ЛЕКЦИЯ 3
ПРЕДСТАВЯНЕ НА ДАННИТЕ В ОП

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

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**.

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**:

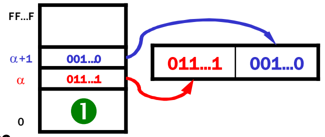
- IBM 360: полудума от адрес, кратен на 2; дума от кратен на 4; двойна дума от кратен на 8.
- IBM 370 и Интел само препоръчват такива граници за по-бърз достъп до ОП.

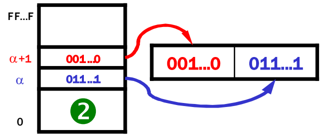
ЗАБЕЛЕЖКИ (прод.)

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.





ЦЕЛИ ЧИСЛА БЕЗ ЗНАК

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

b_{n-1}	b_{n-2}	• • •	b_1	b_0

$\times 2^{n-1} + \times 2^{n-2} + \ldots \quad \times 2^{1} + \times 2^{0}$

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

Такова тълкуване на думата се нарича **цели числа без знак**.

пренос заем

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

KA-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**.

KA-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 до отделим първия **бит за представяне на знака** (0 = +, 1 = −).

b_{n-1}	b_{n-2}	• • •	b_1	b_0
знак	абсолютна стойност на числото			

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

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

KA-03 7/56

ПРИМЕРИ: ПРАВ КОД

Представяне:

1111 → −7 (min) 0111 → +7 (max)

1000 → −0 0000 → +0

1010 → −2 0101 → +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)

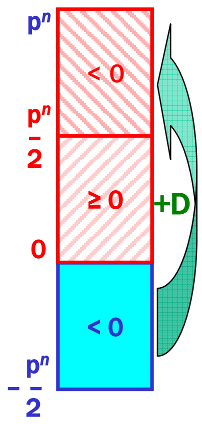
KA-03 8/56

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 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:

❶ **смяна на знака** при числата, записани в обратен код: 0101 (+5) ↔ 1010 (–5).

❷ **намиране на обратния код** на отрицателни числа: |–5| = +5 = 0101 → 1010 (–5).

❸ **разчитане на отрицателните числа**, които са записани в обратен код: 1011 = ?
→ 0100 = 4 → –4 (1011).

КА-03 11/56

ПРИМЕРИ: ОБРАТЕН КОД

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

1000 → –7 (min) 0111 → +7 (max)

1111 → –0 0000 → +0

Събиране:

0010 +2	1101 –2	0010 +2	1101 –2	1101 –2
+ 0011 +3	+ 1100 –3	+ 1100 –3	+ 0011 +3	+ 0010 +2
0 0101 +5	1 1001 –6	0 1110 –1	1 0000 +0	0 1111 –0
+ 0	+ 1	+ 0	+ 1	+ 0
0101 +5	1010 –5	1110 –1	0001 +1	1111 –0

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 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:

❶ **смяна на знака** при числа, записани в допълнителен код: 0101 (+5) ↔ 1011 (–5).

❷ **намиране на допълнителния код** на отрицателните: |–5| = +5 = 0101 → 1011.

❸ **разчитане на отрицателните числа**, които са записани в допълнителен код: 1010 = ? → 0110 = 6 → –6 (1010).

КА-03 14/56

ПРИМЕРИ: ДОПЪЛНИТЕЛЕН КОД

Диапазон: $[-2^{n-1}, +(2^{n-1}-1)]$.

