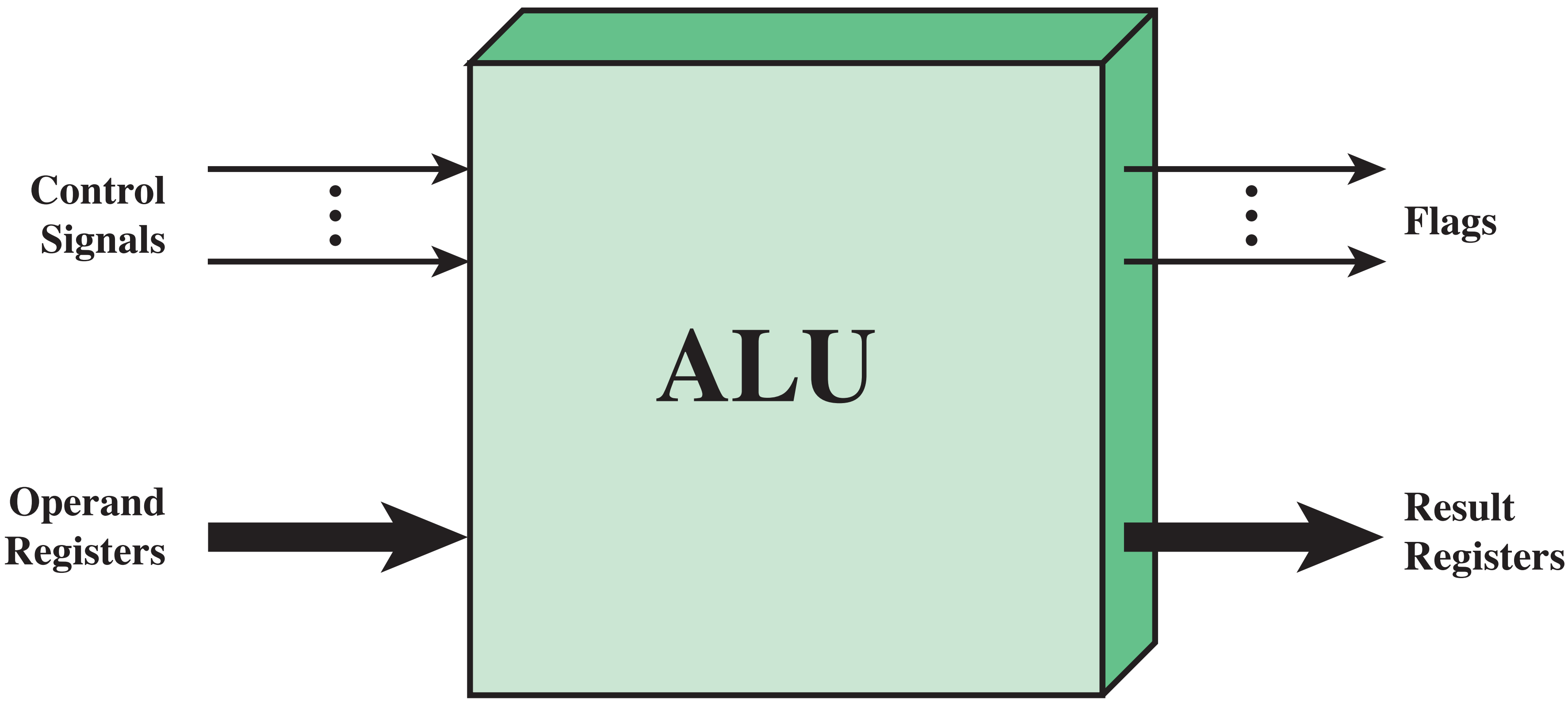
Računalniška aritmetika
Computer Arithmetics

Aritmetično logična enota (ALE) / Arithmetic & Logic Unit (ALU)

- Izvaja računske operacije
- Vse ostale enote v računalniku strežejo tej enoti
- Temelji na uporabi preprostih digitalnih logičnih naprav, ki lahko shranjujejo dvojiške števke in izvajajo preproste Booleanove logične operacije
- Operira nad celimi števili (angl. integer)
  - Lahko dela tudi z realnimi (plavajoča vejica – angl. floating point) števili
  - Lahko obstaja kot ločena enota za delo z realnimi števili (FPU) – npr. matematični koprocesor (486DX+)

- Part of the computer that actually performs arithmetic and logical operations on data
- All of the other elements of the computer system are there mainly to bring data into the ALU for it to process and then to take the results back out
- Based on the use of simple digital logic devices that can store binary digits and perform simple Boolean logic operations
- Operates over integers
  - Can deal with real (floating point) numbers
  - It can exist as a separate unit to deal with real numbers (FPU) – e.g. mathematical coprocessor (486DX +)


Control
Signals
Operand
Registers
ALU
Flags
Result
Registers

Predstavitev celih števil / Integer Representation

- V sistemu binarnih števil lahko poljubna števila predstavljamo s:
  - Števkami 0 in 1
    npr. 41 = 00101001
  - Minusom za negativna števila
  - Vejico za realna števila.
  - $-13.3125_{10} = -1101.0101_2$
- A hkrati na računalniku nimamo posebnega znaka za minus ali decimalno vejico
- Za prikaz števil lahko uporabljajo samo binarne števke (0,1)

- In the binary number system arbitrary numbers can be represented with:
  - The digits zero and one
    e.g. 41 = 00101001
  - The minus sign (for negative numbers)
  - The period, or ***radix point*** (for numbers with a fractional component)
  - $-13.3125_{10} = -1101.0101_2$
- For purposes of computer storage and processing we do not have the benefit of special symbols for the minus sign and radix point
- Only binary digits (0,1) may be used to represent numbers

2 predstavitvi / 2 representations

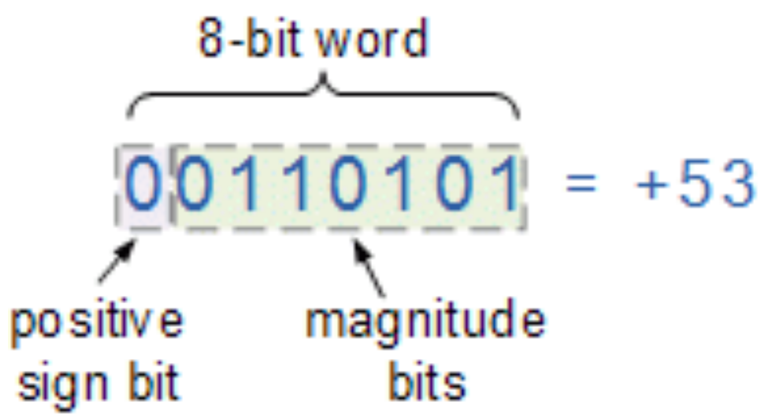
1. Predznak-velikost
2. Dvojiški complement
3. Fiksna vejica

1. Sign-magnitude
2. Twos Complement
3. Fixed point

Predznak-velikost / Sign-Magnitude Representation

- Najbolj levi bit predstavlja bit predznaka
  - 0 pomeni pozitivno
  - 1 pomeni negativno
    - +18=00010010
    - -18 = 10010010

- The left most bit represents the bit of the sign
  - 0 means positive
  - 1 means negative
    - + 18 = 00010010
    - -18 = 10010010



Zapiši število v P-V / Write the number in S-M

-6_{10}

123_{10}

Uporaba predznak-velikost / Usage of sign-magnitude

- Težave
  - Pri računanju je potrebna ločena obravnava predznaka in velikosti števila
  - Dve predstavitvi ničle: +0 in -0

- Zaradi teh pomanjkljivosti se pri računanju celih števil ALU redko uporablja izvedba predznak+velikost.

- Problems
  - Addition and subtraction require a consideration of both the signs of the numbers and their relative magnitudes to carry out the required operation
  - There are two representations of 0

- Because of these drawbacks, sign-magnitude representation is rarely used in implementing the integer portion of the ALU.

Splošna predstavitev P-V/General representation of S-M

$$A = \begin{cases} \sum_{i=0}^{n-2} 2^i a_i & \text{if } a_{n-1} = 0 \\ -\sum_{i=0}^{n-2} 2^i a_i & \text{if } a_{n-1} = 1 \end{cases}$$

Dvojiški komplement / Twos complement

- Za zapis negativnega celega števila v dvojiškem komplemetu, napišemo številko v dvojišk notaciji, invertamo in prištejemo 1.
- Primer -28

- To get the two's complement negative notation of an integer, we write out the number in binary then invert the digits, and add one to the result.
- Example -28

00011100

0 postanejo 1 in obratno

0 become 1 and vice versa

11100011

prištejemo 1

we add 1 to the result

11100100

Obratni postopek / Backward process

- Obratni postopek
- Backward process

11100100

0 postanejo 1 in obratno

0 become 1 and vice versa

00011011

prištejemo 1

we add 1 to the result

00011100

Splošna predstavitev DK / General representation of TC

$$A = -2^{n-1}a_{n-1} + \sum_{i=0}^{n-2} 2^i a_i$$

Dokaz/proof: A + Twos complement(A) = 0

$$B = -2^{n-1}\overline{a_{n-1}} + 1 + \sum_{i=0}^{n-2} 2^i \overline{a_i}$$

Značilnosti dvojiškeg dvojiškega komplementa in aritmetike

Characteristics of Twos Complement Representation and Arithmetic

Range	-2_{n-1} through $2_{n-1} - 1$
Number of Representations of Zero	One
Negation	Take the Boolean complement of each bit of the corresponding positive number, then add 1 to the resulting bit pattern viewed as an unsigned integer.
Expansion of Bit Length	Add additional bit positions to the left and fill in with the value of the original sign bit.
Overflow Rule	If two numbers with the same sign (both positive or both negative) are added, then overflow occurs if and only if the result has the opposite sign.
Subtraction Rule	To subtract *B* from *A*, take the twos complement of *B* and add it to *A*.

