



# RAČUNALNIŠKA ARHITEKTURA

## 5 Operandi



## 5 Operandi - cilji:

- Razumevanje različnih formatov zapisovanja operandov
  - Abecede (znaki)
  - Števila v fiksni vejici (nepredznačena, predznačena – dvojiški kompl.)
  - Števila v plavajoči vejici
  
- Razumevanje izvedbe osnovnih operacij nad operandi
  - Prednosti, slabosti zapisov, predstavitev, ...
  - Pomen standardne izvedbe operacij

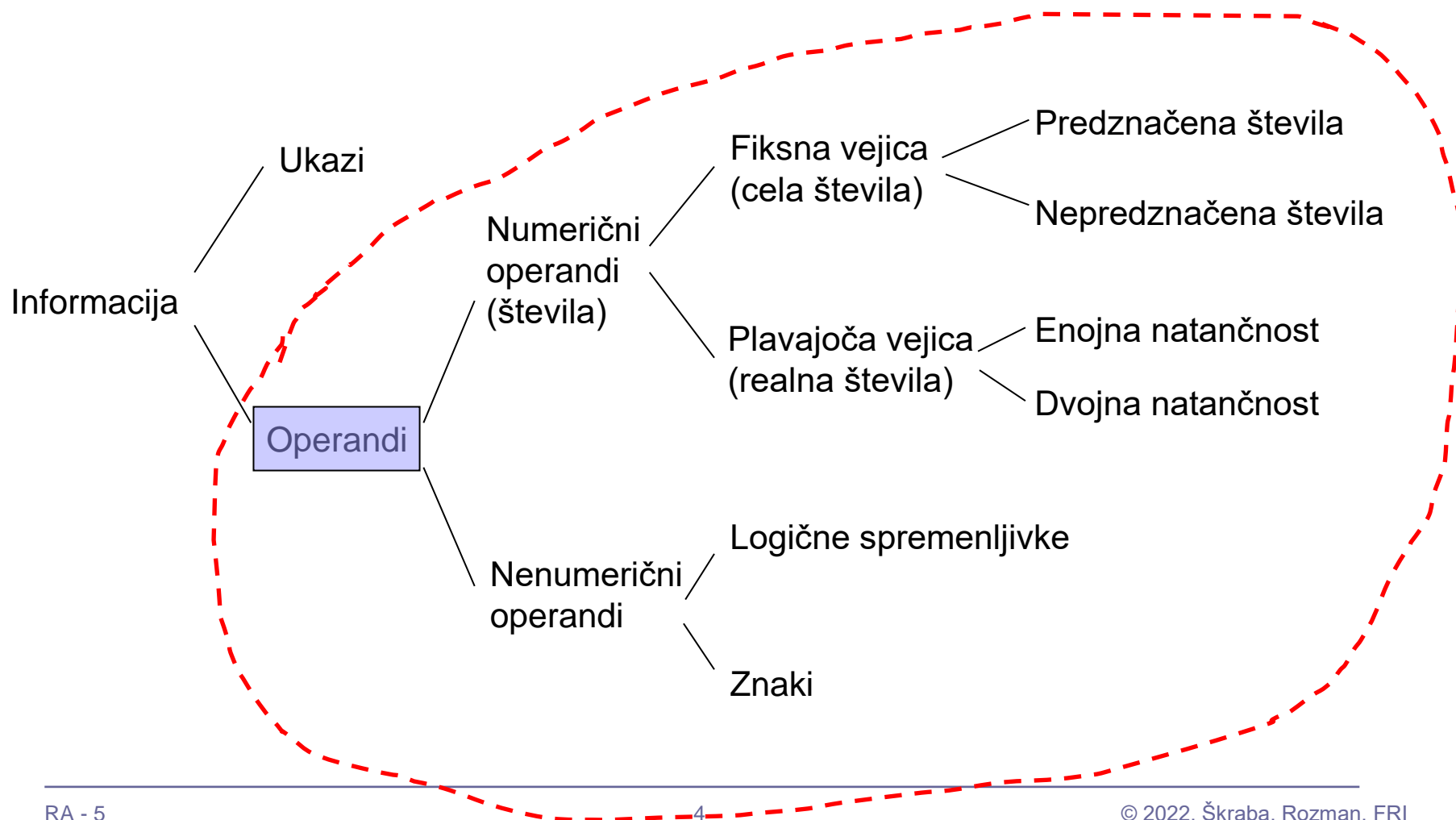


## 5 Operandi - vsebina

- Predstavitev nenumeričnih operandov
  - ASCII abeceda
  - UNICODE abeceda
- Predstavitev numeričnih operandov v fiksni vejici
  - Vrste predstavitev
  - Prenos in preliv
  - Primer-1
- Aritmetika s števili v fiksni vejici
- Predstavitev numeričnih operandov v plavajoči vejici
  - Splošna oblika
  - Standard za predstavitev v plavajoči vejici
  - Osnovne lastnosti standarda IEEE 754
  - Primer-2
- Aritmetika s števili v plavajoči vejici
- Dopolnitev standarda IEEE 754-2008



# Osnovne vrste informacije v računalniku





## Primer 32-bitne vsebine:

1110 0000 1000 0000 0101 0000 0000 0001 (bin) = E0805001 (hex)

- Zasede v 8-bitnem pomnilniku 4 zaporedne pomnilniške besede in lahko predstavlja:
  - Strojni ukaz (ARM 9): `add r5, r0, r1` /\*  $R5 \leftarrow R0 + R1$
  - Celo število brez predznaka: 3.766.505.473
  - Celo število s predznakom (dvojiški komplement): -528.461.823
  - Realno število v plavajoči vejici (enojna natančnost):  $-73,967 * 10^{18}$   
točno:  $-73,967129076026048512 * 10^{18}$
  - Štiri znake v ASCII abecedi: ř nedefiniran znak P NUL
  - Še marsikaj drugega



## 5.1 Predstavitev nenumeričnih operandov

### ■ Nenumerični operandi

- Znaki (angl. character)
- Nizi (angl. string) - sestavljeni iz znakov
- Znak je predstavljen z neko abecedo

- Abeceda je predpis, ki določa preslikavo elementov ene množice v elemente druge množice.

Why use Unicode if your program is English only?

The company I work for, as a policy, will only release software in English, even though we have customers throughout the world.

What if I want to store a customer name which uses **non-english characters**? Or the name of a place in another country?



# Vrste abeced, ki (so) se uporabljajo v računalnikih

## ■ BCD abeceda

- 6-bitna ( $2^6 = 64$  različnih znakov)
- 26 črk angleške abecede, 10 števil, 28 posebnih znakov
- V uporabi do leta 1964 (6-bitne besede)

## ■ EBCDIC abeceda (8-bitna)

- Uporabljalo predvsem podjetje IBM na velikih računalnikih (1963/64 IBM System/360 )

Danes sta bolj v uporabi **8-bitna abeceda ASCII** ter **16-bitna (in več) abeceda Unicode**



## ■ ASCII abeceda (8-bitna)

- V osnovi 7-bitna, vendar se danes v računalnikih uporablja 8-bitna oblika
  - Bit 7 = 0 - osnovna oblika
  - Bit 7 = 1 - razširjena ASCII abeceda, definiranih je dodatnih 128 znakov (IBM PC)
    - Dodatnih 128 znakov je za različne države različnih in tvorijo nacionalne ASCII abecede (npr. Latin2 = ISO 8859-2)

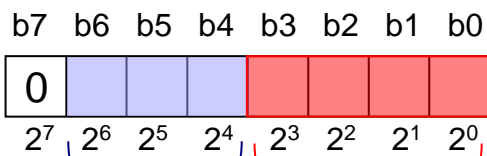
Hex	.0	.1	.2	.3	.4	.5	.6	.7	.8	.9	.A	.B	.C	.D	.E	.F
0.	NUL	SOH	STX	ETX	EOT	ENQ	ACK	BEL	BS	HT	LF	VT	FF	CR	SO	SI
1.	DLE	DC1 XON	DC2	DC3 XOFF	DC4	NAK	SYN	ETB	CAN	EM	SUB	ESC	FS	GS	RS	US
2.	SP	!	"	#	\$	%	&	'	(	)	*	+	,	-	.	/
3.	0	1	2	3	4	5	6	7	8	9	:	;	<	=	>	?
4.	@	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O
5.	P	Q	R	S	T	U	V	W	X	Y	Z	[	\	]	^	_
6.	`	a	b	c	d	e	f	g	h	i	j	k	l	m	n	o
7.	p	q	r	s	t	u	v	w	x	y	z	{		}	~	del





# Osnovna 7-bitna ASCII abeceda

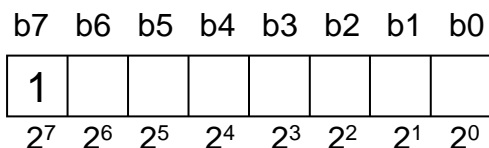
## bit7 = 0



Hex	.0	.1	.2	.3	.4	.5	.6	.7	.8	.9	.A	.B	.C	.D	.E	.F
0.	NUL	SOH	STX	ETX	EOT	ENQ	ACK	BEL	BS	HT	LF	VT	FF	CR	SO	SI
1.	DLE	DC1 XON	DC2	DC3 XOFF	DC4	NAK	SYN	ETB	CAN	EM	SUB	ESC	FS	GS	RS	US
2.	SP	!	"	#	\$	%	&	'	(	)	*	+	,	-	.	/
3.	0	1	2	3	4	5	6	7	8	9	:	;	<	=	>	?
4.	@	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O
5.	P	Q	R	S	T	U	V	W	X	Y	Z	[	\	]	^	_
6.	`	a	b	c	d	e	f	g	h	i	j	k	l	m	n	o
7.	p	q	r	s	t	u	v	w	x	y	z	{		}	~	del



# Razširjena 8-bitna ASCII abeceda Latin2 (ISO 8859-2) - dodatni znaki (b7=1)



NBSP = A0(hex) Non Breaking Space  
SHY = AD(hex) Soft Hyphen

Hex	.0	.1	.2	.3	.4	.5	.6	.7	.8	.9	.A	.B	.C	.D	.E	.F
8.	Neuporabljeno															
9.																
A.	NBSP	Ą	˘	ł	ą	Ł	Ś	§	˝	š	Ş	ř	Ž	SHY	ž	Ž
B.	°	ą	˛	ł	´	ł	ı	˘	˙	š	ş	ř	ž	˝	ž	ž
C.	Ř	Á	Â	Ă	Ä	Í	Ć	Ç	Č	É	Ę	Ë	Ě	Í	Î	Ď
D.	Đ	Ń	Ñ	Ó	Ô	Õ	Ö	×	Ř	Ű	Ú	Û	Ü	Ý		ß
E.	ř	á	â	ă	ä	í	ć	ç	č	é	ę	ë	ě	í	î	ď
F.	đ	ń	ñ	ó	ô	õ	ö	÷	ř	ű	ú	û	ü	ý	t	·



