Računalniška aritmetika
Computer Arithmetics

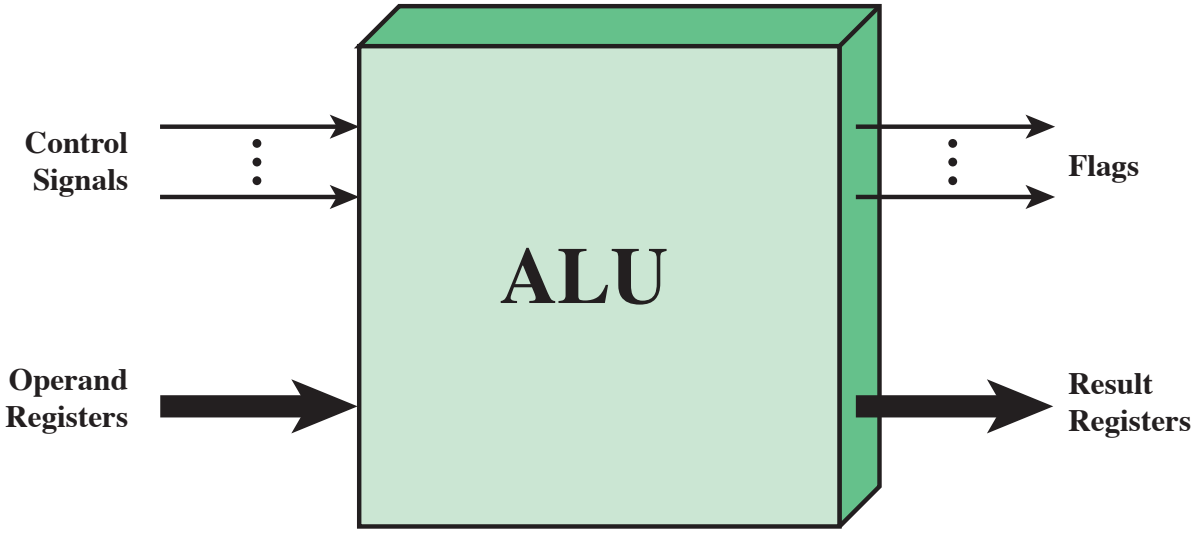
Aritmetično logična enota (ALE) / Arithmetic & Logic Unit (ALU)

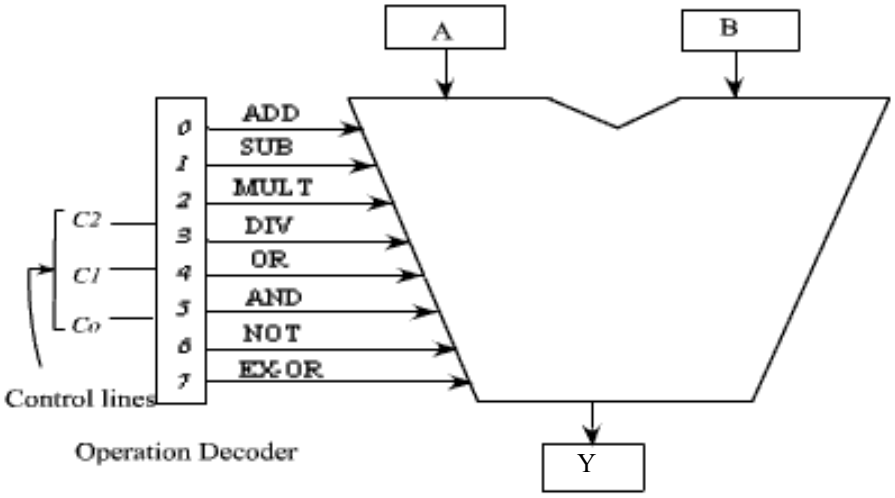
- Izvaja računske operacije na podatkih.
- Drugi elementi računalniškega sistema so namenjeni dostavi podatkov v ALE, ki jih ta obdela, nato pa dobi rezultate nazaj.
- Temelji na uporabi preprostih digitalnih logičnih naprav, ki lahko shranjujejo dvojiške števke in izvajajo preproste Booleanove logične operacije.
- Operira nad celimi števili.

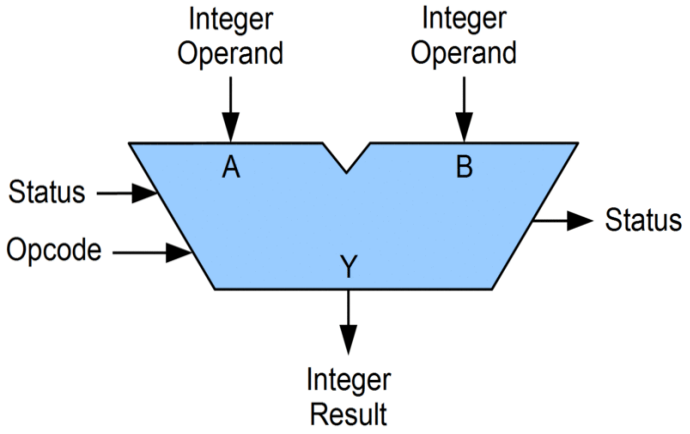
- Enota za delo s plavajočo vejico:
  - Dela z realnimi (plavajoča vejica) števili
  - Lahko obstaja kor ločena enota – npr. matematični koprocesor (486DX+)

- 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.