Alternative Representations for 4-Bit Integers

Decimal Representation	Sign-Magnitude Representation	Twos Complement Representation	Biased Representation
+8	—	—	1111
+7	0111	0111	1110
+6	0110	0110	1101
+5	0101	0101	1100
+4	0100	0100	1011
+3	0011	0011	1010
+2	0010	0010	1001
+1	0001	0001	1000
+0	0000	0000	0111
–0	1000	—	—
–1	1001	1111	0110
–2	1010	1110	0101
–3	1011	1101	0100
–4	1100	1100	0011
–5	1101	1011	0010
–6	1110	1010	0001
–7	1111	1001	0000
–8	—	1000	—

Biased
or
offset binary
or
excess code

Predstavitev z
odmikon

–128	64	32	16	8	4	2	1

(a) An eight-position two's complement value box

–128	64	32	16	8	4	2	1
1	0	0	0	0	0	1	1

–128 +2 +1 = –125

(b) Convert binary 10000011 to decimal

–128	64	32	16	8	4	2	1
1	0	0	0	1	0	0	0

–120 = –128 +8

(c) Convert decimal –120 to binary

Razširitev obsega P-V / Range Extension S-M

- Obseg števil, ki jih lahko izrazimo, se poveča s povečanjem dolžine bitov
- V zapisu P-V se to doseže s premikanjem predznaka na skrajno levo lego in z dopolnjevanjem ničel

- Range of numbers that can be expressed is extended by increasing the bit length
- In sign-magnitude notation this is accomplished by moving the sign bit to the new leftmost position and fill in with zeros

$+18$	$=$	00010010	(sign magnitude, 8 bits)
$+18$	$=$	0000000000010010	(sign magnitude, 16 bits)
-18	$=$	10010010	(sign magnitude, 8 bits)
-18	$=$	1000000000010010	(sign magnitude, 16 bits)

Razširitev obsega DK / Range Extension TC

- Ta postopek ne bo deloval pri dvojiškem komplementu za negativna cela števila
  - Pravilo je, da premaknemo bit predznaka na novo skrajno levo lego in
  - Za pozitivna števila vnesemo ničle, za negativna števila pa enice
  - To imenujemo razširitev predznaka

- This procedure will not work for twos complement negative integers
  - Rule is to move the sign bit to the new leftmost position and fill in with copies of the sign bit
  - For positive numbers, fill in with zeros, and for negative numbers, fill in with ones
  - This is called *sign extension*

+18	=	00010010	(twos complement, 8 bits)
+18	=	0000000000010010	(twos complement, 16 bits)
−18	=	11101110	(twos complement, 8 bits)
−32,658	=	1000000001101110	(twos complement, 16 bits)

−18	=	1111111111101110	(twos complement, 16 bits)

Dokaz / Proof

$$A = -2^{n-1}a_{n-1} + \sum_{i=0}^{n-2} 2^i a_i$$

Razširitev obsega

Extended range

$$A = -2^{m-1}a_{m-1} + \sum_{i=0}^{m-2} 2^i a_i$$

$$m > n$$

A = razširjen obseg (A)

A = extended range (A)

ko

when

$$a_{m-2} = \ldots = a_n = a_{n-1} = 1$$

Fixed-Point Representation

- Položaj decimalne vejice je fiksen in se predpostavlja, da je desno od skrajne desne števke
- Programer lahko uporabi isto predstavitev za binarne ulomke z množenjem števila tako da je dvojiška vejica implicitno nameščena na kakšni drugi lokaciji

- The radix point (binary point) is fixed and assumed to be to the right of the rightmost digit
- Programmer can use the same representation for binary fractions by scaling the numbers so that the binary point is implicitly positioned at some other location

Aritmetika celih števil / Integer arithmetic

- Negacija
- Seštevanje in odštevanje
- Množenje
- Deljenje

- Negation
- Addition and subtraction
- Multiplication
- Division

Negacija / Negation

- Dvojiško komplement
  - Naredimo negacijo po bitih (vključno s predznakom)
  - Rezultat obravnavamo kot nepredznačeno celo število in mu dodamo 1

- Twos complement operation
  - Take the Boolean complement of each bit of the integer (including the sign bit)
  - Treating the result as an unsigned binary integer, add 1

+18 = 00010010 (twos complement)
bitwise complement = 11101101
+ 1
11101110 = -18

- Negativ negativnega števila je število sam0:

- The negative of the negative of that number is itself:

-18 = 11101110 (twos complement)
bitwise complement = 00010001
+ 1
00010010 = +18

Negacija - poseben primer (1) / Negation Special case 1

0	=	00000000	(twos complement)
Bitwise complement	=	11111111	
Add 1 to LSB		+ 1	
Result		100000000	

Carry out (overflow) bite is ignored, so:

- 0 = 0

Negacija - poseben primer (2) / Negation Special case 2

-128	=	10000000	(twos complement)
Bitwise complement	=	01111111	
Add 1 to LSB		+ 1	
Result		10000000	

So:

-(-128) = -128 X

Monitor MSB (sign bit)

It should change during negation

Seštevanje števil v dvojiškem komplementu

Addition of numbers in twos comoplement representation

1001 = –7 +0101 = 5 1110 = –2 (a) (–7) + (+5)	1100 = –4 +0100 = 4 10000 = 0 (b) (–4) + (+4)
0011 = 3 +0100 = 4 0111 = 7 (c) (+3) + (+4)	1100 = –4 +1111 = –1 11011 = –5 (d) (–4) + (–1)
0101 = 5 +0100 = 4 1001 = Overflow (e) (+5) + (+4)	1001 = –7 +1010 = –6 10011 = Overflow (f) (–7) + (–6)

Pravilo prekoračitve / OVERFLOW RULE

- Pri seštevanju dveh pozitivnih ali dveh negativnih števil, se prelivanje pojavi le, če ima rezultat nasproten predznak.

- If two numbers are added, and they are both positive or both negative, then overflow occurs if and only if the result has the opposite sign.

Pravilo odštevanja / Subtraction Rule

- Če želimo odšteti eno številko (odštevanec) od druge (zmanjševanec), vzememo dvojiški komplement odštevanca in ga prištejemo k zmanjševancu.

- To subtract one number (subtrahend) from another (minuend), take the twos complement (negation) of the subtrahend and add it to the minuend.

Primeri: odštevanje števil v dvojiškem komplementu

Examples: subtraction of numbers in twos comoplement representation

```
  0010 =  2
 +1001 = -7
  1011 = -5

(a) M = 2 = 0010
    S = 7 = 0111
   -S =     1001
```

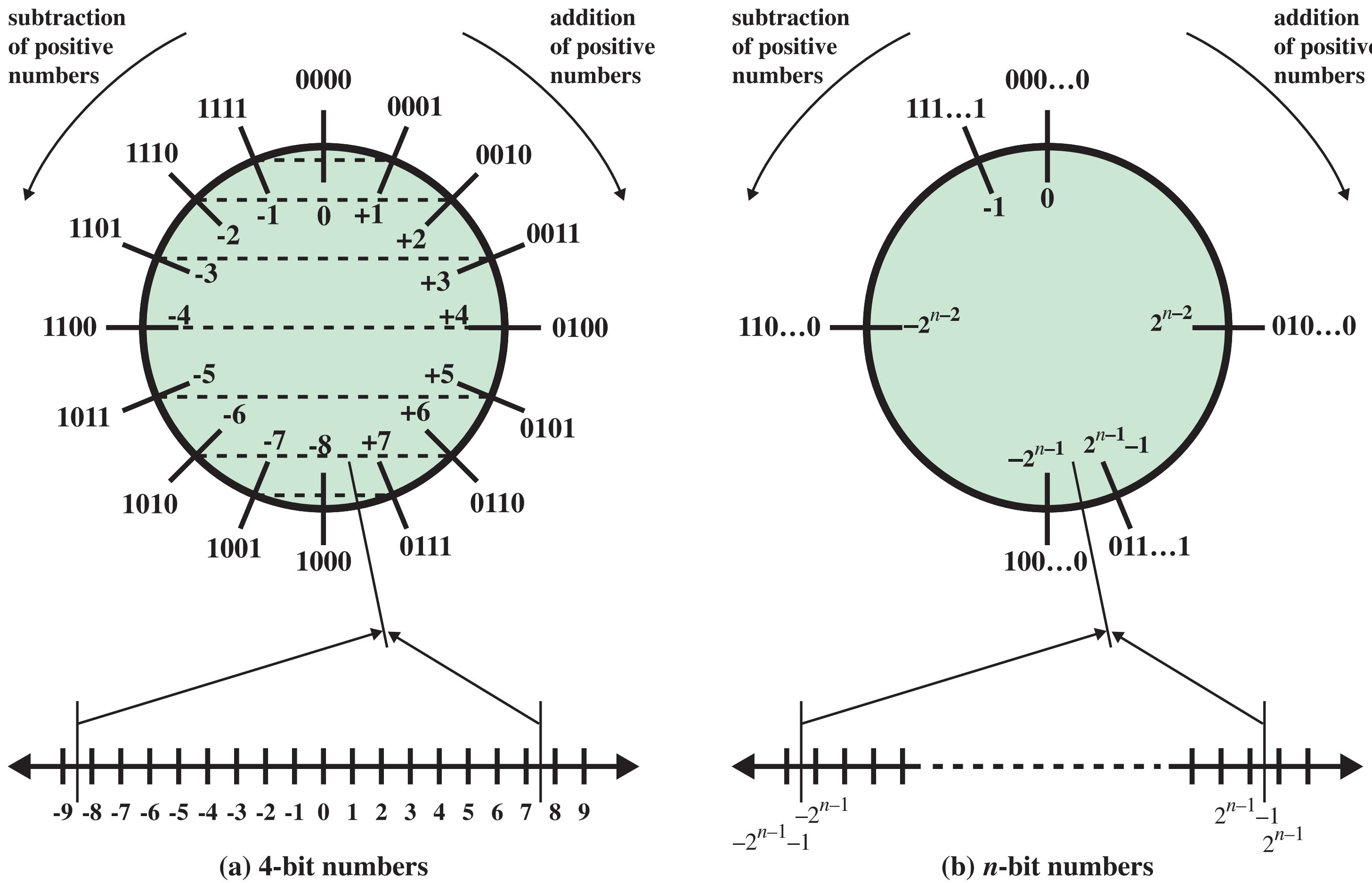
```
  0101 =  5
 +1110 = -2
 10011 =  3

(b) M = 5 = 0101
    S = 2 = 0010
   -S =     1110
```

```
  1011 = -5
 +1110 = -2
 11001 = -7

(c) M =-5 = 1011
    S = 2 = 0010
   -S =     1110
```