# Razširjena 8-bitna ASCII abeceda Latin2 (ISO 8859-2)

Hex	.0	.1	.2	.3	.4	.5	.6	.7	.8	.9	.A	.B	.C	.D	.E	.F
0.	NUL	SOH	STX	ETX	EOT	ENQ	ACK	BEL	BS	HT	LF	VT	FF	CR	SO	SI
1.	DLE	DC1 XON	DC2	DC3 XOFF	DC4	NAK	SYN	ETB	CAN	EM	SUB	ESC	FS	GS	RS	US
2.	SP	!	"	#	\$	%	&	'	(	)	*	+	,	-	.	/
3.	0	1	2	3	4	5	6	7	8	9	:	;	<	=	>	?
4.	@	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O
5.	P	Q	R	S	T	U	V	W	X	Y	Z	[	\	]	^	_
6.	`	a	b	c	d	e	f	g	h	i	j	k	l	m	n	o
7.	p	q	r	s	t	u	v	w	x	y	z	{		}	~	del

8.	Neuporabljeno															
9.	Neuporabljeno															
A.	NBSP	À	Á	Â	Ã	Ä	Å	Ā	Ă	Ą	Ȧ	Ş	Ţ	Ž	ŠHY	Ž
B.	°	à	á	â	ã	ä	å	ā	ă	ą	ȧ	ş	ţ	ž	š	ž
C.	Ř	Á	Â	Ä	Å	Ĺ	Ć	Ç	Č	É	Ę	Ě	Ě	Í	Î	Ď
D.	Đ	Ń	Ň	Ó	Ô	Ö	Ø	×	Ř	Ů	Ú	Ů	Ü	Ý		ß
E.	ř	á	â	ä	å	ĺ	ć	ç	č	é	ę	ě	ě	í	î	ď
F.	đ	ń	ň	ó	ô	ö	ø	+	ř	ů	ú	ů	ü	ý	ı	·



## ■ Unicode – UCS abeceda (standard ISO 10646)

- $\geq 8$ -bitna: omogoča predstavitev znakov v praktično vseh znanih svetovnih jezikih ( $2^{32}$  različnih znakov).

- UCS ravnine (Universal Coded Character Set):  $U+hhhhhh$

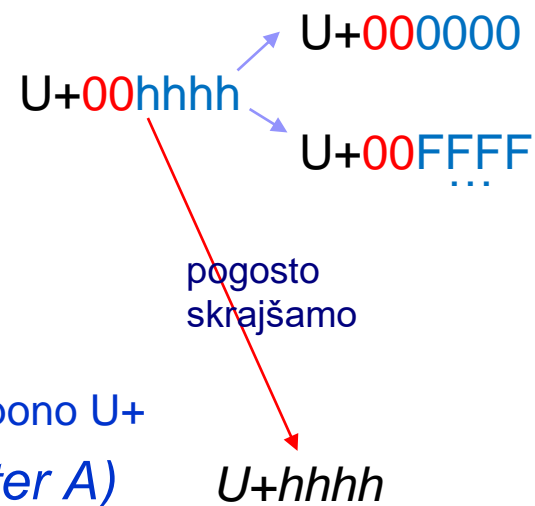
- hhhh podmnožice  $2^{16}$  znakov
- hhhh spodnjih 16 bitov + ravnina (zg. biti) hh

- BMP (Basic Multilingual Plane) ali ravnina #0:  $U+00hhhh$

- najbolj pogosto uporabljeni znaki + vsi starejši standardi
- $0x000000 \dots 0x00FFFF$

- UCS vsakemu znaku določa kodo in tudi uradno ime.

- Šestnajstiško število (UCS ali Unicode kodo), ima predpono U+
  - npr.:  $U+0041$  za znak *A* (Latin capital letter A)





## ■ Unicode – UCS abeceda (standard ISO 10646)

- $\geq$  8-bitna: omogoča predstavitev znakov v praktično vseh znanih svetovnih jezikih ( $2^{32}$  različnih znakov).
- UCS ravnine (Universal Coded Character Set): podmnožice  $2^{16}$  znakov, pri katerih se elementi (v 32-bitni predstavitvi) razlikujejo samo v spodnjih (najlažjih) 16 bitih.
- BMP (Basic Multilingual Plane) ali Plane 0: najbolj pogosto uporabljeni znaki, kjer so vključeni tudi vsi starejši standardi, so zbrani v prvi ravnini (0x000000 .. 0x00FFFF). Se pogosto skrajša v U+hhhh.
- UCS vsakemu znaku določa kodo in tudi uradno ime
  - Šestnajstiško število (UCS ali Unicode kodo), ima predpono U+
    - npr.: *U+0041* za znak *A* (Latin capital letter A)



## Definiranih je več vrst transformacij (UTF - UCS Transformation Format) za predstavitev znakov z zaporedjem bajtov

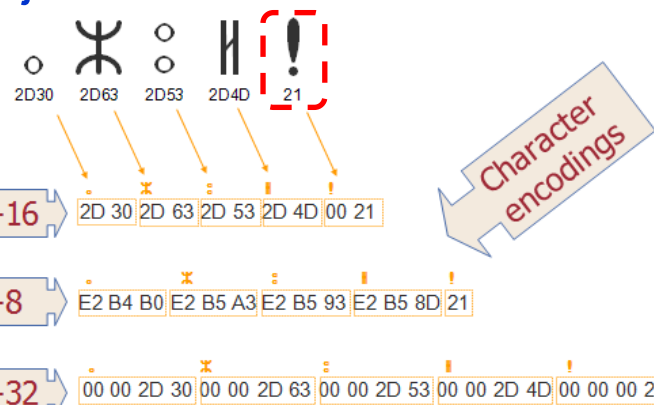
□ npr. UTF-8 in UTF-16.

### ■ UTF-16 (Windows, Java)

- En znak zasede 2 bajta (ali 4)
- Spremenljiv vrstni red bajtov (debeli/tanki konec)

### ■ UTF-8 (www, e-mail)

- Spremenljiva dolžina 1 do 4 bajte
- Kompatibilna s 7-bitno ASCII abecedo (128 znakov)



### UTF-32 (redka)

Fiksna dolžina 4 bajte

Number of bytes	Bits for code point	First code point	Last code point	Byte 1	Byte 2	Byte 3	Byte 4
1	7	U+0000	U+007F	0xxxxxxx			
2	11	U+0080	U+07FF	110xxxxx	10xxxxxx		
3	16	U+0800	U+FFFF	1110xxxx	10xxxxxx	10xxxxxx	
4	21	U+10000	U+10FFFF	11110xxx	10xxxxxx	10xxxxxx	10xxxxxx

4A E9 72 E9 6D 69 65  
 J é r é m i e LATIN1

4A C3A9 72 C3A9 6D 69 65  
 J e ' r e ' m i e UTF-8



## Predstavitev nenumeričnih operandov - Unicode

- Unicode abecedo so kot standard sprejeli IBM, Microsoft, Apple, HP, Sun, Oracle in drugi.
- Uporaba: programski jezik Java, JavaScript, XML, ...
- <http://www.unicode.org>

Znak	Unicode	UTF-16 Pravilo debelega konca	UTF-16 Pravilo tankega konca	UTF-8
Z	U+005A	005A	5A00	5A
ž	U+017D	017D	7D01	C5BD





## Primer zapisa ‚Ž‘ v UTF-8 :

- Ž (Unicode) = U+017D = 0000 0001 0111 1101

- Pravilo za transformacijo v obliko UTF-8 za znake s kodami od U+00000080 do U+000007FF je:

110XXXXX 10XXXXXX

Number of bytes	Bits for code point	First code point	Last code point	Byte 1	Byte 2	Byte 3	Byte 4
1	7	U+0000	U+007F	0xxxxxxx			
2	11	U+0080	U+07FF	110xxxxx	10xxxxxx		
3	16	U+0800	U+FFFF	1110xxxx	10xxxxxx	10xxxxxx	
4	21	U+10000	U+10FFFF	11110xxx	10xxxxxx	10xxxxxx	10xxxxxx





## Primer zapisa ‚Ž‘ v UTF-8 :

- Ž (Unicode) = U+017D = 0000 0001 0111 1101
 

↓            ↓            ↓  
 0001 0111 1101
  
- Ž (UTF-8) = 110X XXXX 10XX XXXX

Number of bytes	Bits for code point	First code point	Last code point	Byte 1	Byte 2	Byte 3	Byte 4
1	7	U+0000	U+007F	0xxxxxxx			
2	11	U+0080	U+07FF	110xxxxx	10xxxxxx		
3	16	U+0800	U+FFFF	1110xxxx	10xxxxxx	10xxxxxx	
4	21	U+10000	U+10FFFF	11110xxx	10xxxxxx	10xxxxxx	10xxxxxx



## Primer zapisa ‚Ž‘ v UTF-8 :

- Ž (Unicode) = U+017D = 0000 0001 0111 1101
- Ž (UTF-8) = 110X XXXX 10XX 1101

Number of bytes	Bits for code point	First code point	Last code point	Byte 1	Byte 2	Byte 3	Byte 4
1	7	U+0000	U+007F	0xxxxxxx			
2	11	U+0080	U+07FF	110xxxxx	10xxxxxx		
3	16	U+0800	U+FFFF	1110xxxx	10xxxxxx	10xxxxxx	
4	21	U+10000	U+10FFFF	11110xxx	10xxxxxx	10xxxxxx	10xxxxxx



# Primer zapisa ‚Ž‘ v UTF-8 :

- Ž (Unicode) = U+017D = 0000 0001 0111 1101
  - ↓ ↓ ↓
  - 0001 0111 1101
  
- Ž (UTF-8) = 110X XX01 1011 1101

Number of bytes	Bits for code point	First code point	Last code point	Byte 1	Byte 2	Byte 3	Byte 4
1	7	U+0000	U+007F	0xxxxxxx			
2	11	U+0080	U+07FF	110xxxxx	10xxxxxx		
3	16	U+0800	U+FFFF	1110xxxx	10xxxxxx	10xxxxxx	
4	21	U+10000	U+10FFFF	11110xxx	10xxxxxx	10xxxxxx	10xxxxxx



# Primer zapisa ‚Ž‘ v UTF-8 :

□ Ž (Unicode) = U+017D = 0000 0001 0111 1101

↓                    ↓                    ↓

0001 0111 1101

□ Ž (UTF-8) = 1100 0101 1011 1101

Number of bytes	Bits for code point	First code point	Last code point	Byte 1	Byte 2	Byte 3	Byte 4
1	7	U+0000	U+007F	0xxxxxxx			
2	11	U+0080	U+07FF	110xxxxx	10xxxxxx		
3	16	U+0800	U+FFFF	1110xxxx	10xxxxxx	10xxxxxx	
4	21	U+10000	U+10FFFF	11110xxx	10xxxxxx	10xxxxxx	10xxxxxx



## Primer zapisa ‚Ž‘ v UTF-8 :

□ Ž (Unicode) = U+017D = 0000 0001 0111 1101

□ Ž (UTF-8) = 1100 0101 1011 1101 = C5BD (hex)