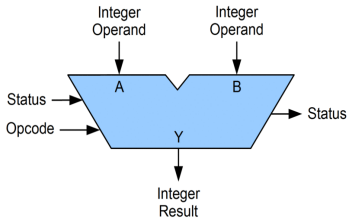
- Floating point unit (FPU)
  - Deals with real (floating point) numbers
  - It can exist as a separate unit – e.g. mathematical coprocessor (486DX +)


Control
Signals
ALU
Flags
Operand
Registers
Result
Registers


A
B
ADD
SUB
MULT
DIV
OR
AND
NOT
EX-OR
0
1
2
3
4
5
6
7
C2
C1
Co
Control lines
Operation Decoder
Y


Integer
Operand
Integer
Operand
A
B
Status
Opcode
Status
Y
Integer
Result

ALE operacije / ALU operations



- Aritmetika celih števil
  - Seštevanje: A se doda B, vsota se prikaže na Y in prenos na Satus.
  - Seštevanje s prenosom: A, B in prenos se seštejejo, vsota se prikaže pri Y in prenos na Satus.
  - Odštevanje: B se odšteje od A (ali obratno), razlika se prikaže na Y in prenos (izposoja ven) na Satus.
  - Odštevanje z izposojo: B se odšteje od A (ali obratno) z izposojo (carry-in), razlika pa se pojavi na Y in prenos (borrow out) na Satus.
  - Dvojiški komplement: A (ali B) se odšteje od nič in razlika se pojavi na Y.
  - Povečanje: A (ali B) se poveča za ena in dobljena vrednost se prikaže na Y.
  - Zmanjšanje: A (ali B) se zmanjša za ena in dobljena vrednost se prikaže na Y.
  - Prehod: vsi biti A (ali B) se prikažejo nespremenjeni na Y (uporablja se za določanje paritete A, ugotavljanje če je A nič ali negativen, ali za nalaganje A v register procesorja).
- Bitne logične operacije
  - AND: bitni A in B, rezultat se prikaže v Y.
  - ALI: bitni A ali B, rezultat se prikaže na Y.
  - Ekskluzivi OR: bitni A xor B, rezultat se pojavi na Y.
  - Komplement: vsi biti A (ali B) so invertirani in se prikažejo na Y.
- Operacije pomikanja
  - Aritmetični pomik: operand se obravnava kot celo število z dvema dopolnitvama, kar pomeni, da je najbolj uteženi bit ”predznak” in se ohrani.
  - Logični pomik: v operand se premakne logična ničla. To se uporablja za premikanje nepredznačenih celih števil.
  - Kroženje (rotacija): operand se obravnava kot krožni zapis bitov, tako da sta njegov najmanjši in najpomembnejši bit dejansko sosednja.
  - Kroženje (rotacija) s prenosom: bit prenosa in operand se skupaj obravnavata kot krožni zapis bitov.

- Arithmetic of integers
  - Addition: A added to B, the sum appears at Y and carry-out at Status.
  - Addition with carry: A, B and carry-in are summed, the sum appears at Y and carry-out at Status.
  - Subtract: B is subtracted from A (or vice versa) and the difference appears at Y and carry-out (borrow out) as Status.
  - Subtract with borrow: B is subtracted from A (or vice versa) with borrow (carry-in) and the difference appears at Y and carry-out (borrow out) as Status.
  - Two's complement: A (or B) is subtracted from zero and the difference appears at Y.
  - Increment: A (B) is increased by one and the resulting value appears at Y.
  - Decrement: A (B) is decreased by one and the resulting value appears at Y.
  - Pass through: all bits of A (B) appear unmodified at Y (used to determine the parity of A, checking if it is zero or negative, or to load A into a processor register).
- Bitwise logical operations
  - AND: the bitwise AND of A and B appears at Y.
  - OR: the bitwise OR of A and B appears at Y.
  - Exclusive-OR: the bitwise XOR of A and B appears at Y.
  - Ones' complement: all bits of A (or B) are inverted and appear at Y.
- Shift operations
  - Arithmetic shift: the operand is treated as a two's complement integer, meaning that the most significant bit is a "sign" bit and is preserved.
  - Logical shift: a logic zero is shifted into the operand. This is used to shift unsigned integers.
  - Circular shift (rotate): the operand is treated as a circular buffer of bits so its least and most significant bits are effectively adjacent.
  - Circular shift (rotate) through carry: the carry bit and operand are collectively treated as a circular buffer of bits.

Predstavitev celih števil / Integer Representation

- V sistemu binarnih števil lahko poljubna števila predstavljamo s:
  - številkami 0 in 1
    npr. $41_{10} = 00101001_{2,}$
  - minusom za negativna števila,
  - vejico za realna števila
    npr. $-13{,}3125_{10} = -1101{,}0101_2$
- A hkrati na računalniku nimamo posebnega znaka za minus ali decimalno vejico
- Za prikaz števil lahko uporabljajo samo binarne številke (0,1).