1000 → –8 (min) 0111 → +7 (max)

0000 → 0 –8 няма противоположно: +8

b_{n-1}	b_{n-2}	• • •	b_1	b_0
$\times -2^{n-1}$ +	$\times 2^{n-2}$ + …		$\times 2^1$ +	$\times 2^0$

Събиране:

0010	+2	1110	–2	0010	+2	1110	–2	1110	–2
+ 0011	+3	+ 1101	–3	+ 1101	–3	+ 0011	+3	+ 0010	+2
0 0101	+5	1 1011	–5	0 1111	–1	1 0001	+1	1 0000	0

Изваждане:

1110 – 0011 = 1110 + 1101

(–2 – +3 = –2 + –3)

КА-03 15/56

ПРЕПЪЛВАНЕ ПРИ ЧИСЛА СЪС ЗНАК

❶ При **сумиране на числа с различни знаци не може** да се получи **препълване**:

$-2^{n-1}\leq a<0,\ 0\leq b<2^{n-1} \Rightarrow -2^{n-1}\leq a + b < 2^{n-1}$

❷ Наличието на **пренос не е препълване**:

1110	–2	1110	–2	0101	+5	1100	–4
+ 1101	–3	+ 0010	+2	+ 0110	+6	+ 1010	–6
1 1011	–5	1 0000	0	0 1011	–5	1 0110	+6

❸ Сигнал за препълване е получаването на **сума с различен от събираемите знак**:

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

КА-03 16/56

МОДИФИЦИРАНИ КОДОВЕ

Формулата за препълване 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 **препълване е различието на двата знакови бита на сумата**.

11110 −2	11110 −2	00101 +5	11100 −4
+ 11101 −3	+ 00010 +2	+ 00110 +6	+ 11010 −6
11011 −5	00000 0	01011 −5	10110 +6

$$прeп = s_n \oplus s_{n-1}$$

КА-03 17/56

КОД С ИЗМЕСТВАНЕ

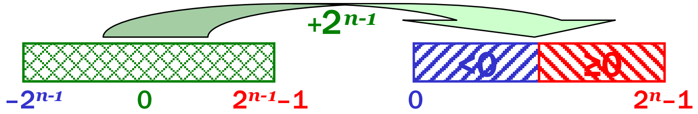
❶ Сетунь има троична симетрична: −1, 0, 1.

❷ В CDC е бил използван обратен код.

❸ Днес се използва само допълнителен код.

❹ 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 $[0, 2^n-1]$.

❺ При изместване знаковият бит е обърнат: $0 \rightarrow -$, $1 \rightarrow +$, а събирането – затруднено.



КА-03 18/56

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 19/56

ДРОБНИ ЧИСЛА

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

❶ с фиксирана запетая;
❷ с естествена запетая;
❸ с плаваща запетая.

Фиксирана запетая означава, че **мястото** на двоичната запетая **никога не се мени**.

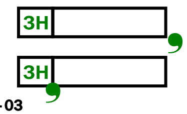
То е определено предварително, т. е. **част от битовете** на думата са **цифрите в цялата част** на числото, а **останалите** – в **дробната**.

$$\underbrace{b_{n-1}}_{\text{знак}}\ \underbrace{b_{n-2}\ \cdots\ b_f}_{\text{цяла част } (i \text{ бита})}\ \underbrace{b_{f-1}\ \cdots\ b_0}_{\text{дробна част } (f \text{ бита})}$$

КА-03 20/56

ФИКСИРАНА ЗАПЕТАЯ

❶ числата са (почти) **симетрични спрямо 0**.
❷ запетаята определя **най-малкото ≠ 0**.
❸ **диапазонът не е** особено **голям**.
❹ **покритието** (на оста) **е равномерно**.
❺ **събирането не зависи** от запетаята.
❻ **умножението зависи** от запетаята.
❼ **опасна** операция е **деленето**.
❽ **смяната на запетаята** е умножение с фиксирано число – **мащабен множител**.
❾ **най-популярни** са два избора на място:



цели числа (f=0);
правилни дроби (i=0) ∈ (–1,1).

ЕСТЕСТВЕНА ЗАПЕТАЯ

❶ Записаното число се дели на две части:
- ➲ k цифри за запис на цифрите на числото;
- ➲ $]\log_p(k+1)[$ цифри посочващи колко цифри има дробната част (изгодно е $k=p^m-1 \rightarrow m$ цифри).

❷ Така запетаята е между цифрите на числото.
❸ Събирането е затруднено.
❹ Диапазонът на числата е по-голям.
❺ Използва се главно в калкулаторите.