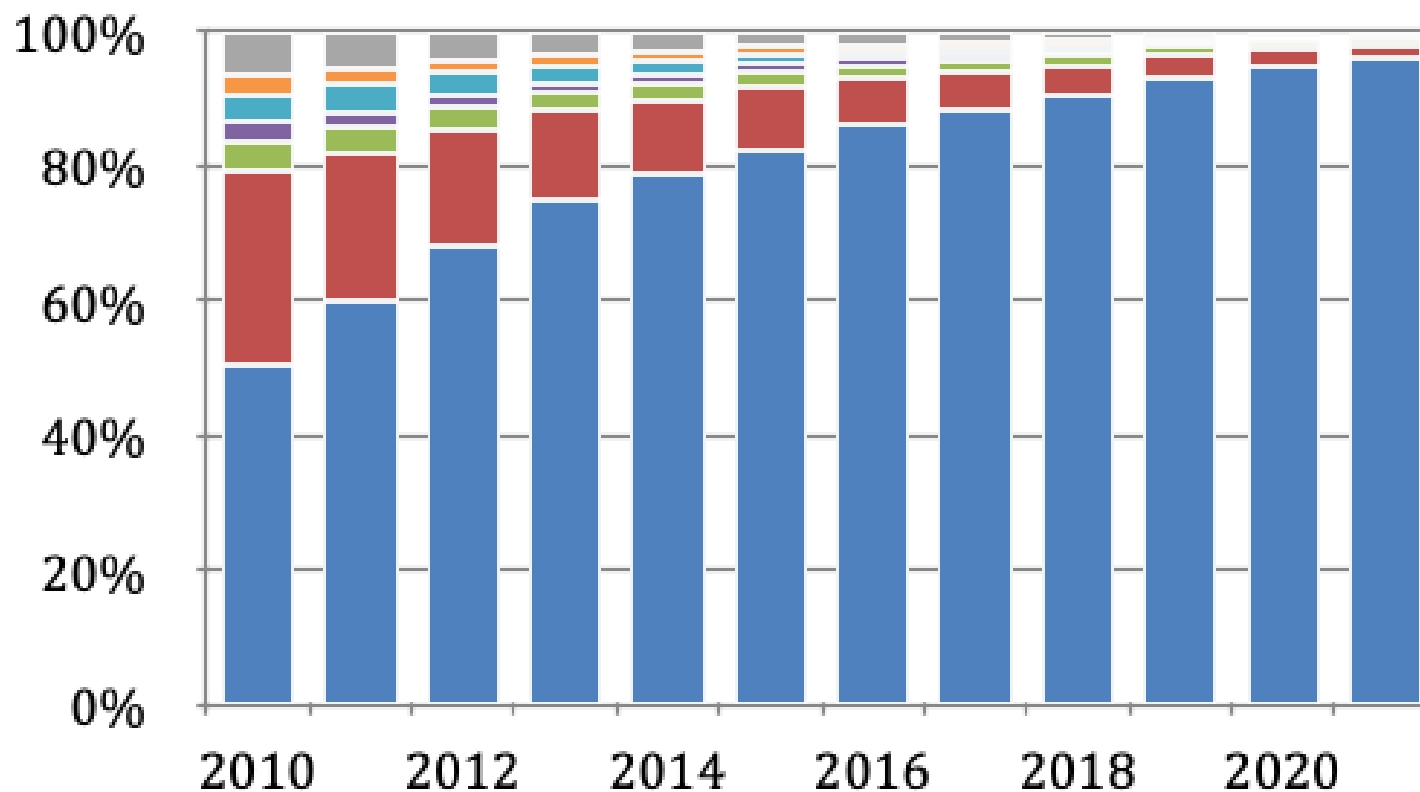
ZAPIS:        C    5        B    D

Number of bytes	Bits for code point	First code point	Last code point	Byte 1	Byte 2	Byte 3	Byte 4
1	7	U+0000	U+007F	0xxxxxxx			
2	11	U+0080	U+07FF	110xxxxx	10xxxxxx		
3	16	U+0800	U+FFFF	1110xxxx	10xxxxxx	10xxxxxx	
4	21	U+10000	U+10FFFF	11110xxx	10xxxxxx	10xxxxxx	10xxxxxx



## Predstavitev nenumeričnih operandov - Unicode

Declared character set for the 10 million most popular websites since 2010



<https://en.wikipedia.org/wiki/UTF-8>



## 5.2 Predstavitev numeričnih operandov v fiksni vejici

- Vejica je na vnaprej določenem fiksnem mestu - zapis s fiksno vejico.
- Če je vejica desno od bita z najnižjo težo, je število celo število (integer) ), sicer je lahko tudi necelo.
- Cela števila („integer“) so delno tudi sinonim za predstavitev s fiksno vejico.



## Nepredznačeno število :

- Najmanjše in največje predstavljivo nepredznačeno (pozitivno) število, ki ga lahko predstavimo z  $n$  biti je:

$$0 \leq x \leq 2^n - 1$$

- Pri 8-bitni dolžini ( $n = 8$ )

$$n = 8 \quad 0_D \leq x \leq 255_D$$

- Pri 32-bitni dolžini ( $n = 32$ )

$$n = 32 \quad 0_D \leq x \leq 4.294.967.295$$

- **Prenos** (angl. carry) - če je rezultat seštevanja ali odštevanja pozitivnih (nepredznačenih) števil izven območja, pride do prenosa iz najvišjega bita (mesta)

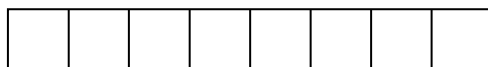




## Predznačeno število :

- Za cela števila s predznakom se uporabljajo (ali so se uporabljali) štiri načini predstavitve:
  - Predznak in velikost
  - Predstavitev z odmikom
  - Eniški komplement (v komplementu so samo negativna števila)
  - Dvojiški komplement (v komplementu so samo negativna števila)
- $n$ -bitno zaporedje  $b_{n-1} \dots b_2 b_1 b_0$  v vsakem od načinov predstavlja neko predznačeno celo število

b7 b6 b5 b4 b3 b2 b1 b0



$2^7$   $2^6$   $2^5$   $2^4$   $2^3$   $2^2$   $2^1$   $2^0$  uteži posameznih bitov

8-bitno zaporedje



## 1. Predznak in velikost :

$$V(b) = (-1)^{b_{n-1}} \sum_{i=0}^{n-2} b_i 2^i$$

- Najvišji bit je predznak (1 – negativno, 0 pozitivno število)

$$10001110_{(2)} = (-1)^1(1 \times 2^3 + 1 \times 2^2 + 1 \times 2^1) = (-1)(14) = -14_{(10)}$$

- Postopek pretvorbe iz desetiškega števila v n-bitno dvojiško število
  1. Pretvori število v dvojiško na n-1 bitov
  2. Najvišji bit nastavi glede na predznak



## 1. Predznak in velikost :

### ■ Primera:

$$-25_{(10)} = 10011001$$

$$33_{(10)} = 00100001$$

### ■ Najvišje število na 8ih bitih

$$\square 01111111_{(2)} = +127_{(10)}$$

### ■ Najmanjše število na 8ih bitih

$$\square 11111111_{(2)} = -127_{(10)}$$

### ■ Ničla

$$\square 00000000_{(2)} = +0_{(10)}$$

$$\square 10000000_{(2)} = -0_{(10)}$$



## 2. Predstavitev z **odmikom** :

**VREDNOST = ZAPIS - ODMIK**

**8b: -128 .. 127 = 0 .. 255 - 128**

$$V(b) = \sum_{i=0}^{n-1} b_i 2^i - \text{odmik}$$

- Po pretvorbi v desetiško število odštejemo odmik
  - v tem primeru **odmik**  $2^{n-1}$

$$10001110_{(2)} = (1 \times 2^7 + 1 \times 2^3 + 1 \times 2^2 + 1 \times 2^1) - (2^7) = 128 + 8 + 4 + 2 - 128 = 14_{(10)}$$



## 2. Predstavitev z **odmikom** :

### ■ Postopek pretvorbe iz desetiškega števila v n-bitno dvojiško število

1. Številu prištejemo *odmik*
2. Pretvorimo kot nepredznačeno število

$$\text{ZAPIS} = \text{VREDNOST} + \text{ODMIK}$$

### ■ Postopek pretvorbe iz n-bitnega dvojiškega števila v desetiško število

1. Pretvorimo kot nepredznačeno število
2. Številu odštejemo *odmik*

$$\text{VREDNOST} = \text{ZAPIS} - \text{ODMIK}$$



## 2. Predstavitev z **odmikom** :

$$\text{VREDNOST} = \text{ZAPIS} - \text{ODMIK}$$

$$8b: -128 .. 127 = 0 .. 255 - 128$$

### ■ Primera (odmik = $2^{n-1} = 128$ ) :

$$-26_{(10)} = 01100110$$

$$32_{(10)} = 10100000$$

### ■ Najvišje število na 8ih bitih

$$\square 11111111_{(2)} = +127_{(10)}$$

### ■ Najmanjše število na 8ih bitih

$$\square 00000000_{(2)} = -128_{(10)}$$

### ■ Ničla

$$\square 10000000_{(2)} = 0_{(10)}$$



### 3. Eniški komplement :

$$V(b) = \sum_{i=0}^{n-2} b_i 2^i - b_{n-1} (2^{n-1} - 1)$$

- Pri pretvorbi v desetiško vrednost, številu odštejemo  $2^{n-1}-1$  če je najpomembnejši bit enica

$$10001110_{(2)} = (1 \times 2^3 + 1 \times 2^2 + 1 \times 2^1) - 1 \times (2^7 - 1) = 8 + 4 + 2 - 127 = -113_{(10)}$$

- Postopek pretvorbe iz desetiškega števila v n-bitno dvojiško število
  1. Pretvorimo kot nepredznačeno število
  2. Če je število negativno, negiramo („obrnemo“) bite



### 3. Eniški komplement :

8b: -127 .. 127

#### ■ Primera

$$-25_{(10)} = 11100110$$

$$33_{(10)} = 00100001$$

#### ■ Najvišje število na 8ih bitih

$$\square 01111111_{(2)} = +127_{(10)}$$

#### ■ Najmanjše število na 8ih bitih

$$\square 10000000_{(2)} = -127_{(10)}$$

#### ■ Ničla

$$\square 00000000_{(2)} = +0_{(10)}$$

$$\square 11111111_{(2)} = -0_{(10)}$$





## 4. Dvojiški komplement :

$$V(b) = \sum_{i=0}^{n-2} b_i 2^i - b_{n-1} (2^{n-1})$$

- Pri pretvorbi v desetiško vrednost
    - odštejemo  $2^{n-1}$  če je najpomembnejši bit enica  
 $10001110_{(2)} = (1 \times 2^3 + 1 \times 2^2 + 1 \times 2^1) - 1 \times (2^7) = 8 + 4 + 2 - 128 = -114_{(10)}$
    - ali če:
      - MSb=0: pretvorimo v vrednost,
      - MSb=1: izračunamo abs. vrednost z 1'k+1 in dodamo predznak
- 
- $10001110_{(2)} \rightarrow 01110001_{(2)} \text{ (1'k)} \rightarrow 01110010_{(2)} = 114 \text{ (+1)} \rightarrow \text{vrednost} = -114_{(10)}$