- In the binary number system arbitrary numbers can be represented with:
  - the digits zero and one
    e.g. $41_{10} = 00101001_2$,
  - the minus sign (for negative numbers),
  - the period, or ***radix point*** (for numbers with a fractional component)
    e.g. $-13.3125_{10} = -1101.0101_2$.
- 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 podrobno / 2 representations in detail

1. Predznak-velikost
2. Dvojiški komplement

Ostale

1. Z odmikom (binarni odmik)
2. Eniški komplement
3. Osnova -2

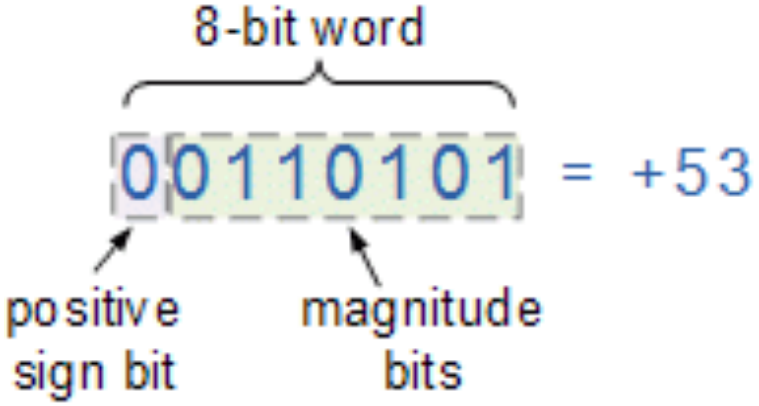
1. Sign-magnitude
2. Two's Complement

Others

1. Biased (also offset binary, excess code)
2. One's complement
3. Base -2

Predznak-velikost (P-V) / Sign-Magnitude (S-M)

- 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

Uporabite 8 bitni zapis

Use the 8 bit format

-6_{10}

123_{10}

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)

Težave P-V / Problems of S-M

- Težave
  - pri seštevanju in odštevanju je treba upoštevati tako znake števil kot njihove relativne velikosti, da se izvede zahtevana operacija,
  - 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

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

Eniški komplement / One's complement

- Negacijo števila v dvojiški notaciji dobimo tako, da invertamo vse števke:
  - 0 postanejo 1 in obratno.
- Primer: spremeni +28 v-28

- Negative notation of an integer id obtained by inverting all digits:
  - 0 become 1 and vice versa.
- Example: change +28 to-28

00011100

11100011

- Težave:
  - 2 ničli,
  - seštevanje in odštevanje zahtevata prišteti h končnemu rezultatu med računanjem pridobljenega končnega "ena dalje".

- Problems:
  - 2 zeros,
  - addition and subtraction require to add any "carry back" (or end-around carry) to the result.

```
    binary    decimal
   11111110     -1
+  00000010     +2
──────────────  ──
 1 00000000      0   ← Not the correct answer
         1      +1   ← Add carry
──────────────  ──
   00000001      1   ← Correct answer
```

Dvojiški komplement / Two's complement

- Za zapis negativnega celega števila v dvojiškem komplementu, napišemo številko v dvojišk notaciji, invertamo (Eniški komplement) in prištejemo 1.
- Primer: spremeni +28 v-28

- To get the two's complement negative notation of an integer, we write out the number in binary then invert the digits (One's complement), and add one to the result.
- Example: change +28 v-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 (eniški komplement)

0 become 1 and vice versa (one's complement)

00011011

prištejemo 1

we add 1 to the result

00011100

–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

Najmanjše negativno število/Lowest negative number:

-2^{n-1}

Največje pozitivno število/Highest positive number:

$2^{n-1} - 1$

Splošna predstavitev DK / General representation of TC

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

Najbolj uteženi bit ni le predznak (1 za negativno in 0 za pozitivno) kot pri P-V, ampak ima tudi utež.

MSB is not only a sign (1 for negative and 0 for positive number) as in S-M, but it has weight as well.

Dokaz:
A + Dvojiški komplement(A) = 0

Proof:
A + Twos complement(A) = 0

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

Razširitev obsega DK / Range Extension TC

- Postopek za P-V ne bo deloval pri dvojiškem komplementu za negativna cela števila.
- The S-M procedure will not work for twos complement negative integers.

+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)

- 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.

- 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	=	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):

- za pozitivna števila to velja (0),
- za negativna pa, ko so:

A = extended range (A):

- for positive integers it is obvious,
- for negative integers only when:

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

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*.

Predstavitev z odmikom / Biased representation

- Predznačeno število n je predstavljeno z bitnim vzorcem, ki ustreza nepredznačenemu številu *n+K*, pri čemer *K* predstavlja *odmik*.
- Ni standarda za binarni odmik, vendar se najpogosteje za *K* pri *n*-bitnem številu uporablja $K = 2^{n-1}$ ali $K = 2^{n-1}-1$ (slednji uporabljan za ekponent v predstavitvi s plavajočo vejico).
- Primer (4 biti, $K = 2^{4-1}-1=7$):

- Signed number *n* is represented by the bit pattern corresponding to the *unsigned number n+K*, with *K* being the *biasing value* or *offset*.
- There is no standard for offset binary, but most often the *K* for an *n*-bit binary word is $K = 2^{n-1}$ or $K = 2^{n-1}-1$ (latter used for exponent in floating point representation).
- Example (4 bits, $K = 2^{4-1}-1=7$):

$$-3_{10}\,(\text{n is signed}) \Rightarrow -3+7\,(\text{n+K is unsigned}) = 4 \Rightarrow \text{biased } 0100$$

- Tako je 0 predstavljen z *K*, *–K* pa je predstavljen s samimi ničlami.