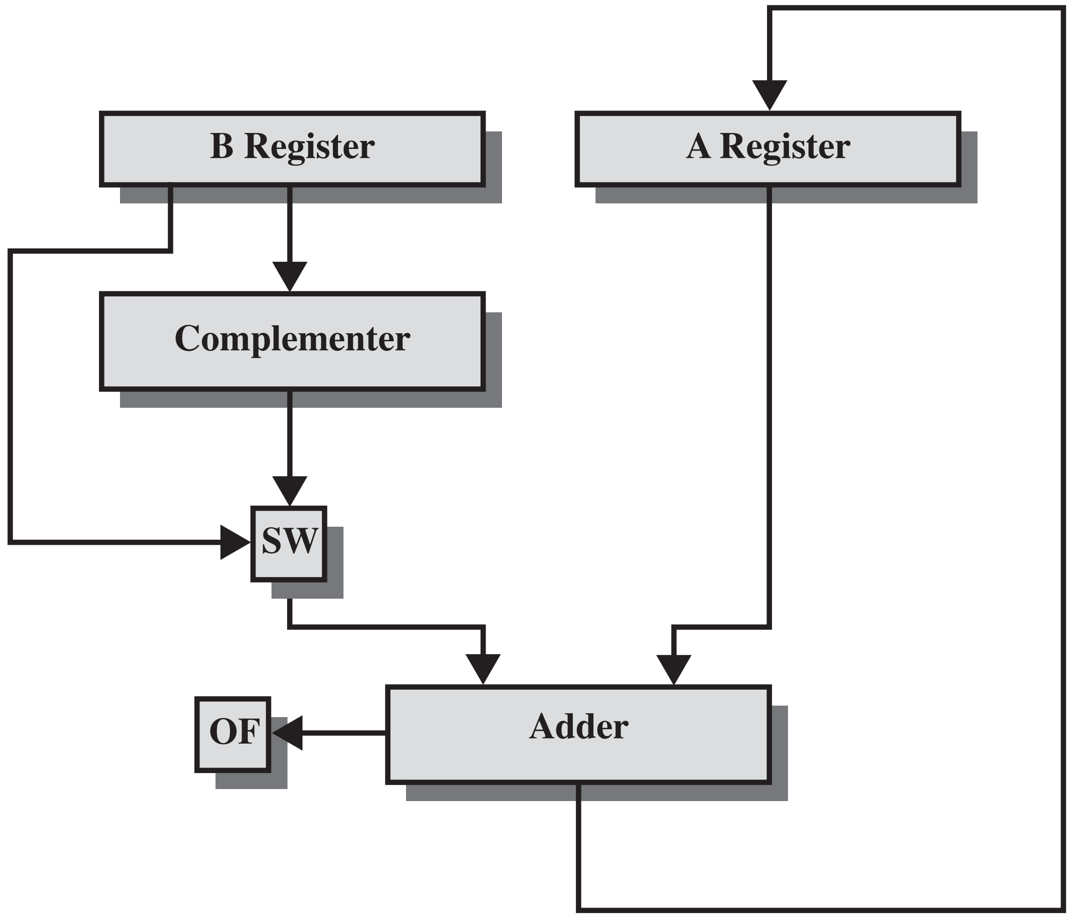
```
  0101 = 5
 +0010 = 2
  0111 = 7

(d) M = 5 = 0101
    S =-2 = 1110
   -S =     0010
```

```
  0111 = 7
 +0111 = 7
  1110 = Overflow

(e) M =  7 = 0111
    S = -7 = 1001
   -S =      0111
```

```
  1010 = -6
 +1100 = -4
 10110 = Overflow

(f) M = -6 = 1010
    S =  4 = 0100
   -S =      1100
```



Geometric Depiction of Twos Complement Integers



OF = overflow bit
SW = Switch (select addition or subtraction)

Block Diagram of Hardware for Addition and Subtraction

Množenje nepredznačenih binarnih celih števil

Multiplication of two unsigned binary integers

4-bitna cela števila, dajo 8-bitni rezultat

4-Bit Integers Yields an 8-Bit Result

$$
\begin{array}{r}
1011 \\
\times\, 1101 \\
\hline
1011 \\
0000 \\
1011 \\
1011 \\
\hline
10001111
\end{array}
$$

množenec (11)
množitelj (13)

multiplicand (11)
multiplier (13)

delni zmnožki

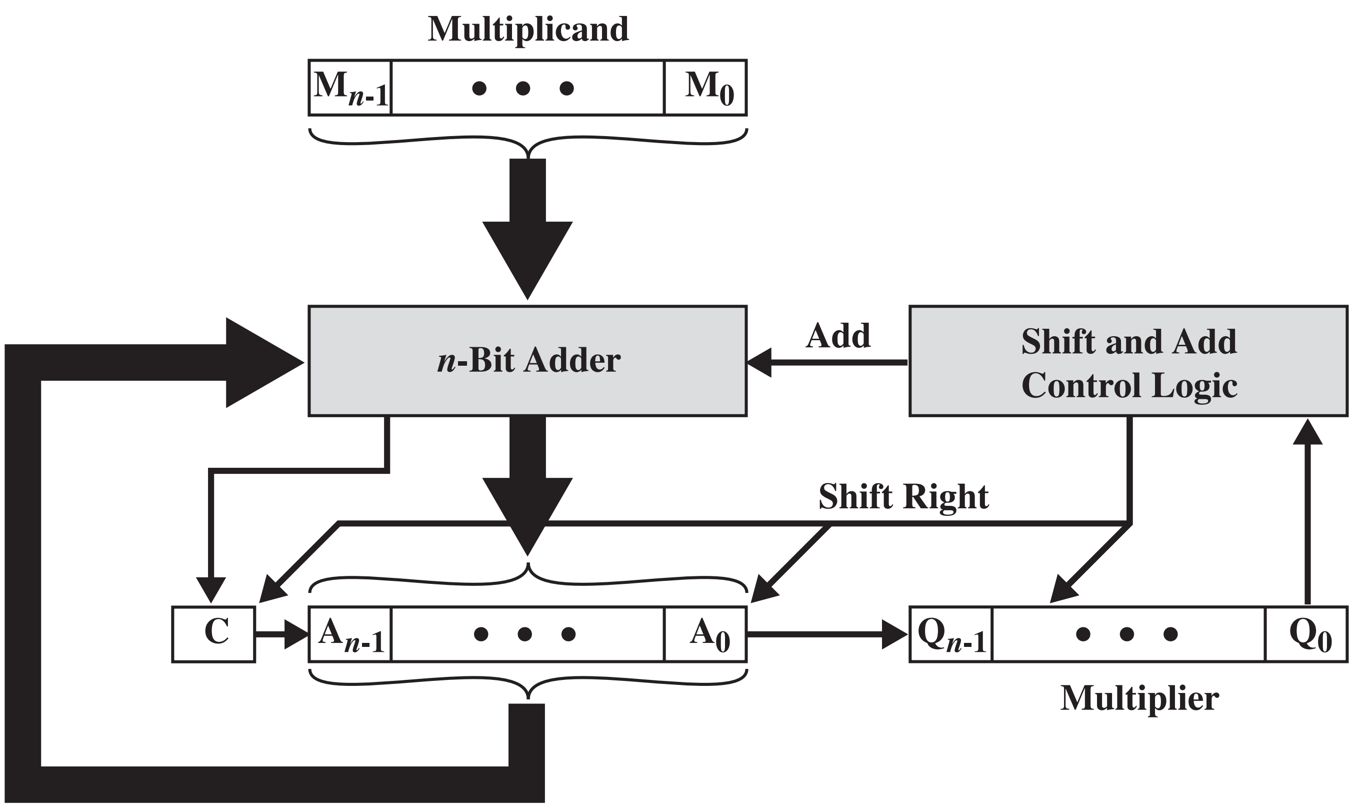
partial products

zmnožek (143)

product (143)

Strojna izvedba nepredznačenega binarnega množenja- blokovni diagram

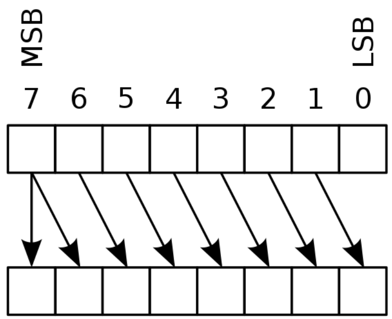
Hardware Implementation of Unsigned Binary Multiplication – Block Diagram



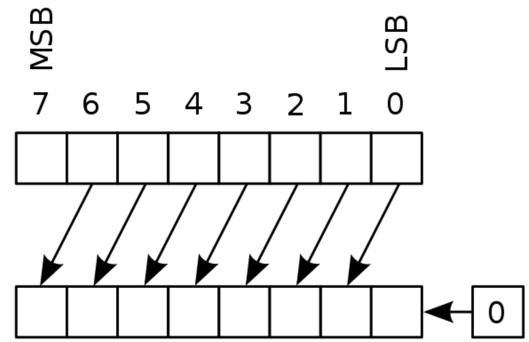
Logični in aritmetičn premik / Logical and aritmetic shift