## 4. Dvojiški komplement :

- Postopek pretvorbe iz desetiškega števila v n-bitno dvojiško število
  1. Pretvorimo kot nepredznačeno število
  2. Če je število **negativno obrnemo bite in prištejemo enico**
  
- Postopek pretvorbe iz n-bitnega dvojiškega števila v desetiško število
  1. Če je število **negativno, negiramo bite in prištejemo enico**
  2. Sicer interpretiramo kot nepredznačeno število (vključujoč predznak)



## 4. Dvojiški komplement :

### ■ Primera

$$-25_{(10)} = 11100111$$

$$33_{(10)} = 00100001$$

### ■ Najvišje število na 8ih bitih

$$\square 01111111_{(2)} = +127_{(10)}$$

### ■ Najmanjše število na 8ih bitih

$$\square 10000000_{(2)} = -128_{(10)}$$

### ■ Ničla

$$\square 00000000_{(2)} = 0_{(10)}$$



- **Primer-1: Katero desetiško število predstavlja 8-bitna kombinacija 10010100 v vsaki od štirih predstavitev s fiksno vejico?**

b7 b6 b5 b4 b3 b2 b1 b0

1	0	0	1	0	1	0	0
---	---	---	---	---	---	---	---

$2^7$   $2^6$   $2^5$   $2^4$   $2^3$   $2^2$   $2^1$   $2^0$  uteži posameznih bitov

Predstavitev predznak in velikost:  $b7 = 1 \Rightarrow$  število je negativno

Vrednost =  $0 \times 2^6 + 0 \times 2^5 + 1 \times 2^4 + 0 \times 2^3 + 1 \times 2^2 + 0 \times 2^1 + 0 \times 2^0 = 16 + 4 = 20(\text{dec})$

V predstavitvi predznak in velikost predstavlja ta kombinacija število -20(dec)

Predstavitev z odmikom: odmik je lahko  $2^{n-1}=128$  ali  $2^{n-1}-1=127$ ; izberemo npr. 128(dec)

Desetiška vrednost 8-bitne kombinacije 10010100 vključuje odmik in je  $128+16+4=148$

Odštejemo odmik:  $148 - 128 = 20$

V predstavitvi z odmikom 128 predstavlja ta kombinacija število +20(dec)

Predstavitev z eniškim komplementom:  $b7 = 1 \Rightarrow$  število je negativno torej je kombinacija 10010100 komplement ustreznega pozitivnega števila.

$10010100 \Rightarrow$  eniški komplement =  $01101011 = 64+32+8+2+1=107(\text{dec})$

Kombinacija 10010100 v eniškem komplementu predstavlja število -107(dec)



## Predstavitev števil v fiksni vejici – primer

Predstavitev z dvojiškim komplementom:  $b_7 = 1 \Rightarrow$  število je negativno torej je kombinacija 10010100 komplement ustreznega pozitivnega števila.

10010100  $\Rightarrow$  dvojiški komplement = 01101100 =  $64+32+8+4=108(\text{dec})$

Kombinacija 10010100 v dvojiškem komplementu predstavlja število -108(dec)

b7 b6 b5 b4 b3 b2 b1 b0

1	0	0	1	0	1	0	0
---	---	---	---	---	---	---	---

$2^7$   $2^6$   $2^5$   $2^4$   $2^3$   $2^2$   $2^1$   $2^0$

= -20(dec) v predstavitvi predznak in velikost

b7 b6 b5 b4 b3 b2 b1 b0

1	0	0	1	0	1	0	0
---	---	---	---	---	---	---	---

= +20(dec) v predstavitvi z odmikom

b7 b6 b5 b4 b3 b2 b1 b0

1	0	0	1	0	1	0	0
---	---	---	---	---	---	---	---

= -107(dec) v predstavitvi z eniškim komplementom

b7 b6 b5 b4 b3 b2 b1 b0

1	0	0	1	0	1	0	0
---	---	---	---	---	---	---	---

= -108(dec) v predstavitvi z dvojiškim komplementom



- Katero desetiško število predstavlja 8-bitna kombinacija 00010100 v vsaki od štirih predstavitev s fiksno vejico?

b7 b6 b5 b4 b3 b2 b1 b0

0	0	0	1	0	1	0	0
---	---	---	---	---	---	---	---

$2^7$   $2^6$   $2^5$   $2^4$   $2^3$   $2^2$   $2^1$   $2^0$  uteži posameznih bitov

Predstavitev predznak in velikost:  $b7 = 0 \Rightarrow$  število je pozitivno

Vrednost =  $0 \times 2^6 + 0 \times 2^5 + 1 \times 2^4 + 0 \times 2^3 + 1 \times 2^2 + 0 \times 2^1 + 0 \times 2^0 = 16 + 4 = 20(\text{dec})$

V predstavitvi predznak in velikost predstavlja ta kombinacija število +20(dec)

Predstavitev z odmikom: odmik je lahko  $2^{n-1}=128$  ali  $2^{n-1}-1=127$ ; izberemo npr. 128(dec)

Desetiška vrednost 8-bitne kombinacije 00010100 vključuje odmik in je  $16+4=20$

Odštejemo odmik  $20 - 128 = -108$

V predstavitvi z odmikom predstavlja ta kombinacija število -108(dec)

Predstavitev z eniškim komplementom:  $b7 = 0 \Rightarrow$  število je pozitivno torej kombinacija 00010100 ni komplement in lahko vrednost izračunamo direktno.

$00010100 = 16+4 = +20(\text{dec})$

Kombinacija 00010100 v eniškem komplementu predstavlja število +20(dec)



## Predstavitev števil v fiksni vejici – primer

Predstavitev z dvojiškim komplementom:  $b_7 = 0 \Rightarrow$  število je pozitivno torej kombinacija 00010100 ni komplement in lahko vrednost izračunamo direktno.

$$00010100 = 16 + 4 = +20(\text{dec})$$

Kombinacija 00010100 v dvojiškem komplementu predstavlja število +20(dec)

b7 b6 b5 b4 b3 b2 b1 b0

0	0	0	1	0	1	0	0
---	---	---	---	---	---	---	---

$2^7$   $2^6$   $2^5$   $2^4$   $2^3$   $2^2$   $2^1$   $2^0$

= +20(dec) v predstavitvi predznak in velikost

b7 b6 b5 b4 b3 b2 b1 b0

0	0	0	1	0	1	0	0
---	---	---	---	---	---	---	---

= -108(dec) v predstavitvi z odmikom

b7 b6 b5 b4 b3 b2 b1 b0

0	0	0	1	0	1	0	0
---	---	---	---	---	---	---	---

= +20(dec) v predstavitvi z eniškim komplementom

b7 b6 b5 b4 b3 b2 b1 b0

0	0	0	1	0	1	0	0
---	---	---	---	---	---	---	---

= +20(dec) v predstavitvi z dvojiškim komplementom



## Predznačeno število – obseg, preliv :

- Največje in najmanjše število, ki ga lahko z  $n$  – *biti* predstavimo v dvojiškem komplementu je:

$$-2^{n-1} \leq x \leq 2^{n-1} - 1$$

- Pri 8-bitni dolžini  $n = 8$   $-2^7 \leq x \leq 2^7 - 1$   
 $-128_D \leq x \leq +127_D$

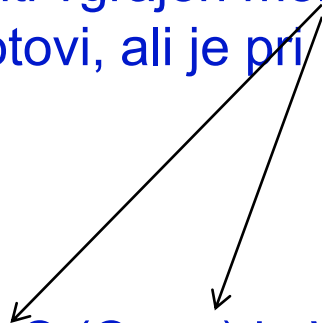
- Pri 32-bitni dolžini  $n = 32$   $-2.147.483.648_D \leq x \leq +2.147.483.647_D$

- **Preliv (angl. overflow)** - če je rezultat operacije izven področja, ki je predstavljivo v dvojiškem komplementu





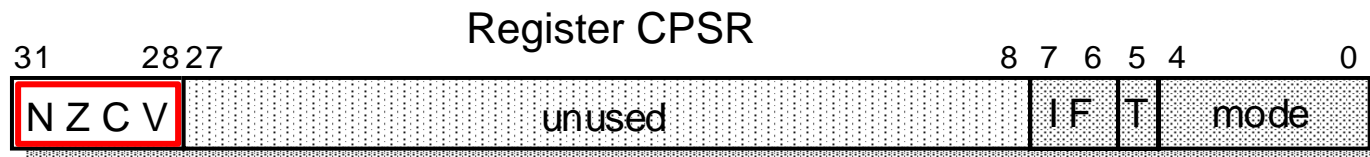
- Preliv ali prenos sta lahko vzrok za napako.
- V CPE mora biti vgrajen mehanizem, s pomočjo katerega lahko programer ugotovi, ali je pri rezultatu operacije prišlo do prenosa ali preliva.
- Biti (zastavici) C (Carry) in V (oVerflow) v registru pogojev v CPE, ki se ustrezno postavita in pokažeta ali je pri operaciji prišlo do prenosa oziroma preliva.





Primer registra pogojev (statusnega registra):

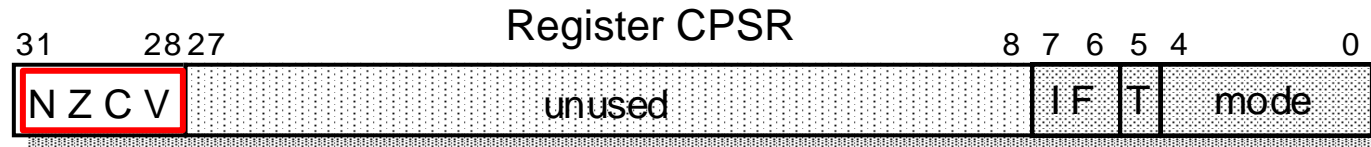
- register CPSR (Current Program Status Register) procesorja ARM9



- Biti N, Z, C in V – zastavice (flag bits, status flags)
- Biti zastavic se lahko postavijo v stanje 1 ali 0 po izvršeni aritmetični ali logični operaciji glede na rezultat operacije.



## ■ register CPSR (Current Program Status Register) procesorja ARM9



- oVerflow** (bit 28)      V = 1: pri rezultatu je prišlo do preliva;  
V = 0: ni preliva
  
- Carry** (bit 29)      seštevanje:
  - C = 1: pri rezultatu je prišlo do prenosa;
  - C = 0: ni prenosaodštevanje:
  - C = 0: pri rezultatu je prišlo do prenosa;
  - C = 1: ni prenosa
  
- Zero** (bit 30)      Z = 1: rezultat je 0;  
Z = 0: rezultat je različen od 0
  
- Negative** (bit 31)      N = 0: bit 31 rezultata je 0;  
N = 1: bit 31 rezultata je 1



# Nepredznačena in predznačena števila – primerjava 32bitov

$$0 \leq x \leq 2^n - 1$$

$$0_D \leq x \leq 4.294.967.295$$

$$-2^{n-1} \leq x \leq 2^{n-1} - 1$$

$$-2.147.483.648_D \leq x \leq +2.147.483.647_D$$



## 5.3 Aritmetika s števili v fiksni vejici

- Aritmetika - štiri osnovne operacije: seštevanje, odštevanje, množenje in deljenje.
- Aritmetične operacije se izvajajo v aritmetično-logični enoti (ALE), ki je del CPE.
- Vrsta in število operacij, ki jih zna izvajati ALE, se med računalniki razlikujeta - pri najpreprostejših samo seštevanje in logične operacije, druge operacije so realizirane s programi.