0025 1 → 2,5   0025 3 → 0,025   0025 0 → 25
0001 4 → 0,0001 (min ≠ 0)
9999 0 → 9999, (max)

ПЛАВАЩА ЗАПЕТАЯ

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

Нека x>0 и p≥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 **точност**та (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**).

ЗАБЕЛЕЖКИ

Представянето чрез m и e **не е еднозначно**:
11 → 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 ≠ 0).

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

СЪБИРАНЕ

❶ при равни порядъци можем да съберем мантисите.

❷ при различни порядъци трябва да денормализираме едната мантиса.

+1201 +5 : 12 010	+1201 +5	+2010 +4	+1201 +5
+2051 +5 : 20 510	+2051 +4	+2051 +4	+0205 +5
+3252 +5 : 32 520	+???? +?	+4061 +4	+1406 +5

+6201 +5	+1201 +5	+1201 +5	+0000 +5
+4051 +5	–1202 +5	+1202 –1	+1202 –1
+13252 +5	-0001 +5	+1201 +5	+0000 +5
+1325 +6	–1000 +2	$a + b = a$	+0000 -9

КА-03 25/56

ПРИМЕРИ: IBM И VAX

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

1	2 … 8	9 … 32	… 64
s	h	[0,] m (къса)	разширяване на m
1	7	56	

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

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

63	62 … 55	54 … 32	… 0
s	h	[0,1] m	разширяване на m
1	8	55 (+1!)	

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

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

63	62 … 52	51 … 0
s	h	[0,1] m
1	11	52 (+1!)

КА-03 26/56

ПРИМЕРИ: VAX И IEEE

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

127	126 … 112	111 … 0
s	h	[0,1] m (свръхдълга мантиса)
1	15	112 (+1!)

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

❺ IEEE 754 (1985 г.): p=2, 1≤$|m|$<2,

31	30 … 23	22 … 0
s	h	[1,] m
1	8	23 (+1!)

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

63	62 … 52	51 … 0
s	h	[1,] m
1	11	52 (+1!)

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

79	78 … 64	63 … 0
s	h	[1,] m
1	15	64 (+1!)

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

27/56

ОСОБЕНИ СТОЙНОСТИ

❶ VAX, h=0:
- s = 0 – представя числото 0,0 (плаваща 0);
- s = 1 – неизползвани стойности (запазени).

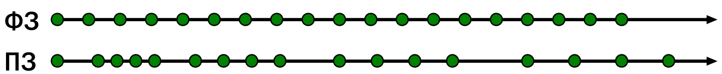
❷ IEEE 754 (ANSI):
- h=111…1 [255, 2047, 32767], $m \neq 0$: NAN (Not A Number);
- h=111…1 [255, 2047, 32767], $m = 0$: $(-1)^s\infty$ [±∞]: $a/0$;
- h=0, $m \neq 0$: денормализирана мантиса (без скритото 1,);
- h=0, m=0: $(-1)^s 0$ [±0].

❸ 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

❶ Покритие на реалната ос.



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

❸ 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

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

0011 3	0101 5	1001 9
+ 0100 4	+ 0110 6	+ 1000 8
0111 7 ☺	1011 ? ☹	0001 1 ☹
+ 0000 0	+ 0110 6	+ 0110 6
0111 7 ☺	0001 1 ☺	0111 7 ☺

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**:
❶ **неопаковани** ДКД: фиксирана стойност в левия (старшия) полубайт – $3_{(16)}$ или $F_{(16)}$, и 1 цифра;
❷ **опаковани** ДКД: цифра във всеки полубайт (2).