- Thus 0 is represented by *K*, and *–K* is represented by an all-zero bit pattern.
- Also referred to as: offset binary, excess-K,excess-N, excess code

Osnova -2 / Base -2

- V običajnem dvojiškem številskem sistemu je osnova 2; tako skrajni desni bit predstavlja 2^0, naslednji bit predstavlja 2^1, naslednji bit 2^2 itd. Vendar pa je možen tudi dvojiški številski sistem z osnovo −2. Skrajni desni bit predstavlja $(-2)^0 = +1$, naslednji bit predstavlja $(-2)^1 = -2$, naslednji bit $(-2)^2 = +4$ in tako naprej, z izmeničnim predznakom.
- Obseg števil, ki jih je mogoče predstaviti, je asimetričen:
  - če je število dolgo sodo bitov, je velikost največjega negativnega števila, ki ga je mogoče predstaviti, dvakrat večja od največjega pozitivnega števila, ki ga je mogoče predstaviti;
  - in obratno, če je število dolgo liho bitov.

- In conventional binary number systems, the base (radix) is 2; the rightmost bit represents 2^0, the next bit represents 2^1, the next bit 2^2, and so on. However, a binary number system with base −2 is also possible. The rightmost bit represents $(-2)^0 = +1$, the next bit represents $(-2)^1 = -2$, the next bit $(-2)^2 = +4$ and so on, with alternating sign.
- The range of numbers that can be represented is asymmetric:
  - if the integer has an even number of bits, the magnitude of the largest negative integer that can be represented is twice as large as the largest positive integer that can be represented;
  - and vice versa if the integer has an odd number of bits.

Decimal	Unsigned	Sign and magnitude	Ones' complement	Two's complement	Excess-8 (biased) With $K=2^{n-1}=2^3=8$	Excess-7 (biased) $K=2^{n-1}-1=2^3-1=7$	Base −2
+16	N/A	N/A	N/A	N/A	N/A	N/A	N/A
+15	1111	N/A	N/A	N/A	N/A	N/A	N/A
+14	1110	N/A	N/A	N/A	N/A	N/A	N/A
+13	1101	N/A	N/A	N/A	N/A	N/A	N/A
+12	1100	N/A	N/A	N/A	N/A	N/A	N/A
+11	1011	N/A	N/A	N/A	N/A	N/A	N/A
+10	1010	N/A	N/A	N/A	N/A	N/A	N/A
+9	1001	N/A	N/A	N/A	N/A	N/A	N/A
+8	1000	N/A	N/A	N/A	N/A	1111	N/A
+7	0111	0111	0111	0111	1111	1110	N/A
+6	0110	0110	0110	0110	1110	1101	N/A
+5	0101	0101	0101	0101	1101	1100	0101
+4	0100	0100	0100	0100	1100	1011	0100
+3	0011	0011	0011	0011	1011	1010	0111
+2	0010	0010	0010	0010	1010	1001	0110
+1	0001	0001	0001	0001	1001	1000	0001
+0	0000	0000	0000	0000	1000	0111	0000
−0		1000	1111				
−1	N/A	1001	1110	1111	0111	0110	0011
−2	N/A	1010	1101	1110	0110	0101	0010
−3	N/A	1011	1100	1101	0101	0100	1101
−4	N/A	1100	1011	1100	0100	0011	1100
−5	N/A	1101	1010	1011	0011	0010	1111
−6	N/A	1110	1001	1010	0010	0001	1110
−7	N/A	1111	1000	1001	0001	0000	1001
−8	N/A	N/A	N/A	1000	0000	N/A	1000
−9	N/A	N/A	N/A	N/A	N/A	N/A	1011
−10	N/A	N/A	N/A	N/A	N/A	N/A	1010
−11	N/A	N/A	N/A	N/A	N/A	N/A	N/A

Aritmetika celih števil / Integer arithmetic

- Negacija
- Seštevanje in odštevanje
- Množenje
- Deljenje

- Negation
- Addition and subtraction
- Multiplication
- Division

Negacija / Negation

- Dvojiški 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 samo:
- 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
komplement / complement   = 01111111
dodamo 1 / add 1        +           1
rezultat / result         10000000