Signed binary numbers

Aritmetic right



Aritmetic left

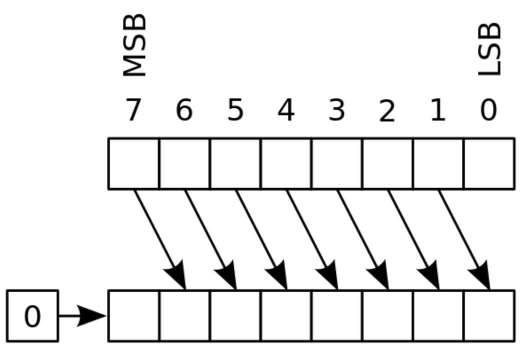


Division

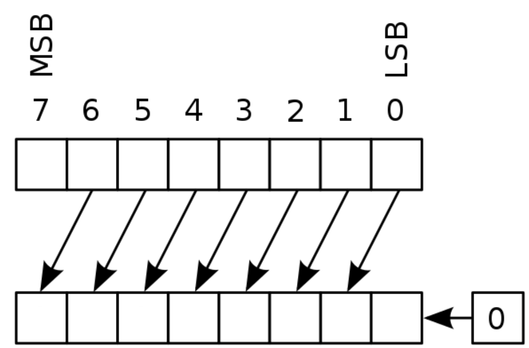
Multiplication

Unsigned binary numbers

Logical right



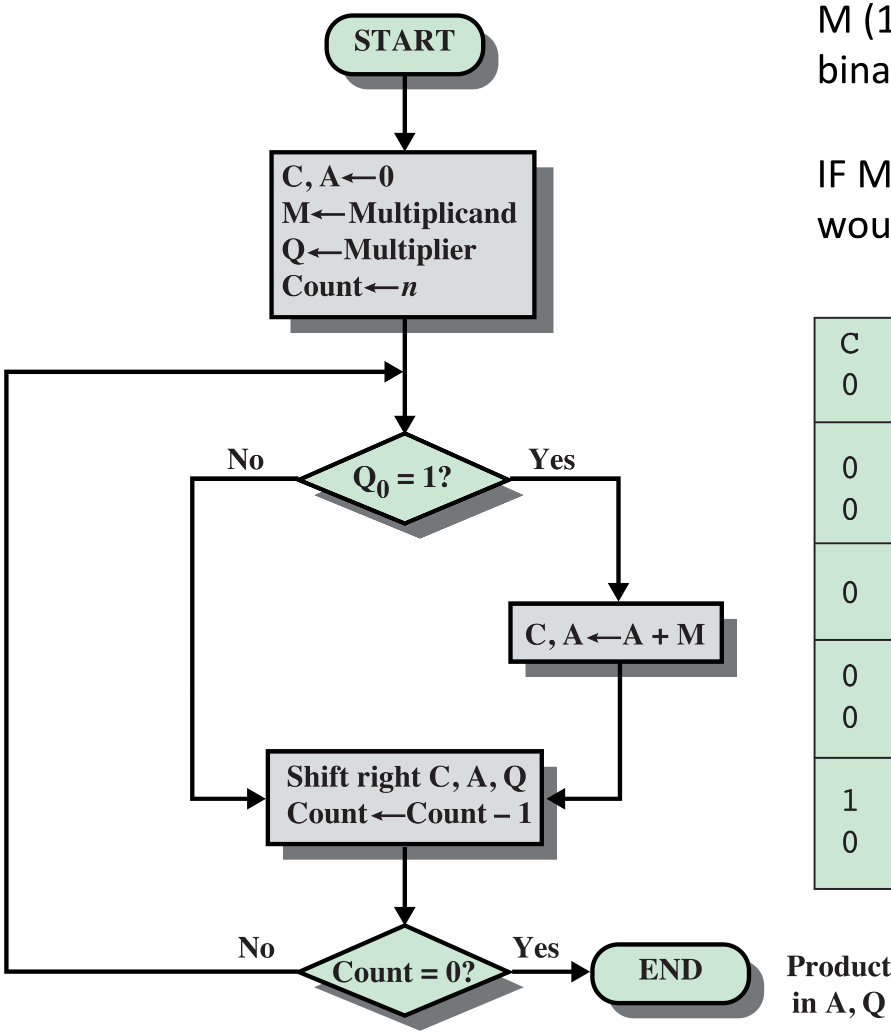
Logical left



Poskusimo rešiti / Try to solve it

M (1011 or 11) and Q (1101 or 13) are unsigned binary numbers. Result AQ is 1000111 (143).

IF M (-5) and Q (-3) would be T-C, the result would be -113 which is incorrect.



C	A	Q	M		
0	0000	1101	1011	Initial Values	
0	1011	1101	1011	Add	First Cycle
0	0101	1110	1011	Shift	
0	0010	1111	1011	Shift	Second Cycle
0	1101	1111	1011	Add	Third Cycle
0	0110	1111	1011	Shift	
1	0001	1111	1011	Add	Fourth Cycle
0	1000	1111	1011	Shift	

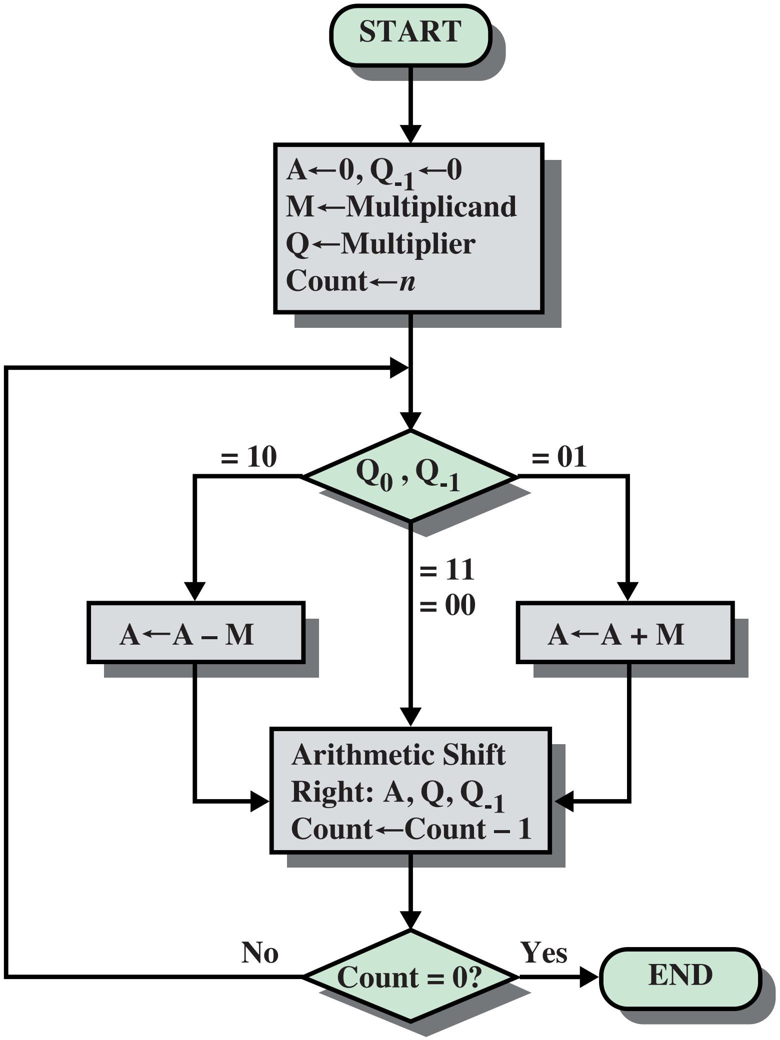
C bit used for overflow

Množenje negativnih števil / Multiplying Negative Numbers

- Ne dela!
- Rešitev 1
  - Pretvori v pozitivno število, če je potrebno
  - Množi kot prej
  - Če sta predznaka različna, negiraj rezultat
- Rešitev 2
  - Boothov algoritem

- Does not work!
- Solution 1
  - Convert to a positive number if needed
  - Multiply as before
  - If the signs are different, negate the result
- Solution 2
  - Booth's algorithm

Boothov algoritem / Booth's alghoritm



A	Q	Q_{-1}	M		
0000	0011	0	0111	Initial Values	
1001	0011	0	0111	A ← A − M	First Cycle
1100	1001	1	0111	Shift	
1110	0100	1	0111	Shift	Second Cycle
0101	0100	1	0111	A ← A + M	Third Cycle
0010	1010	0	0111	Shift	
0001	0101	0	0111	Shift	Fourth Cycle

7 * 3

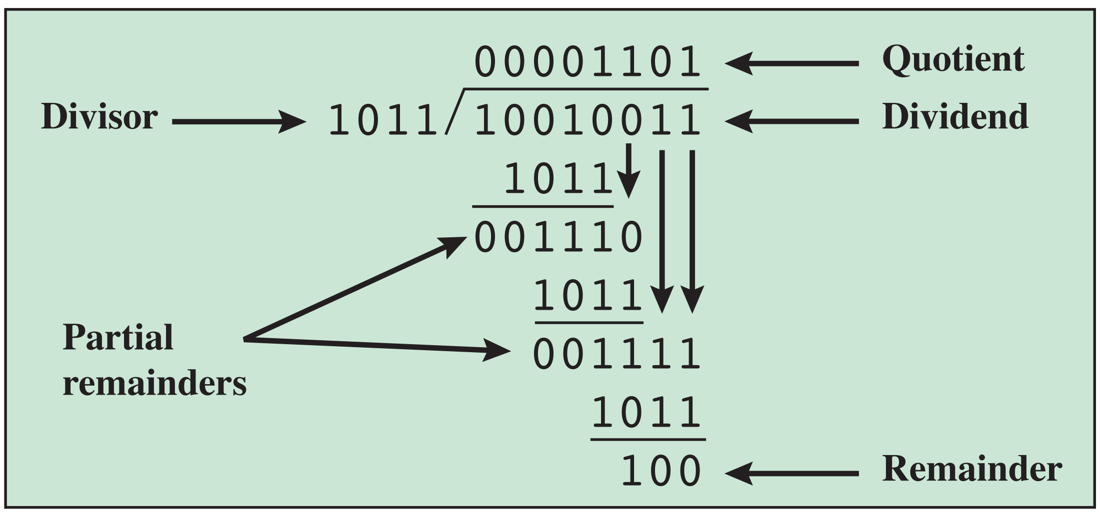
Poskusimo rešiti / Try to solve it

- Bolj kompleksno kot množenje
- Negativna števila so problematična!

- More complex than multiplication
- Negative numbers are problematic!

deljitelj

delni ostanki



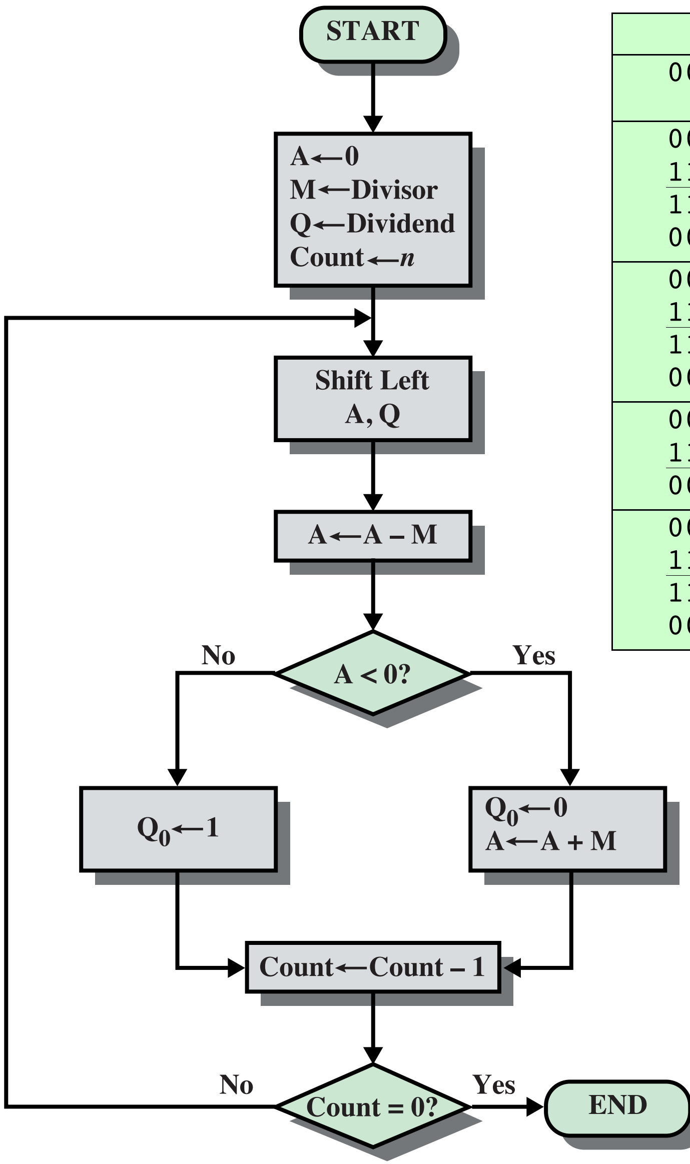
količnik

deljenec

ostanek

Division of unsigned binary numbers

Poskusimo rešiti / Try to solve it



Unsigned binary

A	Q	
0000	0111	Initial value
0000	1110	Shift
1101		Use twos complement of 0011 for subtraction
1101		Subtract
0000	1110	Restore, set $Q_0 = 0$
0001	1100	Shift
1101		
1110		Subtract
0001	1100	Restore, set $Q_0 = 0$
0011	1000	Shift
1101		
0000	1001	Subtract, set $Q_0 = 1$
0001	0010	Shift
1101		
1110		Subtract
0001	0010	Restore, set $Q_0 = 0$

Quotient in Q
Remainder in A

```
          Q        M
7 / 3 =  0111  /  0011
```

For twos complement division convert the operands into unsigned values and, at the end, to account for the signs Q and R where needed.