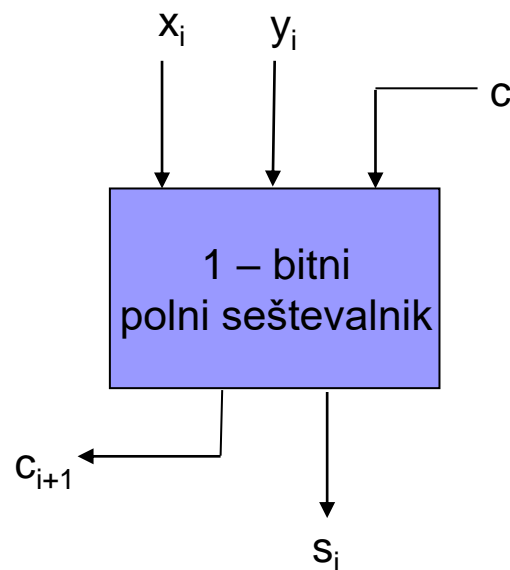
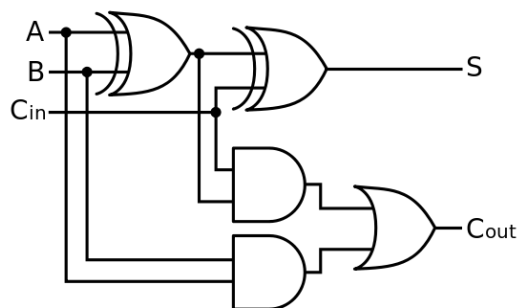
- Ključno vezje za realizacijo aritmetičnih operacij je  $n$ -bitni paralelni dvojiški seštevalnik, ki iz dveh nepredznačenih celih števil tvori njuno vsoto.
- Z njim so narejene vse operacije, tudi odštevanje (za predstavitev negativnih števil se običajno uporablja dvojiški komplement), lahko tudi množenje in deljenje (če ni posebnih enot).
- Osnovni element, s katerim zgradimo  $n$ -bitni seštevalnik, je 1-bitni polni seštevalnik.

## ■ 1-bitni polni seštevalnik ima tri vhode

- Dva sumanda  $x_i$  in  $y_i$
- Vhodni prenos  $c_i$

## ■ in dva izhoda

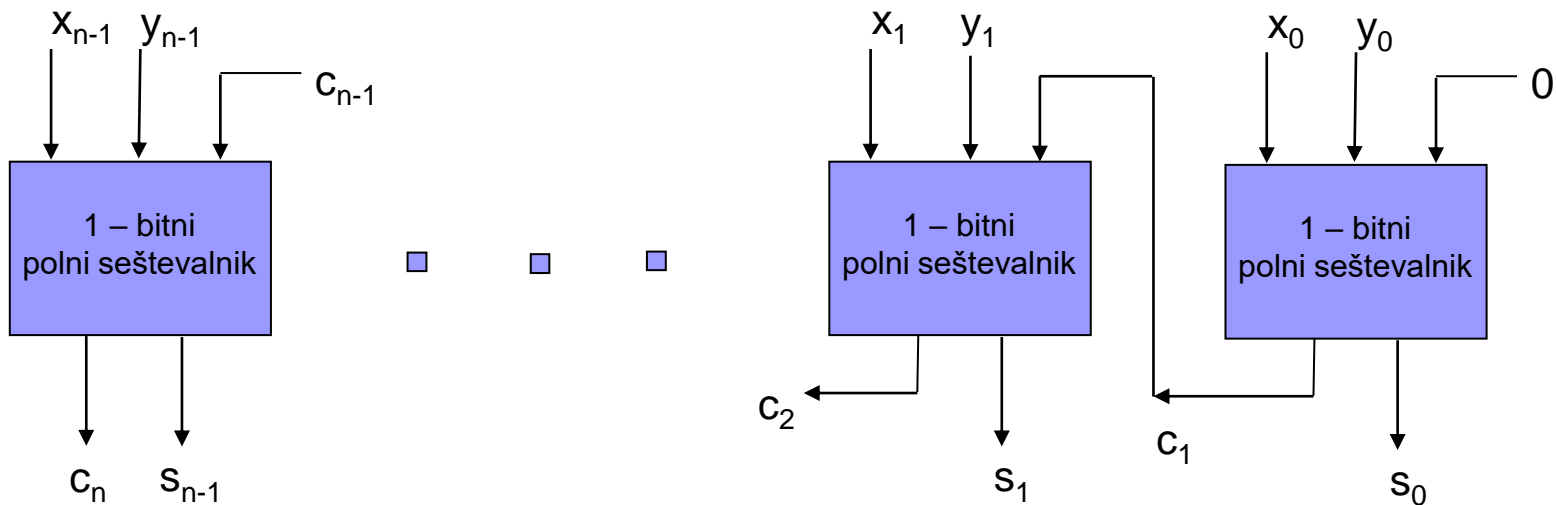
- Vsota  $s_i$
- Izhodni prenos  $c_{i+1}$



Pravilnostna tabela

Vhodi			Izhodi	
$x_i$	$y_i$	$c_i$	$s_i$	$c_{i+1}$
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

- $n$ -bitni seštevalnik dobimo, če povežemo  $n$  eno-bitnih seštevalnikov - seštevalnik s plazovitim prenosom.

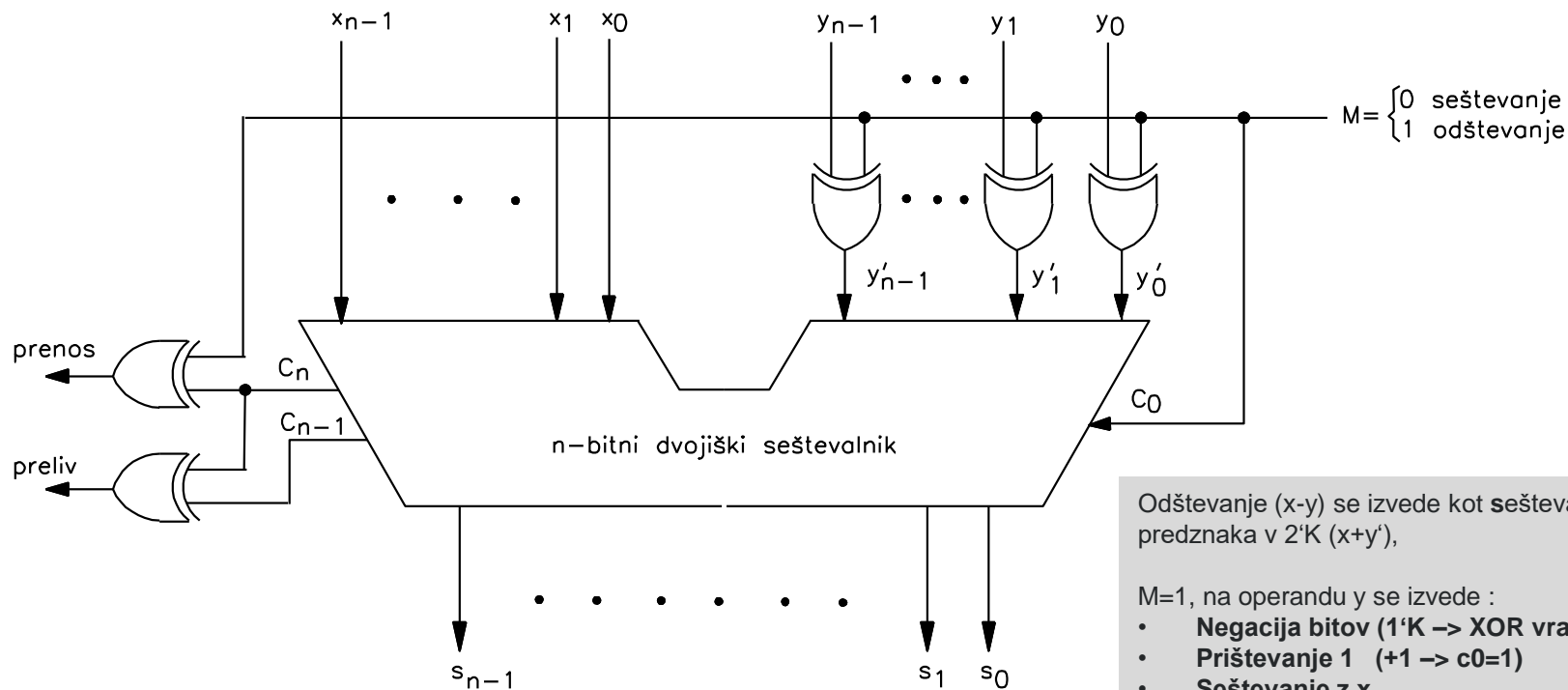


- seštevalnik s plazovitim prenosom („ripple carry“)
  - Slabost: počasnost (narašča s številom bitov)
  - Hitrejša, bolj zapletena rešitev
    - > seštevalnik z vnaprejšnjim prenosom („carry-lookahead adder“)



## Univerzalni seštevalnik

(seštevanje, odštevanje, nepredznačena in predznačena števila)



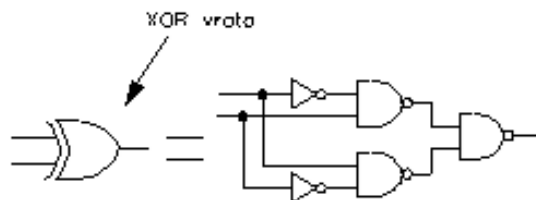
Odštevanje ( $x-y$ ) se izvede kot seštevanje s spremembo predznaka v  $2^k(x+y')$ ,

$M=1$ , na operandu  $y$  se izvede :

- **Negacija bitov** ( $1^k \rightarrow \text{XOR vrata}$ )
- **Prištevanje 1** ( $+1 \rightarrow c_0=1$ )
- **Seštevanje z  $x$**

$M=0$ , operand  $y$  nespremenjen, izvede se:

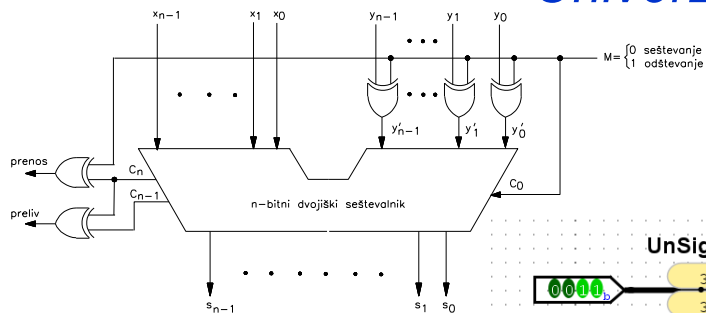
- **Seštevanje z  $x$**





# Aritmetika s števili v fiksni vejici

## Univerzalni seštevalnik - Logisim



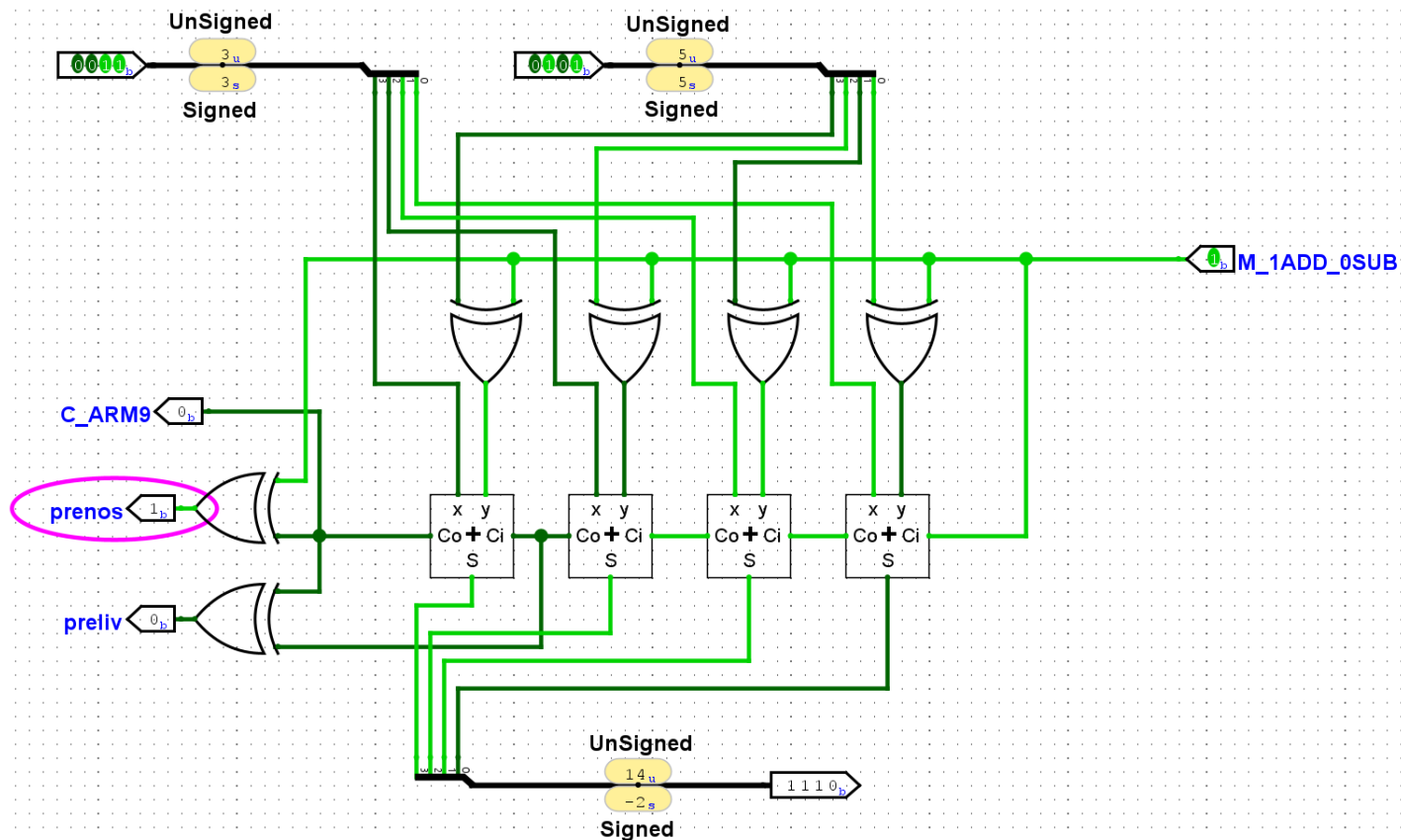
Odštevanje ( $x-y$ ) se izvede kot seštevanje s spremembo predznaka v  $2^k(x+y')$ ,

$M=1$ , na operandu  $y$  se izvede :

- Negacija bitov ( $1'K \rightarrow XOR$  vrata)
- Prištevanje 1 ( $+1 \rightarrow c0=1$ )
- Seštevanje z  $x$

$M=0$ , operand  $y$  nespremenjen, izvede se:

- Seštevanje z  $x$





## 5.4 Predstavitev numeričnih operandov v plavajoči vejici

- Obseg števil, ki jih lahko predstavimo v predstavitvi s fiksno vejico, je za tehnične probleme običajno premajhen.
- Ta števila pišemo običajno v znanstveni notaciji, ki omogoča predstavitev z razmeroma malo številkami.

$$3.200.000,00 = 3,20000000 \cdot 10^6 = 0,03200000 \cdot 10^8 = 32000000,0 \cdot 10^{-1} =$$

- Predstavitev števil v plavajoči vejici je samo za računalnik prirejena oblika znanstvene notacije.



- Splošna oblika

$$m \cdot r^e \rightarrow \text{npr.} : 0,03200000 \cdot 10^8$$

- $m$  – mantisa (koeficient, fraction, significand) = 0,03200000
- $r$  – baza (osnova, radiks) = 10
- $e$  – eksponent (karakteristika) = 8



# Standard za predstavitev v plavajoči vejici

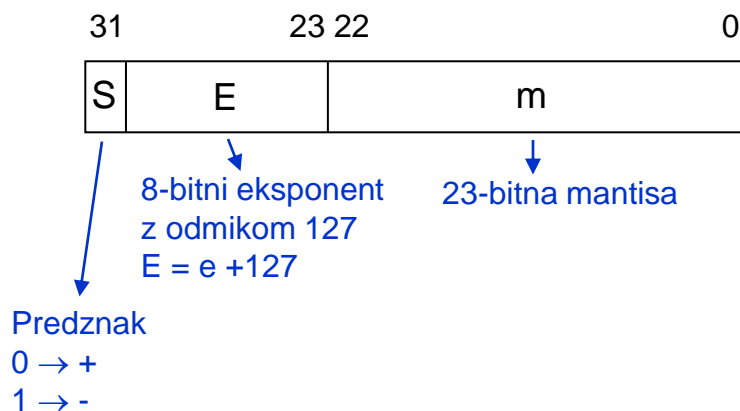
- Števila v plavajoči vejici se da predstaviti na veliko načinov:
  - različno število bitov za predstavitev mantise in eksponenta,
  - različni načini predstavitve eksponenta in mantise,
  - različni načini zaokroževanja.
- Proizvajalci računalnikov so veliko let uporabljali različne formate, ki med seboj niso bili kompatibilni. Isti program je zato na različnih računalnikih dal različne rezultate.
- Leta 1981 je bil v okviru organizacije IEEE predlagan standard za aritmetiko s plavajočo vejico, leta 1985 pa sprejet v končni obliki z oznako IEEE 754 in ga danes uporablja večina računalnikov.
- Poleg formata za predstavitev števil določa standard še načine izvajanja aritmetičnih operacij (zaokroževanje) in postopke v primeru napak (preliv, deljenje z 0, itn.).



- Osnovne lastnosti predstavitve števil v standardu IEEE 754
  - Standard uporablja bazo  $r = 2$ .
  - Mantisa je predstavljena v načinu predznak in velikost.
  - Implicitna predstavitev normalnega bita. Vejica je desno od normalnega bita (= levo od prvega bita mantise).
  - Eksponent je predstavljen v predstavitvi z odmikom.
  - Definirana sta dva formata:
    - 32-bitni format ali enojna natančnost in
    - 64-bitni format ali dvojna natančnost.



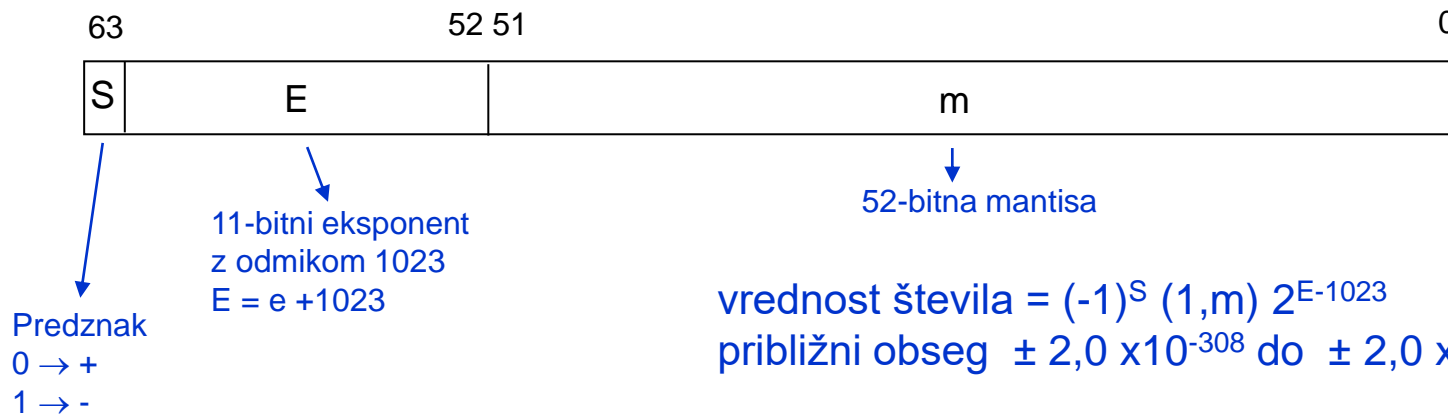
## Predstavitev števil v plavajoči vejici – standard IEEE 754 32-bitni in 64-bitni format



### 32-bitni format (enojna natančnost)

vrednost števila =  $(-1)^S (1,m) 2^{E-127}$   
približni obseg  $\pm 2,0 \times 10^{-38}$  do  $\pm 2,0 \times 10^{38}$

### 64-bitni format (dvojna natančnost)



vrednost števila =  $(-1)^S (1,m) 2^{E-1023}$   
približni obseg  $\pm 2,0 \times 10^{-308}$  do  $\pm 2,0 \times 10^{308}$



## Predstavitev števil po standardu IEEE 754

Predstavljeno število	Eksponent E	Mantisa m
Normalizirano število (1,m)	000...001 do 111...110	karkoli
Denormalizirano število (0,m)	000...000	različna od 0
Ničli $\pm 0$	000...000	000...000
Neskončnost $\pm \infty$	111...111	000...000
Neveljavno število NaN	111...111	različna od 0