                  -(-128) = -128
```

Potrebno nadzorovati najbolj uteženi bit (predznak).

Se mora spremeniti med negacijo!

We need to monitor the MSB (sign bit).

It should change during negation!

Seštevanje števil v dvojiškem komplementu

Addition of numbers in twos complement 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 = Prekoračitev/Overflow (e) (+5) + (+4)	1001 = −7 +1010 = −6 10011 = Prekoračitev/Overflow (f) (–7) + (–6)

Pravilo prekoračitve / Overflow rule

- Pri seštevanju dveh pozitivnih ali dveh negativnih števil, se prekoračitev 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.

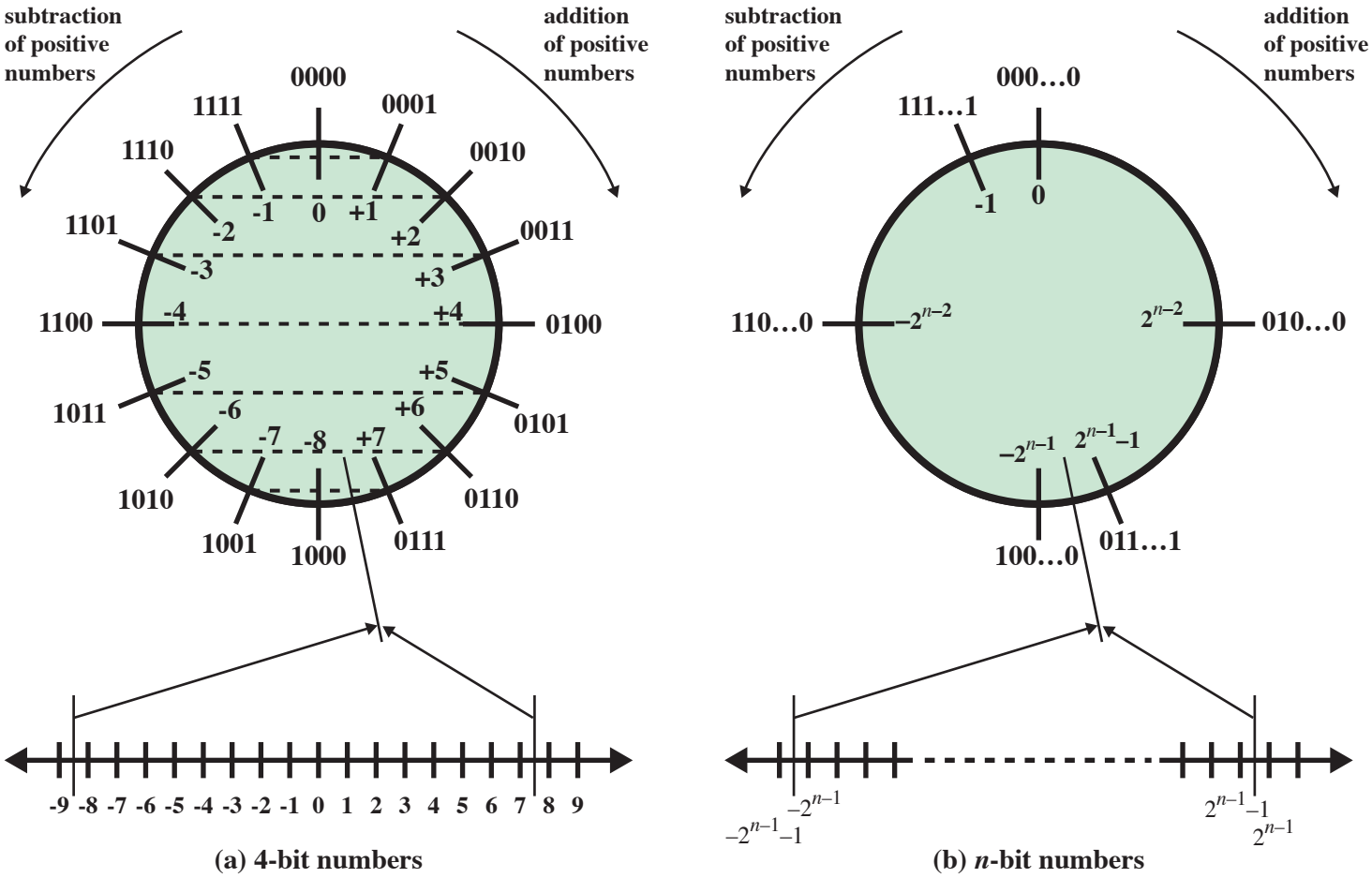
Pravilo odštevanja / Subtraction Rule