Predstavitev s plavajočo vejico / Floating-point representation

- Principi
- IEEE standard

- Principles
- IEEE standard

Realna števila / Real Numbers

- Ulomki (racionalna) + iracionalna števila
- Lahko predstavimo v čisti dvojiški obliki

- Fractions (rational) + irrational numbers
- Can be presented in pure binary form

$$1001{,}1010 = 2^3 + 2^0 + 2^{-1} + 2^{-3} = 9{,}625$$

- Kje je dvojiška vejica?
- Premična?
  - Kako pokazati kje se nahaja?

- Where's the radix point?
- Movable?
  - How to show where it is located?

- Fiksna?
- Z zapisom s fiksno vejico je mogoče predstavljati obseg pozitivnih in negativnih celih števil, osredotočenih okoli 0
- Omejitve:
  - Zelo velikega števila ni mogoče zastopati niti zelo majhnih decimalnih vrednosti (ulomek)
  - Decimalni del količnika pri deljenju dveh velikih števil se lahko izgubi

- Fixed?
- With a fixed-point notation it is possible to represent a range of positive and negative integers centered on or near 0
- Limitations:
  - Very large numbers cannot be represented nor can very small fractions
  - The fractional part of the quotient in a division of two large numbers could be lost

Števila s plavajočo vejico / Floating-point numbers

$$\pm S * B^{\pm E}$$

$$\pm 1.bbbbb\ldots b * 2^{\pm E} \quad b=\{0,1\}$$

- Predznak
- Mantisa ali Signifikand
- Exponent

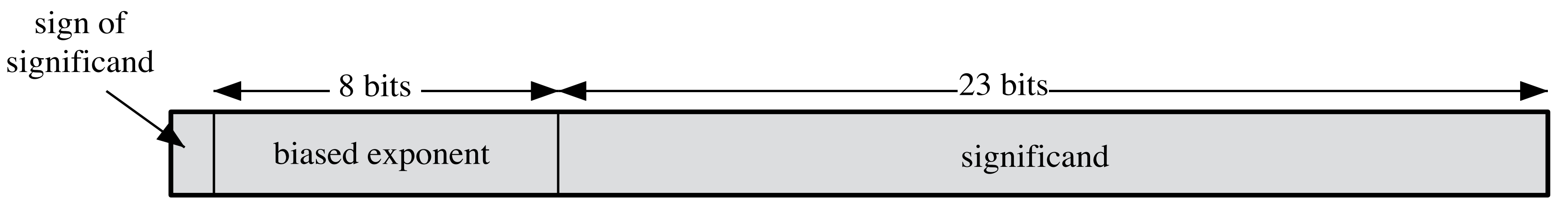
- Sign: plus or minus
- Significand S
- Exponent E

Predznak / Sign of significand	Pristranski exponent / Biased exponent	Mantisa / Significand

- Napačno poimenovanje
  - Vejica je v resnici fiksirana med bit predznaka in mantiso
- Eksponent določa položaj vejice

- Wrong naming
  - The radix point is fixed between the sign bit and the significand
- The exponent determines the location of the radix point

Primeri 32 bitnih števil s plavajočo vejico / Examples of 32-bit floating-point number



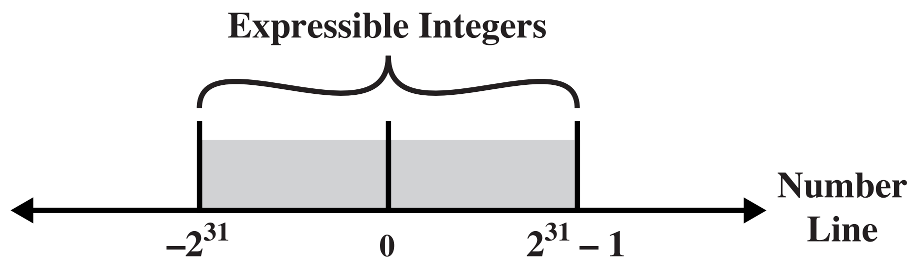
(a) Format

$$1.1010001 \times 2^{10100} = 0\ 10010011\ 10100010000000000000000 = 1.6328125 \times 2^{20}$$
$$-1.1010001 \times 2^{10100} = 1\ 10010011\ 10100010000000000000000 = -1.6328125 \times 2^{20}$$
$$1.1010001 \times 2^{-10100} = 0\ 01101011\ 10100010000000000000000 = 1.6328125 \times 2^{-20}$$
$$-1.1010001 \times 2^{-10100} = 1\ 01101011\ 10100010000000000000000 = -1.6328125 \times 2^{-20}$$

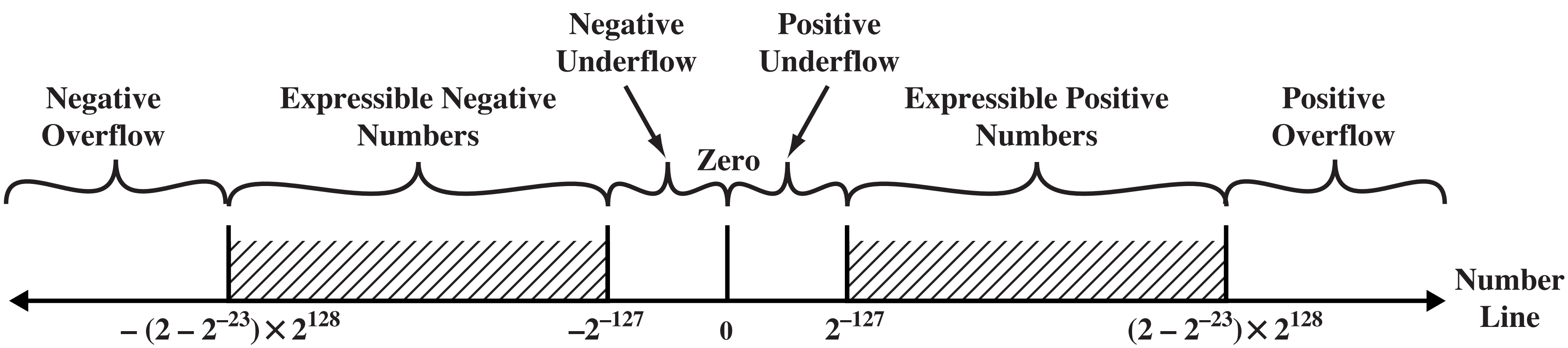
Deli predstavitve PV / Parts of the FP representation

- Eksponent je v pristranski obliki. Na primer 32-bitni PV
  - 8 bitno eksponentno polje
  - Možne vrednosti 0-255
  - Odštej 127 za pravilno vrednost
  - Obseg-127 do +128
- Significand
  - FP števila so običajno normalizirana
  - Tj. eksponent je poravnan tako, da je vodilni bit (MSB) mantise enak 1
  - Ker je vedno 1, ni potrebe, da se ga hrani

- Exponent is in the biased form. For example in 32-bit FP:
  - 8 bits
  - Posible values 0-255
  - Substract 127 to get the real value
  - Range-127 do +128
- Significand
  - FP numbers are usually normalised
  - Ie. the exponent is aligned such that the leading bit (MSB) is 1
  - Since it is always 1, there is no need to present it



(a) Twos Complement Integers



(b) Floating-Point Numbers

Expressible Numbers in Typical 32-Bit Formats

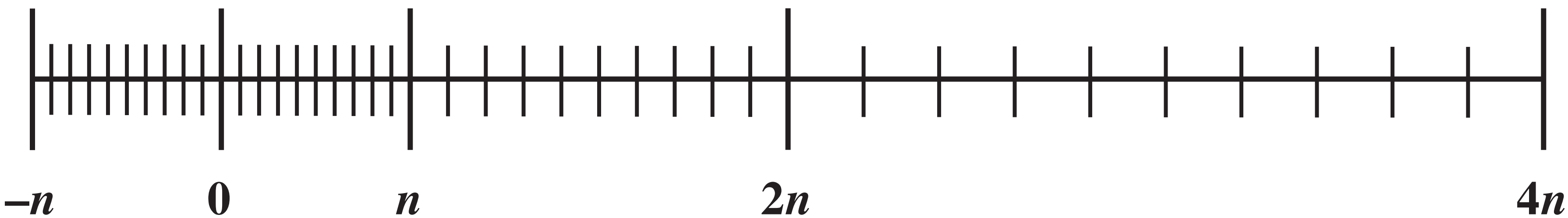


Figure 10.20 Density of Floating-Point Numbers

IEEE Standard 754

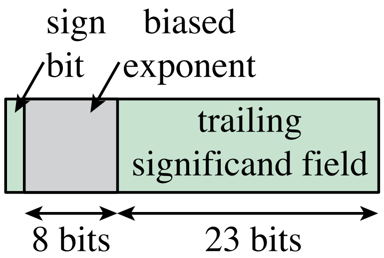
- Opredeljuje najpomembnejši prikaz s plavajočo vejico
- Standard je bil razvit za lažjo prenosljivost programov z enega procesorja na drugega in za spodbujanje razvoja prefinjenih numerično usmerjenih programov
- Standard je bil široko sprejet in se uporablja na skoraj vseh sodobnih procesorjih in aritmetičnih koprocesorjih
- IEEE 754-2008 zajema binarne in decimalne predstavitve s plavajočo vejico