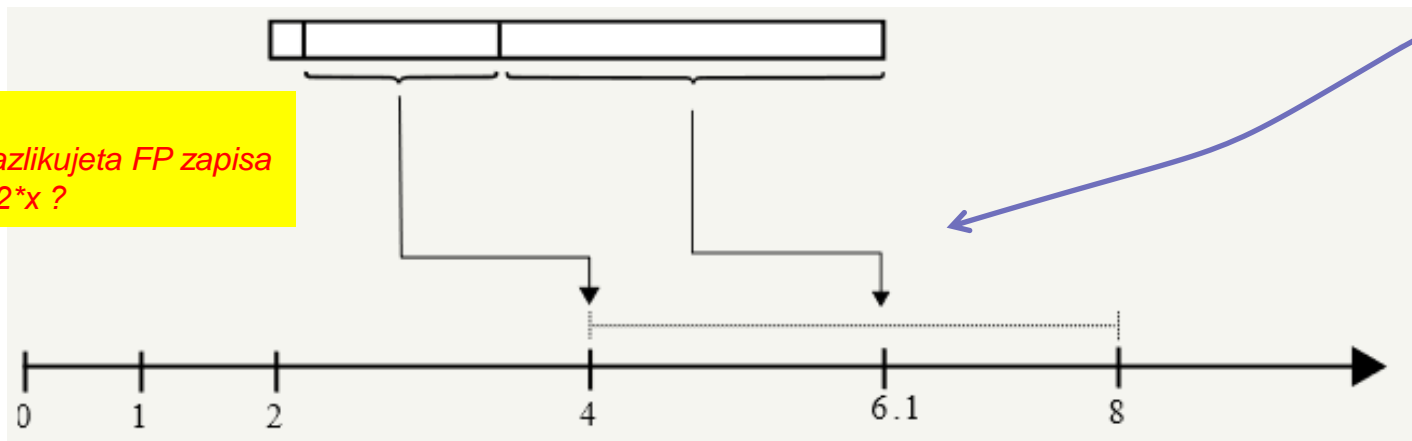
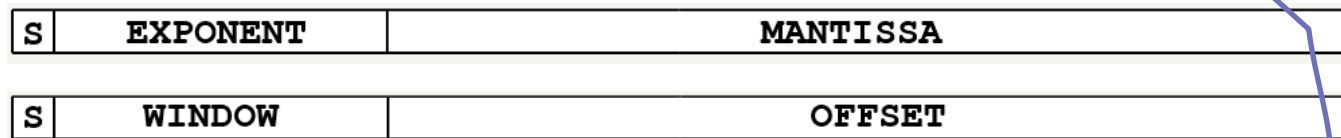


■ **Praktični poudarki v zvezi s** predstavitvijo števil v plavajoči vejici 1:

□ Vidik interpretacije :

- $1 \leq \text{Mantisa} \leq 2$ , če eksponent = 0
- $2 \leq 2 \cdot \text{Mantisa} \leq 4$ , če eksponent = 1
- $4 \leq 4 \cdot \text{Mantisa} \leq 8$ , če eksponent = 2

■ Eksponent je okno, znotraj katerega je mantisa



**IZZIV:**  
• Kako se razlikujeta FP zapisa števil  $x$  in  $2 \cdot x$  ?



# ■ Praktični poudarki v zvezi s predstavitvijo števil v plavajoči vejici 2:

□ Še vedno le končno število bitov in posledično števil: n bitov ->  $2^n$  števil

□ Omejitve (baza=2):

- $2^0 = 1$
- $2^{-1} = \frac{1}{2^1} = \frac{1}{2} = 0.5$
- $2^{-2} = \frac{1}{2^2} = \frac{1}{4} = 0.25$
- $2^{-3} = \frac{1}{2^3} = \frac{1}{8} = 0.125$
- $2^{-4} = \frac{1}{2^4} = \frac{1}{16} = 0.0625$
- $2^{-5} = \frac{1}{2^5} = \frac{1}{32} = 0.03125$

- Primer: (8.5 - 8.4)
- Primer: (0.1, 0.125)

*IZZiV: Razlaga?*

```

STEVI1=8.5
STEVI2=8.4
-----
REZ = STEVI1 - STEVI

```

Print output (drag lower right corner to resize)

```

STEVI1 = 8.50000000000000000000000000000000
-STEVI2 = 8.40000000000000000000000000000035527
-----
REZ = 0.0999999999999999999999999964473

```

```

A=0.1
B=0.125
print "A=%.30f" % (A)
print "B=%.30f" % (B)

```

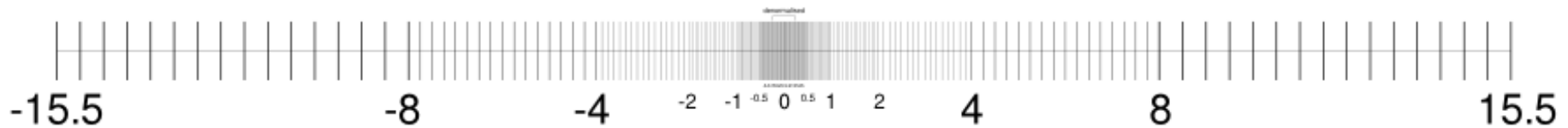
Print output (drag lower right corner to resize)

```

A=0.1000000000000000000000000000005551115123126
B=0.12500000000000000000000000000000000000000000000

```

□ Neenakomerna porazdelitev (gostota) :





# Predstavitev števil v plavajoči vejici – standard IEEE 754

## ■ Praktični poudarki v zvezi s predstavitvijo števil v plavajoči vejici 3:

### ■ Omejitve (baza=2):

$$2^0 = 1$$

$$2^{-1} = \frac{1}{2^1} = \frac{1}{2} = 0.5$$

$$2^{-2} = \frac{1}{2^2} = \frac{1}{4} = 0.25$$

$$2^{-3} = \frac{1}{2^3} = \frac{1}{8} = 0.125$$

$$2^{-4} = \frac{1}{2^4} = \frac{1}{16} = 0.0625$$

$$2^{-5} = \frac{1}{2^5} = \frac{1}{32} = 0.03125$$

Fraction	Base	Positional Notation	Rounded to 4 digits	Rounded value as fraction	Rounding error
1/10	10	0.1	0.1	1/10	0
1/3	10	0.3	0.3333	3333/10000	1/30000
1/2	2	0.1	0.1	1/2	0
1/10	2	0.00011	0.0001	1/16	3/80

### IZZIV: Razlaga?

■ Primer: (8.5 - 8.4)

```

STEV1=8.5
STEV2=8.4
REZ = STEV1 - STEV

```

Print output (drag lower right corner to resize)

```

STEV1 = 8.50000000000000000000
-STE2 = 8.40000000000000003527
-----
REZ = 0.099999999999999964473

```

■ Primer: (0.1, 0.125)

```

A=0.1
B=0.125
print "A=%.30f" % (A)
print "B=%.30f" % (B)

```

Print output (drag lower right corner to resize)

```

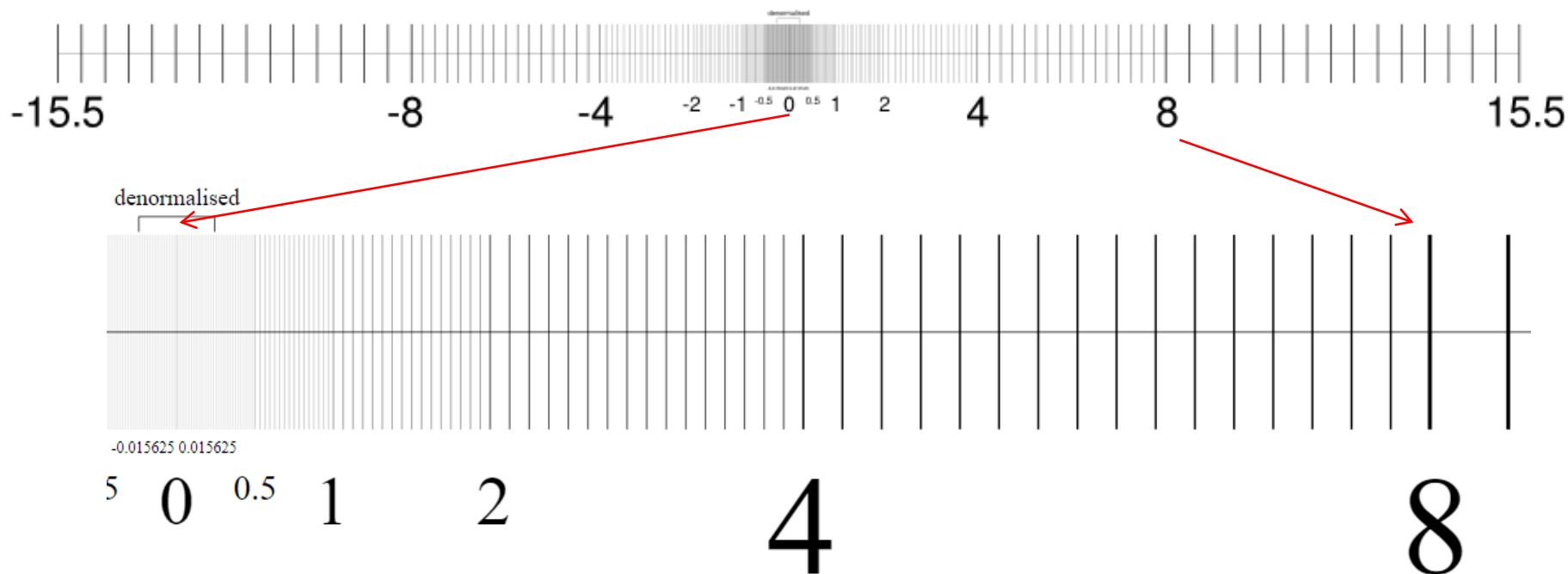
A=0.1000000000000000000000000000000000000000
B=0.1250000000000000000000000000000000000000

```



## Predstavitev števil v plavajoči vejici – standard IEEE 754

- **Praktični poudarki v zvezi s predstavitvijo števil v plavajoči vejici 4:**
  - Neenakomerna gostota po „oknih“ :
    - V vsakem „oknu“ enako število točk !
    - Primer: 8-bitni FP format (4-bitna mantisa, 3-bitni eksponent)





■ **Praktični poudarki v zvezi s predstavitvijo števil v plavajoči vejici 5:**

□ Samo okrajšani ulomki s potencami števila 2 v imenovalcu so lahko točno predstavljeni !!!!

$$2^0 = 1$$

$$2^{-1} = \frac{1}{2^1} = \frac{1}{2} = 0.5$$

$$2^{-2} = \frac{1}{2^2} = \frac{1}{4} = 0.25$$

$$2^{-3} = \frac{1}{2^3} = \frac{1}{8} = 0.125$$

$$2^{-4} = \frac{1}{2^4} = \frac{1}{16} = 0.0625$$

$$2^{-5} = \frac{1}{2^5} = \frac{1}{32} = 0.03125$$

```

STEV1=8.5
STEV2=8.4
REZ = STEV1 - STEV

```

Print output (drag lower right corner to resize)

```

STEV1 = 8.50000000000000000000
-STE2  = 8.40000000000000003527
-----
REZ    = 0.099999999999999964473

```

```

A=0.1
B=0.125
print "A=%.30f" % (A)
print "B=%.30f" % (B)

```

Print output (drag lower right corner to resize)

```

A=0.100000000000000005551115123126
B=0.125000000000000000000000000000

```

□ Nenegativni FP zapisi so primerljivi tudi kot cela števila ! *IZZIV: Zakaj?*

S	EXPONENT	MANTIŠSA
---	----------	----------

□ Seštevanje lahko bolj problematično od množenja



- Primer 1: Zapiši negativno desetiško število  $-4,625$  v predstavitvi s plavajočo vejico v enojni natančnosti.