- 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.

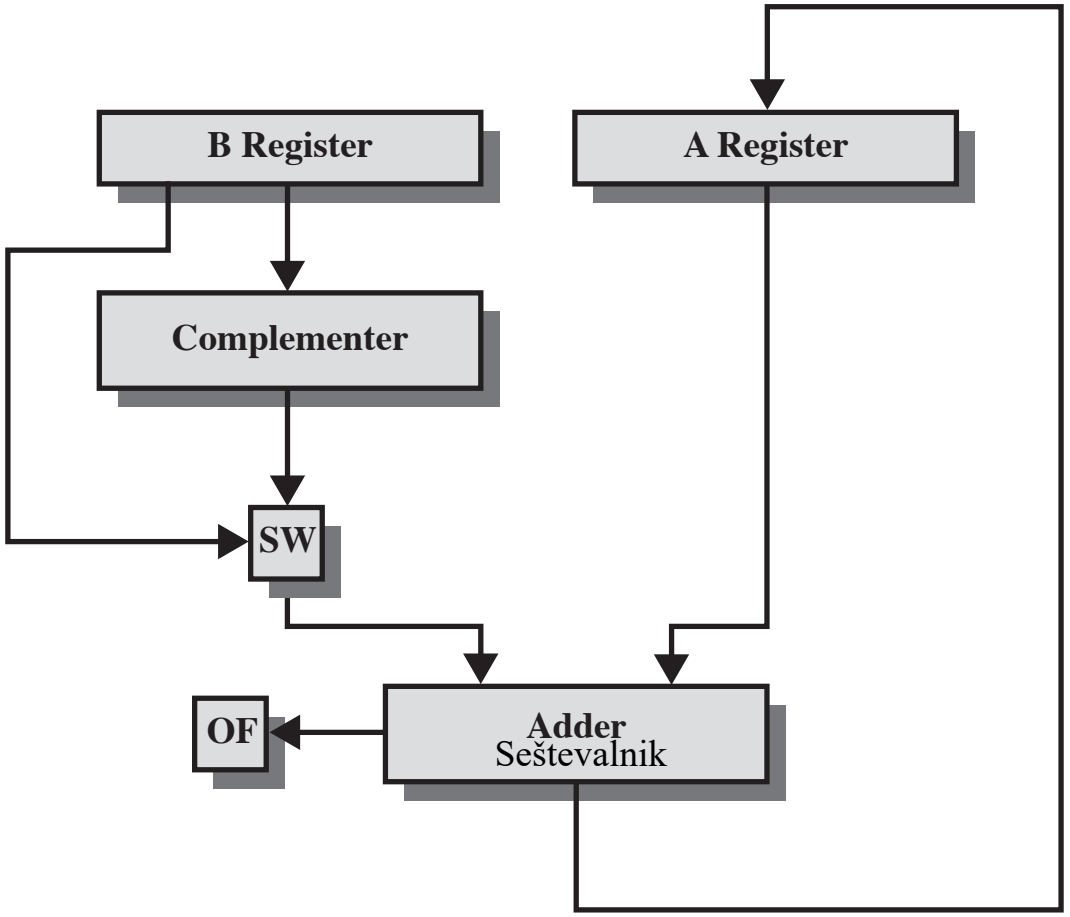
Primeri: odštevanje števil v dvojiškem komplementu

Examples: subtraction of numbers in twos complement representation

2-7=? 0010 = 2 +1001 = −7 1011 = −5	5-2=? 0101 = 5 +1110 = −2 10011 = 3
(a) M = 2 = 0010 S = 7 = 0111 −S = 1001	(b) M = 5 = 0101 S = 2 = 0010 −S = 1110
-5-2=? 1011 = −5 +1110 = −2 11001 = −7	5-(-2)=? 0101 = 5 +0010 = 2 0111 = 7
(c) M =−5 = 1011 S = 2 = 0010 −S = 1110	(d) M = 5 = 0101 S =−2 = 1110 −S = 0010
7-(-7)=? 0111 = 7 +0111 = 7 1110 = Prekoračitev/ Overflow	-6-4=? 1010 = −6 +1100 = −4 10110 = Prekoračitev/ Overflow
(e) M = 7 = 0111 S = −7 = 1001 −S = 0111	(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