- Most important floating-point representation is defined
- Standard was developed to facilitate the portability of programs from one processor to another and to encourage the development of sophisticated, numerically oriented programs
- Standard has been widely adopted and is used on virtually all contemporary processors and arithmetic coprocessors
- IEEE 754-2008 covers both binary and decimal floating-point representations

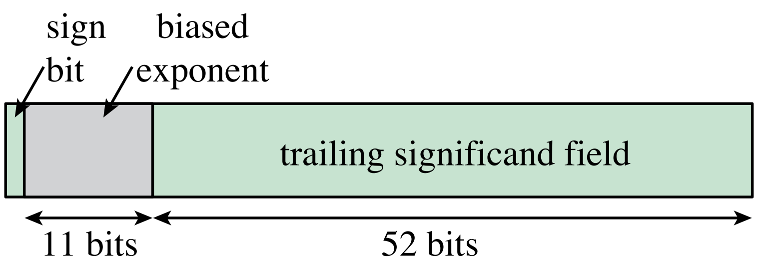
IEEE 754-2008

- Določi naslednje različne vrste formatov s plavajočo vejico:
- Aritmetična oblika
  - Oblika podpira vse obvezne operacije, ki jih določa standard. Oblika se lahko uporabi za prikaz operandov s plavajočo vejico ali rezultatov za operacije, opisane v standardu.
- Osnovna oblika
  - Ta oblika zajema pet predstavitev s plavajočo vejico, tri binarne in dve decimalki, katerih kodiranje določa standard in jih je mogoče uporabiti za aritmetiko. Vsaj eden od osnovnih formatov je izveden v kateri koli skladni izvedbi.
- Oblika izmenjave
  - Popolnoma določeno binarno kodiranje s fiksno dolžino, ki omogoča izmenjavo podatkov med različnimi platformami in se lahko uporablja za shranjevanje.

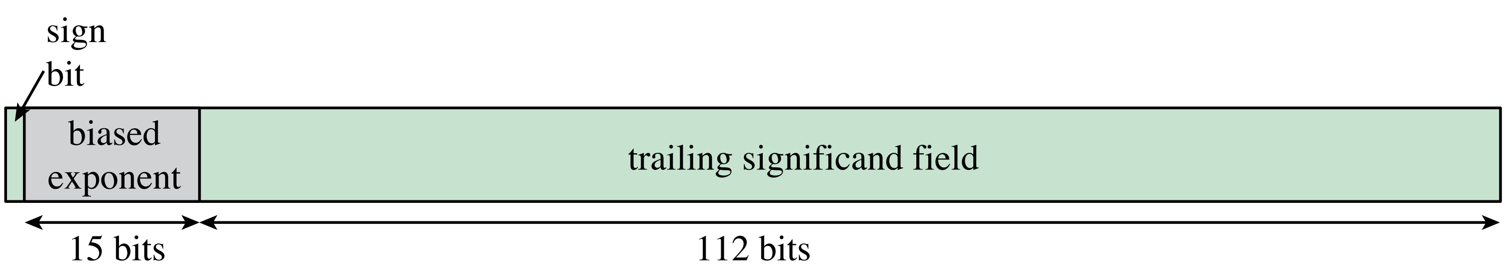
- Defines the following different types of floating-point formats:
- Arithmetic format
  - All the mandatory operations defined by the standard are supported by the format. The format may be used to represent floating-point operands or results for the operations described in the standard.
- Basic format
  - This format covers five floating-point representations, three binary and two decimal, whose encodings are specified by the standard, and which can be used for arithmetic. At least one of the basic formats is implemented in any conforming implementation.
- Interchange format
  - A fully specified, fixed-length binary encoding that allows data interchange between different platforms and that can be used for storage.



(a) binary32 format



(b) binary64 format



(c) binary128 format

Table 10.3 IEEE 754 Format Parameters

Parameter	Format		
	binary32	**binary64**	**binary128**
Storage width (bits)	32	64	128
Exponent width (bits)	8	11	15
Exponent bias	127	1023	16383
Maximum exponent	127	1023	16383
Minimum exponent	–126	–1022	–16382
Approx normal number range (base 10)	$10_{-38}, 10_{+38}$	$10_{-308}, 10_{+308}$	$10_{-4932}, 10_{+4932}$
Trailing significand width (bits)*	23	52	112
Number of exponents	254	2046	32766
Number of fractions	2_{23}	2_{52}	2_{112}
Number of values	$1.98 \times 2_{31}$	$1.99 \times 2_{63}$	$1.99 \times 2_{128}$
Smallest positive normal number	2_{-126}	2_{-1022}	2_{-16362}
Largest positive normal number	$2_{128} - 2_{104}$	$2_{1024} - 2_{971}$	$2_{16384} - 2_{16271}$
Smallest subnormal magnitude	2_{-149}	2_{-1074}	2_{-16494}

* not including implied bit and not including sign bit

Dodatne oblike / Additional formats

Razširjeni natančni formati

- Omogočajo dodatne bite v eksponentu (razširjeni obseg) in v mantisi (razširjena natančnost)
- Zmanjšujejo pretirano napako končnega rezultata, ki lahko nastane z zaokroževanjem
- Zmanjšuje možnost vmesne prekoračitve, ki prekine izračun, katerega končni rezultat bi bil predstavljiv v osnovnem format
- Omogoča nekatere prednosti večjega osnovnega formata, ne da bi prišlo do časovne kazni, ki je običajno povezana z večjo natančnostjo

Razširljiva oblika natančnosti

- Natančnost in domet sta določena od in pod nadzorom uporabnika
- Lahko se uporablja za vmesne izračune, vendar standardni ne omejujejo ali oblikujejo ali dolžine

Extended precision formats

- Provide additional bits in the exponent (extended range) and in the significand (extended precision)
- Lessens the chance of a final result that has been contaminated by excessive roundoff error
- Lessens the chance of an intermediate overflow aborting a computation whose final result would have been representable in a basic format
- Affords some of the benefits of a larger basic format without incurring the time penalty usually associated with higher precision

Extendable precision form

- Precision and range are defined under user control
- May be used for intermediate calculations but the standard places no constraint or format or length

Table 10.4
IEEE Formats

Format	Format Type		
	Arithmetic Format	Basic Format	Interchange Format
binary16			X
binary32	X	X	X
binary64	X	X	X
binary128	X	X	X
binary{k} ($k = n \times 32$ for $n > 4$)	X		X
decimal64	X	X	X
decimal128	X	X	X
decimal{k} ($k = n \times 32$ for $n > 4$)	X		X
extended precision	X		
extendable precision	X		

Table 10.5
Interpretation of IEEE 754 Floating-Point Numbers (page 1 of 3)

	Sign	**Biased exponent**	**Fraction**	**Value**
positive zero	0	0	0	0
negative zero	1	0	0	–0
plus infinity	0	all 1s	0	∞
Minus infinity	1	all 1s	0	$-\infty$
quiet NaN	0 or 1	all 1s	$\neq 0$; first bit = 1	qNaN
signaling NaN	0 or 1	all 1s	$\neq 0$; first bit = 0	sNaN
positive normal nonzero	0	$0 < e < 255$	f	$2_{e-127}(1.f)$
negative normal nonzero	1	$0 < e < 255$	f	$-2_{e-127}(1.f)$
positive subnormal	0	0	$f \neq 0$	$2_{e-126}(0.f)$
negative subnormal	1	0	$f \neq 0$	$-2_{e-126}(0.f)$

(a) binary32 format

Table 10.5
Interpretation of IEEE 754 Floating-Point Numbers (page 2 of 3)

	Sign	**Biased exponent**	**Fraction**	**Value**
positive zero	0	0	0	0
negative zero	1	0	0	–0
plus infinity	0	all 1s	0	∞
Minus infinity	1	all 1s	0	–∞
quiet NaN	0 or 1	all 1s	≠ 0; first bit = 1	qNaN
signaling NaN	0 or 1	all 1s	≠ 0; first bit = 0	sNaN
positive normal nonzero	0	$0 < e < 2047$	f	$2_{e-1023}(1.f)$
negative normal nonzero	1	$0 < e < 2047$	f	$-2_{e-1023}(1.f)$
positive subnormal	0	0	$f \neq 0$	$2_{e-1022}(0.f)$
negative subnormal	1	0	$f \neq 0$	$-2_{e-1022}(0.f)$

(a) binary64 format

Table 10.5
Interpretation of IEEE 754 Floating-Point Numbers (page 3 of 3)

	Sign	**Biased exponent**	**Fraction**	**Value**
positive zero	0	0	0	0
negative zero	1	0	0	–0
plus infinity	0	all 1s	0	∞
minus infinity	1	all 1s	0	$-\infty$
quiet NaN	0 or 1	all 1s	$\neq 0$; first bit $= 1$	qNaN
signaling NaN	0 or 1	all 1s	$\neq 0$; first bit $= 0$	sNaN
positive normal nonzero	0	all 1s	f	$2_{e-16383}(1.f)$
negative normal nonzero	1	all 1s	f	$-2_{e-16383}(1.f)$
positive subnormal	0	0	$f \neq 0$	$2_{e-16383}(0.f)$
negative subnormal	1	0	$f \neq 0$	$-2_{e-16383}(0.f)$

(a) binary128 format

FP Aritmetične operacije / FP Arithmetic Operations

Floating Point Numbers	Arithmetic Operations
$X = X_S \times B^{X_E}$ $Y = Y_S \times B^{Y_E}$	$\left.\begin{array}{l} X + Y = \left(X_s \times B^{X_E - Y_E} + Y_s\right) \times B^{Y_E} \\ X - Y = \left(X_s \times B^{X_E - Y_E} - Y_s\right) \times B^{Y_E} \end{array}\right\} X_E \leq Y_E$ $X \times Y = \left(X_s \times Y_s\right) \times B^{X_E + Y_E}$ $\frac{X}{Y} = \left(\frac{X_s}{Y_s}\right) \times B^{X_E - Y_E}$

Examples:

$X = 0.3 \times 10^2 = 30$
$Y = 0.2 \times 10^3 = 200$

$X + Y = (0.3 \times 10_{2-3} + 0.2) \times 10_3 = 0.23 \times 10_3 = 230$
$X - Y = (0.3 \times 10_{2-3} - 0.2) \times 10_3 = (-0.17) \times 10_3 = -170$
$X \times Y = (0.3 \times 0.2) \times 10_{2+3} = 0.06 \times 10_5 = 6000$
$X \div Y = (0.3 \div 0.2) \times 10_{2-3} = 1.5 \times 10_{-1} = 0.15$

Izjemni rezultati / Exceptional results

Operacija s plavajočo vejico lahko povzroči tudi enega od teh pogojev:

- Prekoračitev eksponenta: + ∞ ali - ∞.
- Podkoračitev eksponentov: npr. -200 <-127
- Prekoračitev significanda
- Podkoračitev significanda

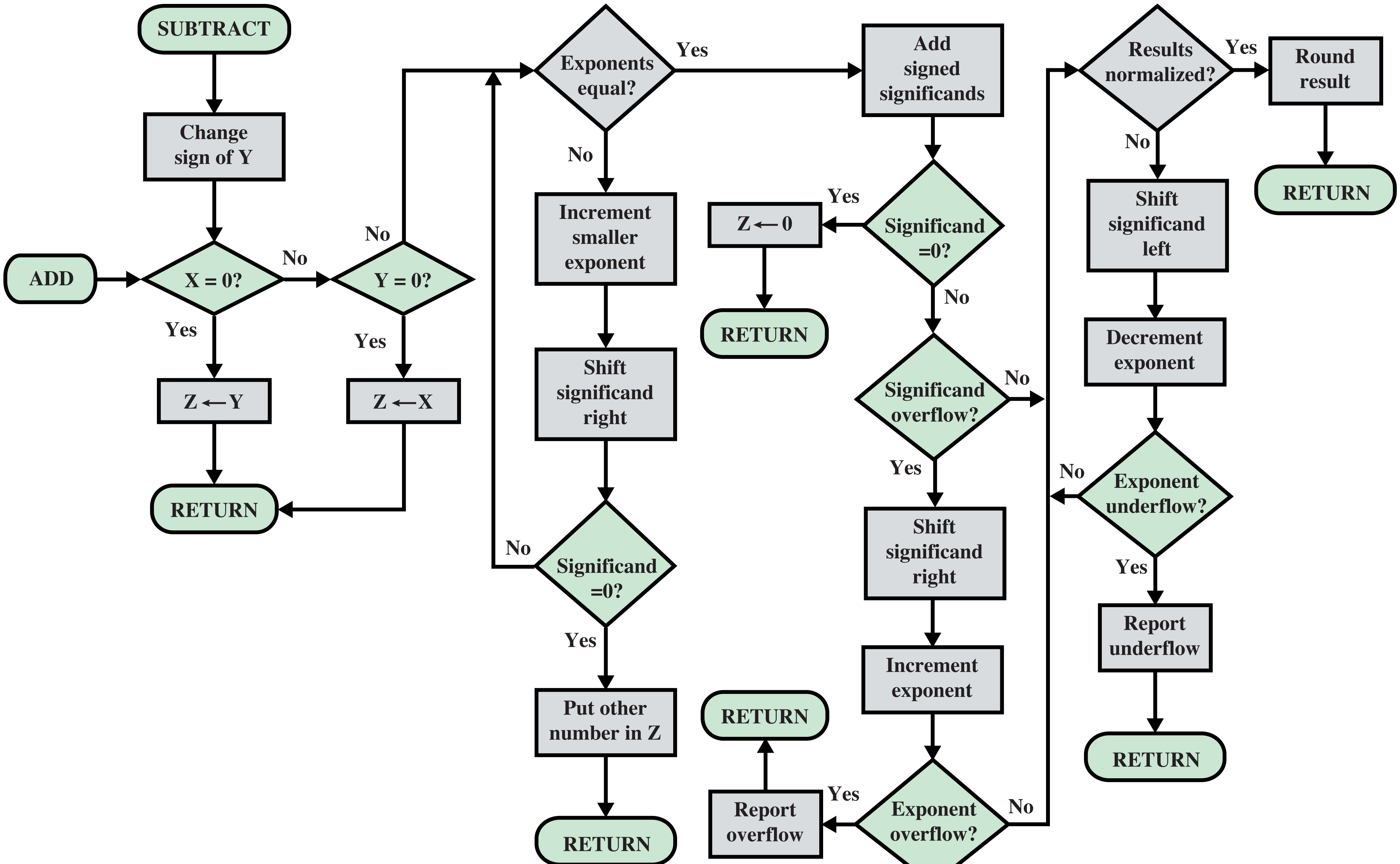
A floating-point operation may produce also one of these conditions:

- Exponent overflow: + ∞ or - ∞.
- Exponent underflow: e.g. -200 < -127
- Significand underflow
- Significand overflow

FP aritmetika: +/- / FP arithmetic +/-

- Preveri se za ničlo
- Poravna se mantisi s popravljanjem eksponenta
- Mantisi se sešteje ali odšteje
- Normalizira se rezultat

- Check for zero
- Aligns the significand with the exponent correction
- Significands are added up or subtracted
- The result is normalized

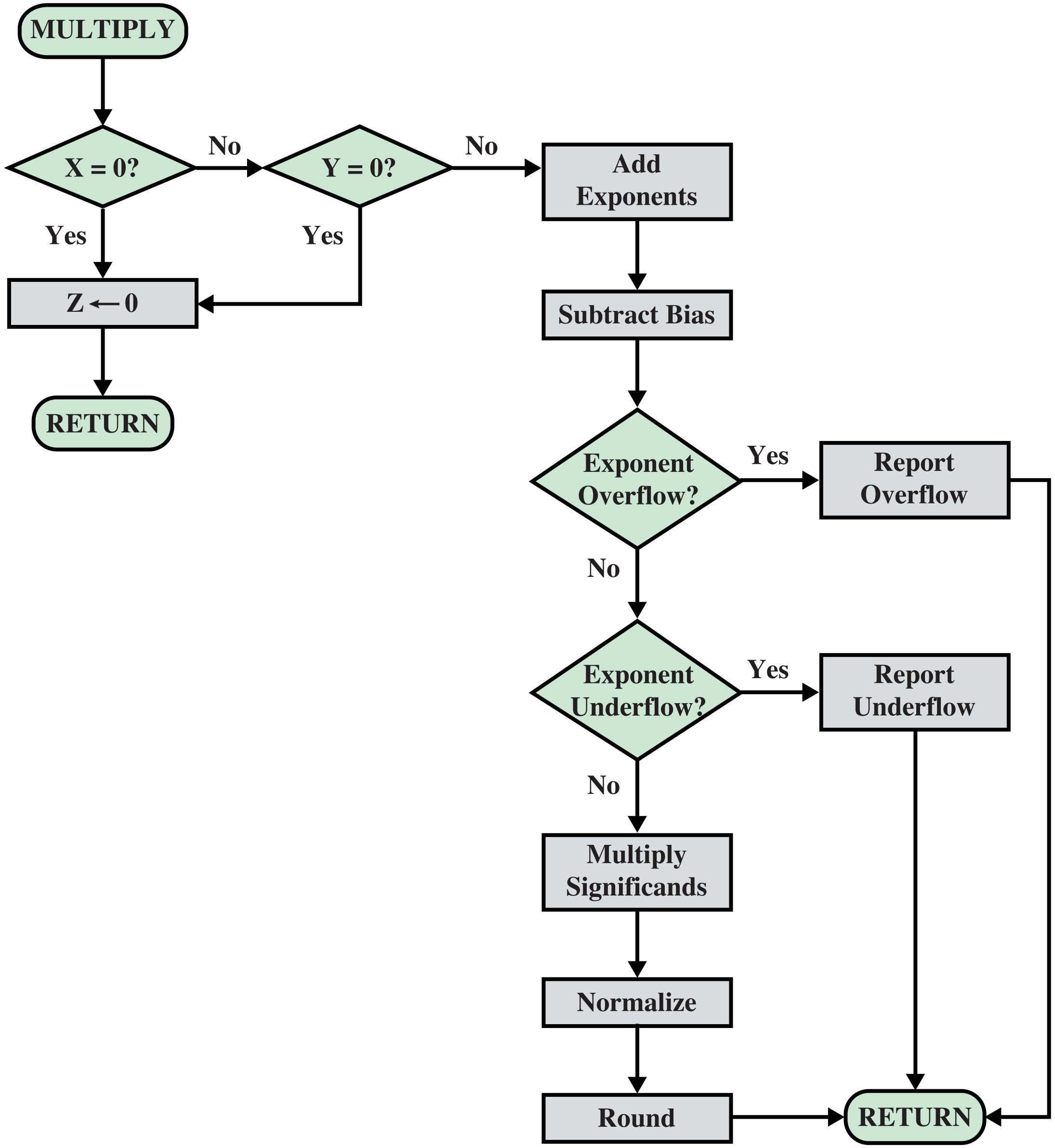


Floating-Point Addition and Subtraction (Z ← X ± Y)

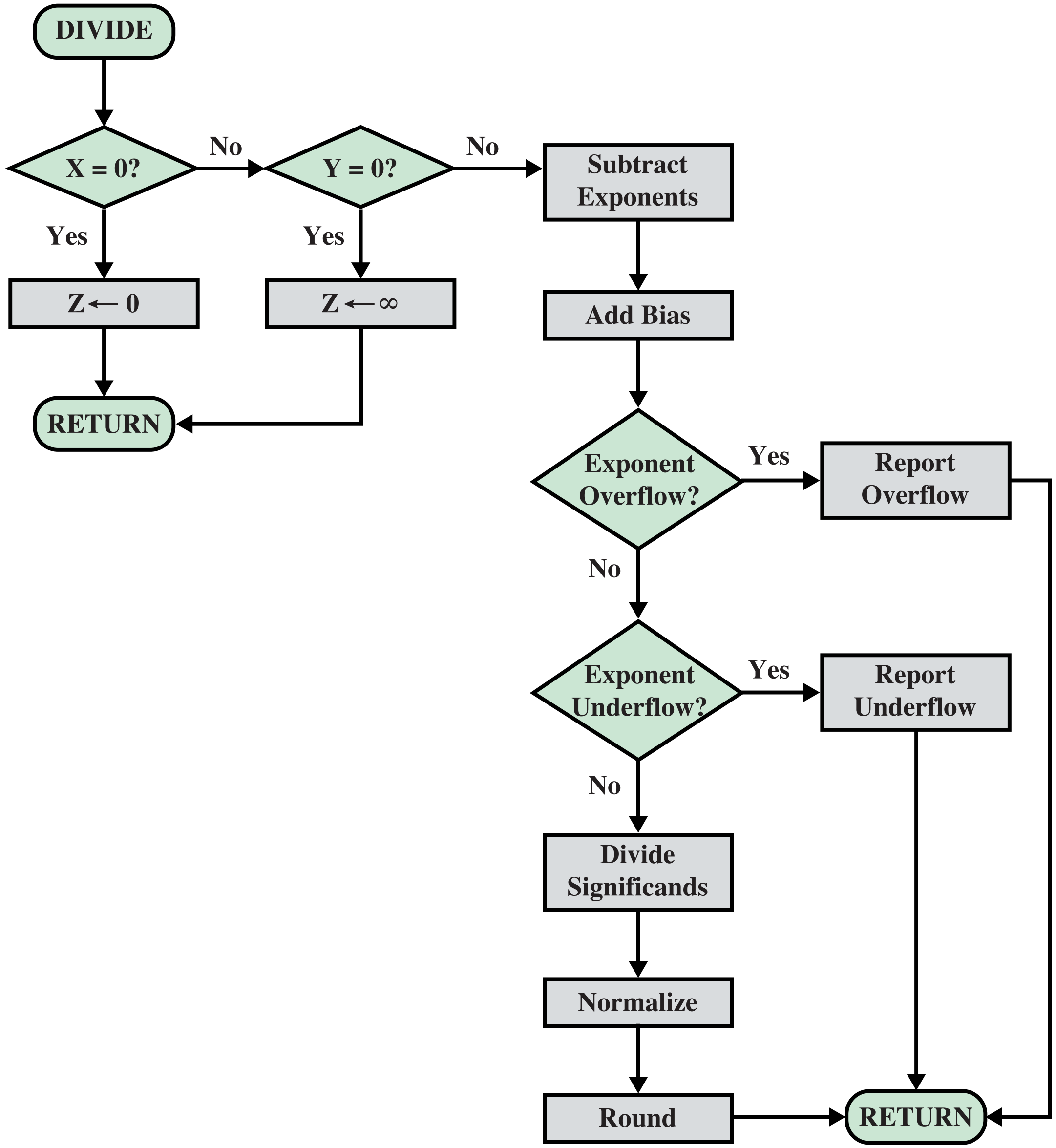
FP aritmetika: × / ÷ / FP arithmetic × / ÷