зонов полубайт — **цифров** полубайт

0	0	1	1	една цифра
1	1	1	1	0000÷1001

❶ неопакован формат

старша цифра	младша цифра
0000÷1001	0000÷1001

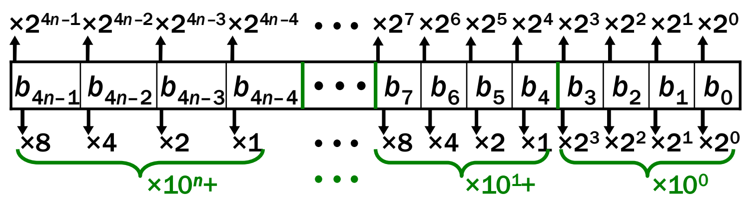
❷ опакован формат

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.

❶ Тълкуване като **двоично** число.



❷ Тълкуване като **десетично** число.

❶ 133 ⇐ 10000101 ⇒ =85 ❷

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 33/56

ЗНАКОВИ КОДОВЕ

Принципи на знаковите кодове:

❶ Десетичните цифри се кодират с последователни цели числа;

❷ Буквите се кодират съгласно азбучния им ред (но, за коя азбука говорим?).

Примери за знакови кодове са:

❶ Код на Бодо, изобретил буквопечатащия телеграф: 5 битов код (32 знака).

❷ При 12 битовата си клетка DEC (PDP–8) използва 6 битов код (64 знака).

КА-03 34/56

ЗНАКОВИ КОДОВЕ (прод.)

❸ Американски Стандартен Код за Обмен на Информация (АСКОИ) – American Standard Code for Information Interchange (ACSII): 7-битов код (128 знака), но в 8 бита.

❹ Разширен Двоично-Кодиран Десетичен Код (РДКДК) – Extended Binary Coded Decimal Information Code (EBCDIC): 8 бита (256 зн.).

❺ Разширен АСКОИ: 8 бита (256 знака).

❻ Код на Американския Национален Стандартов Институт (АНСИ) – American National Standard Institute (ANSI): 8 бита.

КА-03 35/56

ЗАБЕЛЕЖКИ

❶ 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.

❷ Днес ISO предлага нови 16- и 32-битови кодове, наречени UNICODE, кодиращи съответно 65 536 и над 4 милиарда знака.

❸ В UNICODE има място за всички азбуки: латинската, българската, гръцката, арабската, еврейската, арменската и дори(!) китайската.

КА-03 36/56

РАЗШИРЕН АСКОИ (EXTENDED ASCII)

Някои групи от знакове и техните кодове са:

❶ 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).

❷ цифри (0÷9): 47÷57 (30÷39) [F0÷F9 в EBCDIC!].

❸ главни букви латиница (A÷Z): 65÷90 (41÷5A).

❹ редовни букви латиница (a÷z): 97÷122 (61÷7A).

❺ главни букви кирилица (А÷Я): 128÷159 (80÷9F).

❻ редовни букви кирилица (а÷я): 160÷191 (A0÷BF).

КА-03 37/56

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 38/56

ЦИФРОВО ПРЕДСТАВЯНЕ НА ИЗОБРАЖЕНИЯ

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

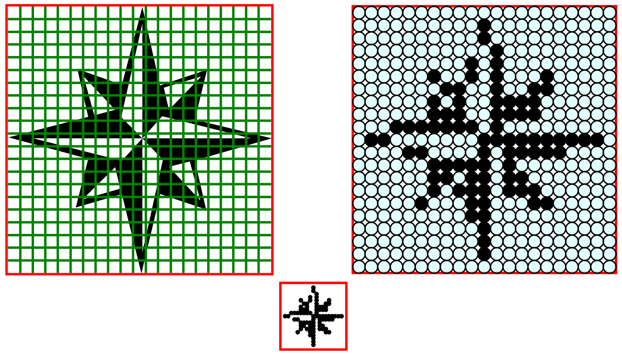
Растерният вариант експлоатира **недостатъци**те **на** човешкото **око**.