```
   1011
 × 1101
 ------
   1011
  0000
 1011
1011
--------
10001111
```

množenec (11)
množitelj (13)

multiplicand (11)
multiplier (13)

delni zmnožki

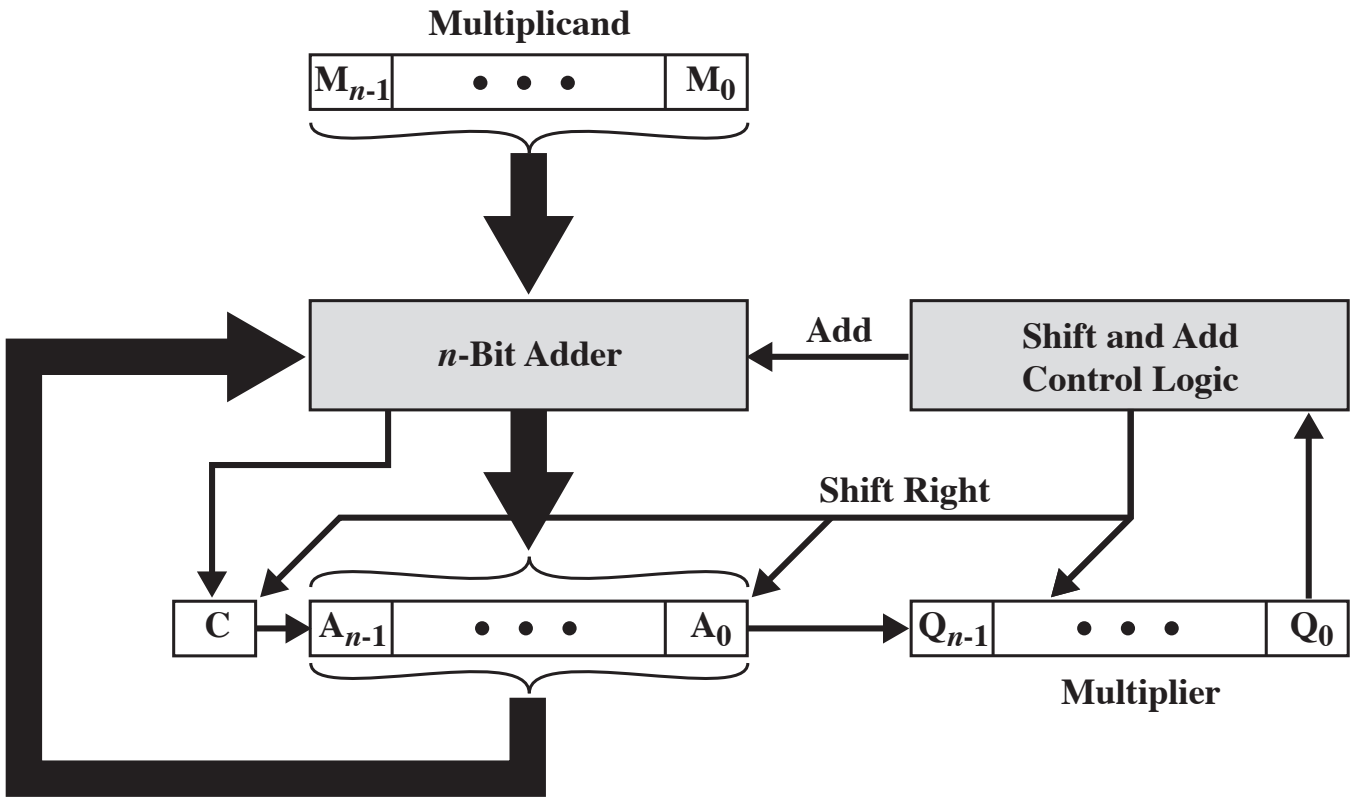
partial products

zmnožek (143)

product (143)

Strojna izvedba nepredznačenega binarnega množenja - blokovni diagram

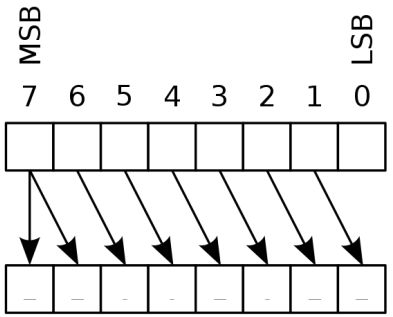
Hardware Implementation of Unsigned Binary Multiplication – Block Diagram



Logični in aritmetični premik / Logical and arithmetic shift

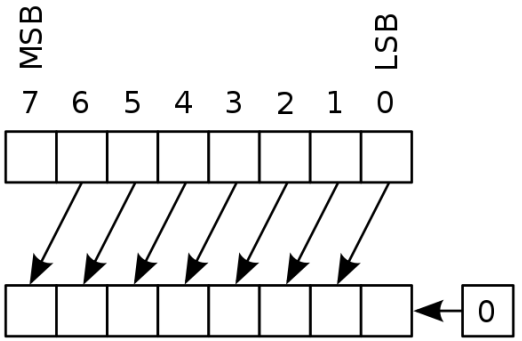
Signed binary numbers

Arithmetic right



Division