- Preveri se za ničlo
- Eksponenta se seštejeta ali odštejeta
- Mantisi se zmnožita ali delita
- Število se normalizira
- Nato zaokroži
- Vmesni rezultat bi moral biti shranjen v pomnilniku dvojne dolžine

- Check for zero
- The exponent is summed or subtracted
- Significands are multiplied or divided
- The number is normalized
- Then rounded up
- The intermediate result should be stored in double-length memory



Floating-Point Multiplication (Z ← X × Y)



Floating-Point Division (Z← X/Y)

Uporaba varovalnih bitov / The Use of Guard Bits

$$
\begin{array}{rcl}
x & = & \mathtt{1.000.....00} \times 2^{1} \\
\underline{-y} & = & \underline{\mathtt{0.111.....11}} \times 2^{1} \\
z & = & \mathtt{0.000.....01} \times 2^{1} \\
 & = & \mathtt{1.000.....00} \times 2^{-22}
\end{array}
$$

(a) Binary example, without guard bits

$$
\begin{array}{rcl}
x & = & \mathtt{.100000} \times 16^{1} \\
\underline{-y} & = & \underline{\mathtt{.0FFFFF}} \times 16^{1} \\
z & = & \mathtt{.000001} \times 16^{1} \\
 & = & \mathtt{.100000} \times 16^{-4}
\end{array}
$$

(c) Hexadecimal example, without guard bits

$$
\begin{array}{rcl}
x & = & \mathtt{1.000.....00\ 0000} \times 2^{1} \\
\underline{-y} & = & \underline{\mathtt{0.111.....11\ 1000}} \times 2^{1} \\
z & = & \mathtt{0.000.....00\ 1000} \times 2^{1} \\
 & = & \mathtt{1.000.....00\ 0000} \times 2^{-23}
\end{array}
$$

(b) Binary example, with guard bits

$$
\begin{array}{rcl}
x & = & \mathtt{.100000\ 00} \times 16^{1} \\
\underline{-y} & = & \underline{\mathtt{.0FFFFF\ F0}} \times 16^{1} \\
z & = & \mathtt{.000000\ 10} \times 16^{1} \\
 & = & \mathtt{.100000\ 00} \times 16^{-5}
\end{array}
$$

(d) Hexadecimal example, with guard bits

Natančnost / Precision Considerations

Standardni pristopi IEEE:

- Zaokrožitev do najbližje:
  - Rezultat je zaokrožen na najbližjo predstavljivo številko.
- Zaokrožitev proti + ∞:
  - Rezultat je zaokrožen proti plusu.
- Zaokrožitev proti-∞:
  - Rezultat je zaokrožen navzdol proti negativni neskončnosti.
- Zaokrožitev proti 0:
  - Rezultat se zaokroži na nič.

IEEE standard approaches:

- Round to nearest:
  - The result is rounded to the nearest representable number.
- Round toward +∞ :
  - The result is rounded up toward plus infinity.
- Round toward -∞:
  - The result is rounded down toward negative infinity.
- Round toward 0:
  - The result is rounded toward zero.

Intervalna aritmetika / Interval Arithmetic

Negativna neskončnost in *zaokrožitev na plus* sta uporabna pri izvajanju intervalne aritmetike

- Zagotavlja učinkovito metodo za spremljanje in nadzor napak pri izračunih s plavajočo vejico, tako da ustvari dve vrednosti za vsak rezultat
- Obe vrednosti ustrezata spodnji in zgornji končni točki intervala, ki vsebuje resnični rezultat
- Širina intervala označuje natančnost rezultata
- Če končne točke niso predstavljive, se intervalne končne točke zaokrožijo navzdol oziroma navzgor
- Če je razpon med zgornjim in spodnjim robom dovolj ozek, smo dobili dovolj natančen rezultat

Minus infinity and *rounding to plus* are useful in implementing interval arithmetic

- Provides an efficient method for monitoring and controlling errors in floating-point computations by producing two values for each result
- The two values correspond to the lower and upper endpoints of an interval that contains the true result
- The width of the interval indicates the accuracy of the result
- If the endpoints are not representable then the interval endpoints are rounded down and up respectively
- If the range between the upper and lower bounds is sufficiently narrow then a sufficiently accurate result has been obtained

Odrezovanje / Truncation

- *Zaokrožitev proti ničli*
- Dodatni biti se ne upoštevajo
- Najenostavnejša tehnika
- Stalna pristranskost proti ničli v operaciji
  - Resna pristranskost, ker vpliva na vsako operacijo, za katero obstajajo neničelni dodatni biti

- *Round toward zero*
- Extra bits are ignored
- Simplest technique
- A consistent bias toward zero in the operation
  - Serious bias because it affects every operation for which there are nonzero extra bits

IEEE standard za binarno aritmetiko s PV – neskončnost / IEEE Standard for Binary FP Arithmetic - Infinity

- Obravnava se kot omejevalni primer prave aritmetike, pri čemer so vrednosti neskončnosti podane z naslednjo razlago:

- Is treated as the limiting case of real arithmetic, with the infinity values given the following interpretation:

$$-\infty < (\text{every finite number}) < +\infty$$

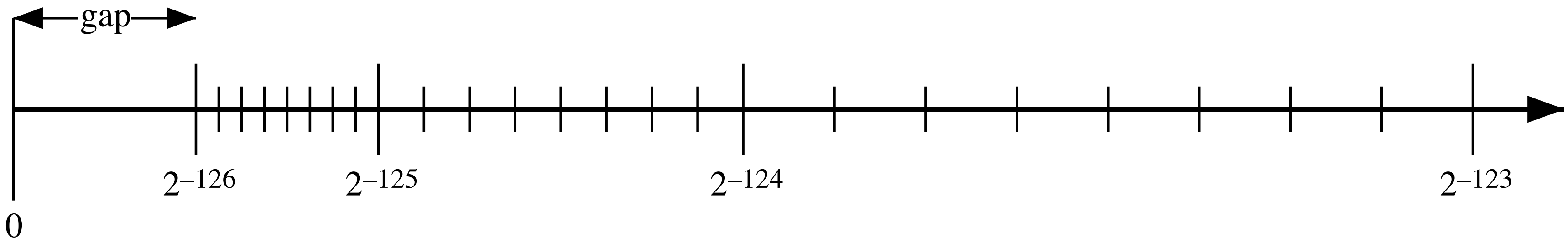
For example:

```
5 + (+ ∞ ) = + ∞
5 - (+ ∞ ) = - ∞
5 + (- ∞ ) = - ∞
5 - (- ∞ ) = + ∞
5 * (+ ∞ ) = + ∞
```

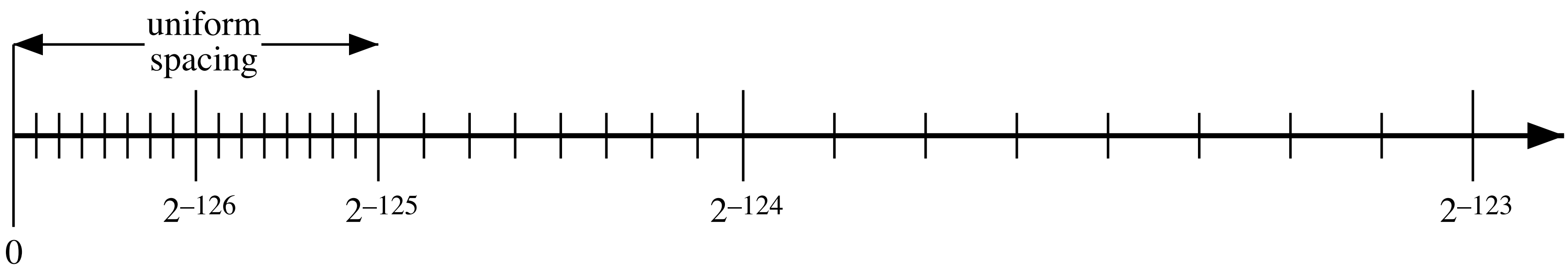
```
5÷ (+ ∞ ) = +0
(+ ∞ ) + (+ ∞ ) = + ∞
(- ∞ ) + (- ∞)  = - ∞
(- ∞ ) - (+ ∞ ) = - ∞
(+ ∞ ) - (- ∞ ) = + ∞
```

Table:
Operations that Produce a Quiet NaN

Operation	**Quiet NaN Produced by**
Any	Any operation on a signaling NaN
Add or subtract	Magnitude subtraction of infinities: $(+\infty) + (-\infty)$ $(-\infty) + (+\infty)$ $(+\infty) - (+\infty)$ $(-\infty) - (-\infty)$
Multiply	$0 \times \infty$
Division	$\frac{0}{0}$ or $\frac{\infty}{\infty}$
Remainder	x REM 0 or ∞ REM y
Square root	$\sqrt{x}$ where $x < 0$



(a) 32-bit format without subnormal numbers



(b) 32-bit format with subnormal numbers

The Effect of IEEE 754 Subnormal Numbers

Vrpašanja / Questions