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

КА-03 39/56

ПРИМЕР: РАСТЕРЕН ФОРМАТ

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



КА-03 40/56

ПРЕДСТАВЯНЕ НА ТОЧКА

I. Монохромен формат (черно-бял):

1) с 1 бит (има – няма, черно-бяла графика);

2) с 256 оттенъка (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

❶ При 256 степенна скала (чисти цветове – true color) – 24 бита (16 777 216 цвята);

❷ При 32 степенна скала – 15 бита (32 768);

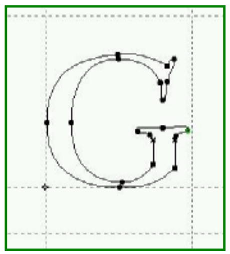
❸ При смесена скала (червено и синьо с 32, а зелено с 64 бита) – 16 бита (65 538);

❹ 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 желания цвят и свързваме две точки с крива линия.



Точките и кривите се представят с определен брой числа.

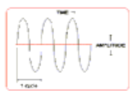
Нарича се и обектно-ориентирана графика.

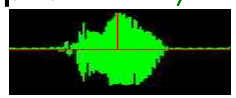
КА-03 43/56

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

❶ С 256 степени – 8-битов звук (достатъчни са за човешки говор).

❷ 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 тълкуване на думата, но вече като машинна инструкция.

КА-03 46/56

МАШИННИ ИНСТРУКЦИИ

Всяка МИ трябва да посочи две неща:

❶ какво трябва да се свърши: +, – и др.;

❷ 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.

Битовете 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:

❶ обработващи за извършване на определени пресмятания: събиране, изваждане и др.;

❷ 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 трябват:

❶ адрес за четене на **I операнд**;
❷ адрес за четене на **II операнд**;
❸ адрес за запис на получения **резултат**;
❹ адрес на **МИ, която** трябва да **бъде изпълнена след** тази МИ.

📖 В **управляващите МИ** ни трябват:

❶ адрес на **проверявана стойност**;
❷ адрес на **МИ**, която ще се изпълни **при <0**;
❸ адрес на **МИ**, която ще се изпълни **при =0**;
❹ адрес на **МИ**, която ще се изпълни **при >0**.

АНАЛИЗ НА МИ

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

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

ЕЛИМИНИРАНЕ ①

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

❶ **Няма** особено **голяма логика** последователно изпълнявани **МИ да бъдат разпръснати из ОП**.
❷ **По-нормално е** те да бъдат записани **в клетки с последователни адреси**.
❸ Това показва, че **четвъртото АП 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**.
❹ **Цената** е: **ЦП** трябва да **помни до къде е стигнало** изпълнението в регистър, наречен **Програмен Брояч** – **ПБ** (**Program Counter** – **PC**).

ЕЛИМИНИРАНЕ ②

КОП	I операнд	II операнд	~~резултат~~

❶ Следващата възможност е **елиминиране на полето за** запис на **резултата** ($a + b + c$).
❷ Решението е резултатът да бъде записан **на мястото на първия операнд**.
❸ В много от случаите това е **добра стратегия**, **но** може да се окаже, че **унищожената стойност ще продължи участието** си **в изчисленията** ни.
❹ **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**.

ЕЛИМИНИРАНЕ ③

КОП	~~I оп./рез.~~	II операнд

❶ Следва елиминиране на полето за първи операнд/резултат ($a + b + c$).

❷ По-добре е резултатът да остане в ЦП вместо да се разхожда до и веднага след това от ОП.

❸ За целта в ЦП се изработва регистър за временно съхраняване на данни, наречен Акумулатор – А (Accumulator – A).

❹ 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~~	?

❶ Може да се елиминира и последното адресно поле стига местата на операндите и резултата да се подразбират от КОП.

❷ Цената е: твърде драстично изменение на машинния език за подразбиране на операндите и резултата по някакъв начин.

❸ Съотношението КОП:адресна част падна на 1:2 (или 1:4 при 32-битов адрес).

❹ 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

❶ Броят на адресните полета във всички МИ на първите компютри е бил един и същ и тяхна основна характеристика.

❷ Този брой се е наричал адресност на компютъра (три-, дву- и едно-адресни).

❸ Днес МИ са с различен брой адресни полета, т. е. за адресност на компютър въобще не може да се говори.

❹ 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