V dvojiško obliko pretvorimo posebej celi del in posebej ulomljeni del števila (za vejico)

$$-4,625 = -(4 + 0,625)$$

$$4(\text{dec}) = 100(\text{bin}) \quad \leftarrow \begin{array}{l} 4 : 2 = 2 \text{ ostanek } 0 \quad b_0 (\text{LSB}) = 0 \\ 2 : 2 = 1 \text{ ostanek } 0 \quad b_1 = 0 \\ 1 : 2 = 0 \text{ ostanek } 1 \quad b_2 = 1 \end{array}$$

$$0,625(\text{dec}) = 0,101(\text{bin}) \quad \leftarrow \begin{array}{l} 0,625 \times 2 = 1,25 \Rightarrow 0,1 \\ 0,25 \times 2 = 0,5 \Rightarrow 0,10 \\ 0,5 \times 2 = 1,0 \Rightarrow 0,101 \\ 0,0 \times 2 = 0 \Rightarrow 0,1010 \end{array}$$

$$4,625 = 100,101 = 100,1010000\dots \quad \text{zadaj lahko dodamo poljubno število ničel}$$



Število normaliziramo tako, da vejico pomaknemo skrajno levo za prvo enico in korigiramo vrednost z množenjem s potenco števila 2

Število normaliziramo  $\Rightarrow 100,101 = 1,00101 \times 2^2$   
↑ normalni bit

Vsak pomik vejice za eno mesto levo pomeni deljenje z dva, pomik desno pa množenje z dva.

Da ohranimo vrednost števila, pomnožimo s potenco števila 2.

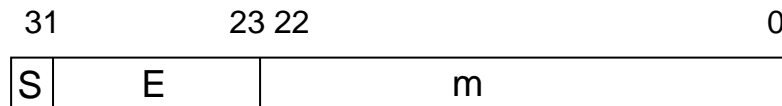
Če pomaknemo vejico za n-mest levo, pomnožimo z  $2^n$ .

Če pomaknemo vejico za n-mest desno, pa pomnožimo z  $2^{-n}$ .



## Predstavitev števil v plavajoči vejici – primer

$$- 4,625 = - 1,00101 \times 2^2$$



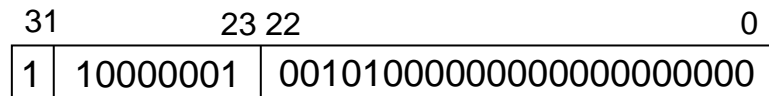
Število je negativno  $\Rightarrow S = 1$

Mantisa brez normalnega bita  $\Rightarrow m = 001010\dots 0$

Eksponent  $\Rightarrow e = 2$

Eksponent v predstavitvi z odmikom 127(dec)  $\Rightarrow E = e+127 = 2 + 127 = 129(\text{dec})$

$E = 129(\text{dec}) = 10000001(\text{bin})$



Desetiško število – 4,624 v plavajoči vejici z enojno natančnostjo





## 5.5 Aritmetika s števili v plavajoči vejici

- Aritmetika v plavajoči vejici se je v računalnikih obravnavala ločeno od aritmetike v fiksni vejici
  
- Osnovne razlike glede na operacije v fiksni vejici so:
  - Pri operacijah je treba poleg mantise uporabiti še eksponent – za te operacije je potrebna aritmetika v fiksni vejici
  
  - Zaokroževanje – rezultat operacije mora biti enak matematično točni vrednosti, ki se nato zaokroži na dolžino bitov mantise
  
  - Pri rezultatu operacije v plavajoči vejici lahko poleg preliva (overflow) pride tudi do podliva (underflow)



- Preliv (overflow), če je rezultat operacije večji kot največje predstavljivo število (eksponent je večji kot ga omogoča število bitov eksponenta)
  - Če pride do preliva, se rezultat predstavi kot  $+\infty$  ali  $-\infty$ .
  
- Podliv (underflow)
  - Pri predstavitvi števil v plavajoči vejici lahko pride tudi do podliva (underflow), če je rezultat operacije manjši kot je najmanjše predstavljivo število (ko je negativni eksponent premajhen za število bitov eksponenta).
  
  - Če pride do podliva, se število zamenja z ničlo, ali pa predstavi kot denormalizirano število.



- Čas izvedbe operacij (ukazov) za Intel Core arhitekturo (2007):  
[Kodek]

Ukaz	Latenca
ADD, SUB	1
IMUL	3
IDIV	22
FADD,FSUB	3
FMUL	5
FDIV	32
FSQRT	58
FCOS	119



■ Čas izvedbe operacij (ukazov) :

ARM Cortex M7 (STM32H750)

Table 11. Cortex<sup>®</sup>-M7 performance comparison HW SP FPU vs. SW implementation FPU with MDK-ARM<sup>™</sup> tool-chain V5.17

Frame	Zoom	Duration with HW FPU [ms]	Duration with SW implementation FPU [ms]	Ratio
0	120	134	1759	13,13
1	110	118	1519	12,87

Table 12. Performance comparison HW DP FPU versus SW implementation FPU with MDK-ARM<sup>™</sup> tool-chain V5.17

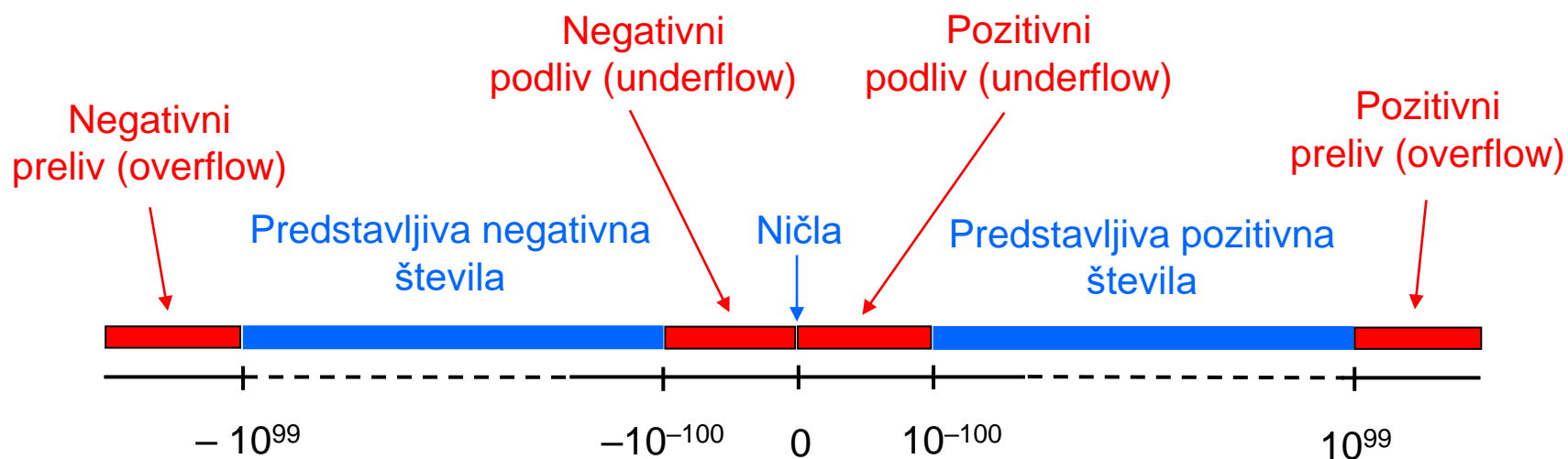
Frame	Zoom	Duration with HW DP FPU [ms]	Duration with SW implementation FPU [ms]	Ratio
0	120	408	2920	7,16
1	110	355	2523	7,11

Table 8. Some floating-point single-precision data processing instructions

Instruction	Description	Cycles
VABS.F32	Absolute value	1
VADD.F32	Addition	1
VSUB.F32	Subtraction	1
VMUL.F32	Multiply	1
VDIV.F32	Division	14
VCVT.F32	Conversion to/from integer/fixed-point	1
VSQRT.F32	Square root	14



Primer številske premice desetiških realnih števil z dvomestnim eksponentom in trimestno mantiso z območjem  $0,1 \leq |m| < 1$



Pozitivna: Min  $0.1 \cdot 10^{-99} = 10^{-100}$  Max  $0.999 \cdot 10^{99}$



## 5.6 Dopolnitve standarda IEEE 754:

( IEEE 754 → IEEE 754-2008)

- Avgusta 2008 je bil objavljen dopolnjen standard **IEEE 754-2008**, ki zamenjuje Standard IEEE 754 iz leta 1985
  - Najpomembnejše dopolnitve:
    - **Dva nova dvojiška formata z bazo  $r = 2$** 
      - 128-bitni format (štirikratna natančnost) s 112-bitno mantiso in 15-bitnim eksponentom.
      - 16-bitni format (polovična natančnost) z 10-bitno mantiso in 5-bitnim eksponentom.
    - **Dva nova desetiška formata z bazo  $r = 10$** 
      - 64-bitni format s 16 mestno mantiso (16 desetiških števil)
      - 128-bitni format s 34 mestno mantiso



- **Standard IEEE 754-2008 tako definira:**
  - **Šest osnovnih formatov**, štiri dvojiške in dva desetiška.
  - Aritmetične formate, ki se uporabljajo pri aritmetičnih in drugih operacijah.
  - Formate za izmenjavo, ki se uporabljajo pri izmenjavi operandov v plavajoči vejici
  - **Algoritme za zaokroževanje**, ki določajo metode zaokroževanja števil pri računanju in pretvorbah.
  - **Aritmetične in druge operacije** nad aritmetičnimi formati.
  - Obravnavo **izjemnih dogodkov** (deljenje z 0, preliv, podliv, ...).
  
- **Najnovejša dopolnitev : IEEE 754-2019**



## Predstavitev števil v plavajoči vejici – dopolnjen standard IEEE 754-2008

Oznaka	Ime	Osnova	Število mest mantise *	E min	E max	Desetiška natančnost	Max desetiški eksponent
binary32	Enojna natančnost	2	23+1	-126	+127	7,22	38,23
binary64	Dvojna natančnost	2	52+1	-1022	+1023	15,95	307,95
binary128	Štirikratna natančnost	2	112+1	-16382	+16383	34,02	4931,77
decimal64		10	15+1	-383	+384	16	384
decimal128		10	33+1	-6143	+6144	34	6144

\* mantisa + 1 bit za predznak