МИ**, която ще се изпълни **при <0**;
- 3 адрес на **МИ**, която ще се изпълни **при =0**;
- 4 адрес на **МИ**, която ще се изпълни **при >0**.

КА-03

49/56

АНАЛИЗ НА МИ

КОП	адресна част			
	I операнд	II операнд	резултат	следв. МИ
8 бита	16 бита	16 бита	16 бита	16 бита

- 1 **8 бита** за КОП осигуряват кодиране на **256 операции** (128 при две дължини на данни).
- 2 За **адрес** са необходими най-малко **16 бита**, осигуряващи поне **64 KB ОП**.
- 3 **Съотношението КОП:адресна част е 1:8**.
- 4 ЦП чете МИ от ОП ⇒ **намаляването на МИ ускорява четенето**, т. е. и изпълнението.
- 5 **Полезно е да намалим адресните полета**.

КА-03

50/56

ЕЛИМИНИРАНЕ ①

КОП	I операнд	II операнд	резултат	следв. МИ
-----	-----------	------------	----------	----------------------

- 1 Няма особено **голяма логика** последователно изпълнявани **МИ да бъдат разпръснати из ОП**.
- 2 По-нормално е те да бъдат записани в **клетки с последователни адреси**.
- 3 Това показва, че **четвъртото ОП е излишно**: достатъчно е да въведем **правило за естествен ред за изпълнение на програмата** – след като се изпълни **една МИ се изпълнява тази**, която е на **следващия я адрес в ОП**.
- 4 **Цената е**: ЦП трябва да **помни до къде е стигнало изпълнението в регистър**, наречен **Програмен Брояч – ПБ (Program Counter – PC)**.

КА-03

51/56

ЕЛИМИНИРАНЕ ②

КОП	I операнд	II операнд	резултат
-----	-----------	------------	---------------------

- 1 Следващата възможност е **елиминиране на полето за запис на резултата** ($a + b + c$).
- 2 Решението е резултатът да бъде записан **на мястото на първия операнд**.
- 3 В много от случаите това е **добра стратегия**, но може да се окаже, че **унищожената стойност ще продължи участието си в изчисленията** ни.
- 4 **Цената е**: **включване на тривиалната операция за пренос** на втория операнд по адреса на първия операнд/резултат, т. е. **разширяване на машинния език с нов КОП**.

КА-03

52/56

ЕЛИМИНИРАНЕ ③

КОП	I операнд	II операнд
-----	----------------------	------------

- 1 Следва **елиминиране на полето за първи операнд/резултат** ($a + b + c$).
- 2 По-добре е **резултатът да остане в ЦП** вместо да се разхожда **до и веднага след това от ОП**.
- 3 За целта в ЦП се изработва **регистър за временно съхраняване на данни**, наречен **Акумулатор – А (Accumulator – A)**.
- 4 **Цената е**: изработване на **акумулатора и удвояване на операцията за пренос** – четене на втория операнд в **акумулатора и запис на акумулатора по адреса на втория операнд**, т. е. **ново разширяване на машинния език**.

КА-03

53/56

ЕЛИМИНИРАНЕ ④

КОП	II операнд	?
-----	-----------------------	---

- 1 Може да се **елиминира** и последното адресно поле стига местата на **операндите и резултата да се подразбират** от КОП.
- 2 **Цената е**: **твърде драстично изменение на машинния език за подразбиране** на операндите и резултата по някакъв начин.
- 3 **Съотношението КОП:адресна част падна на 1:2** (или 1:4 при 32-битов адрес).
- 4 **Вместо да се елиминира последното** оцеляло адресно поле **възможно е и да се помисли за задаване на пълен адрес от ОП с по-малък брой битове** от необходимите 16 (32).

КА-03

54/56

АДРЕСНОСТ

- 1 **Броят на адресните полета** във всички МИ на първите компютри е бил **един и същ** и тяхна основна характеристика.
- 2 Този брой се е наричал **адресност на компютъра** (три-, дву- и едно-адресни).
- 3 **Днес МИ са с различен брой адресни полета**, т. е. **за адресност на компютър** въобще **не може да се говори**.
- 4 Днес се говори за **адресност на дадена машинна инструкция** – брой на нейните адресни полета.

КА-03

55/56

**БЛАГОДАРЯ ВИ
ЗА ВНИМАНИЕТО!**

**БЪДЕТЕ С МЕН И В
СЛЕДВАЩАТА ЛЕКЦИЯ,
КОЯТО ЩЕ НИ ОТВЕДЕ
В НЕВЕРОЯТНИЯ СВЯТ НА
ЦЕНТРАЛНИЯ
ПРОЦЕСОР**