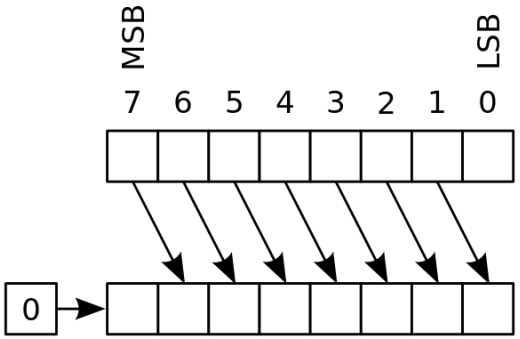
Arithmetic left



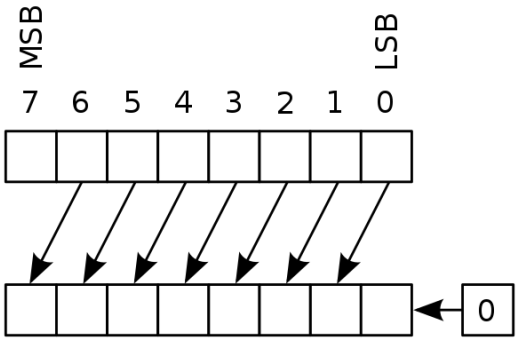
Multiplication

Unsigned binary numbers

Logical right



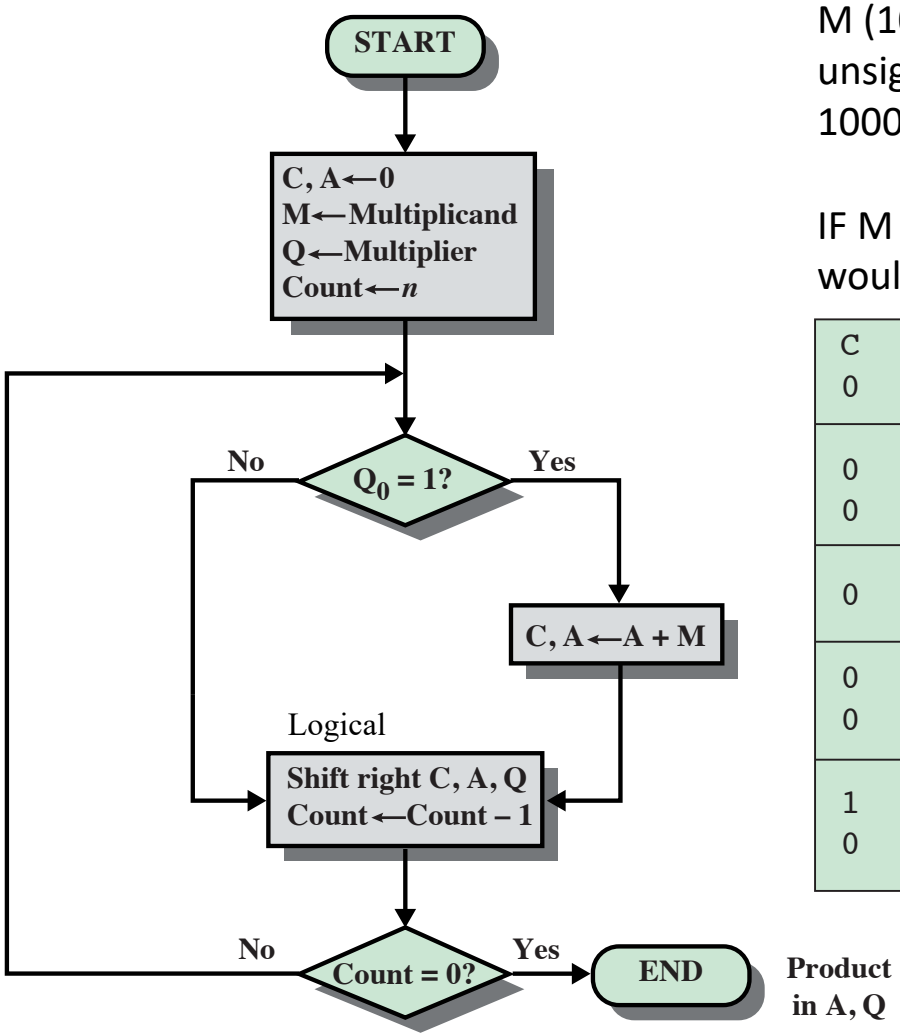
Logical left



Poskusimo rešiti / Try to solve it

M (1011_2 or 11_{10}) and Q (1101_2 or 13_{10}) are unsigned binary numbers. Result AQ is 10001111_2 (143_{10}).

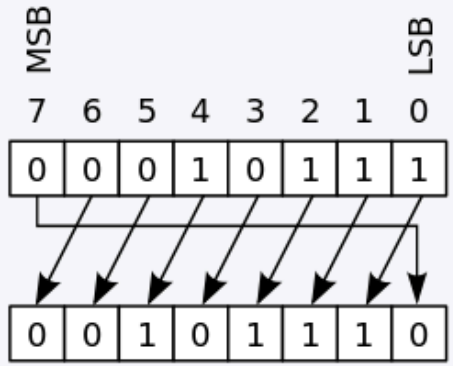
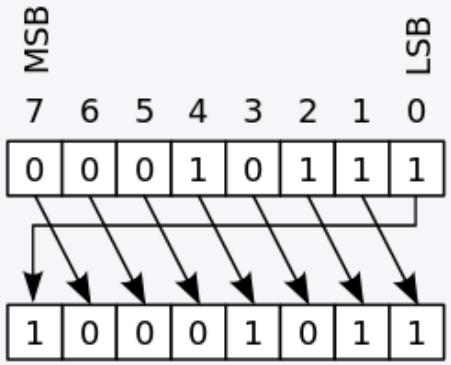
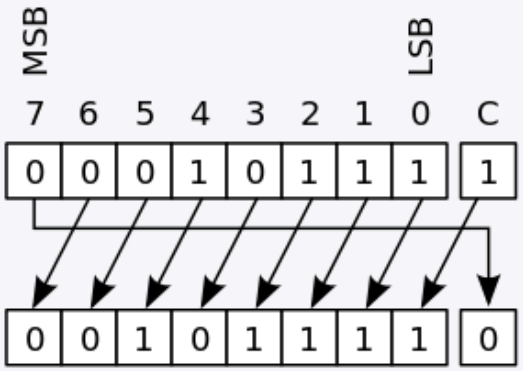
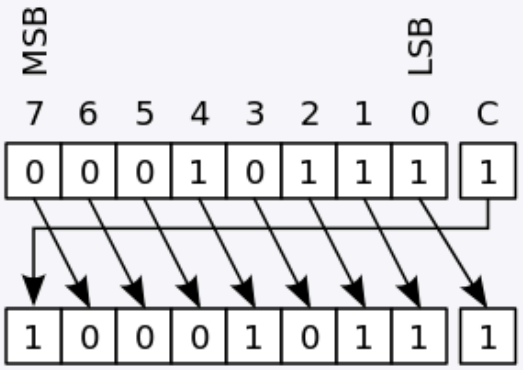
IF M (-5_{10}) and Q (-3_{10}) would be T-C, the result would be -113_{10} which is incorrect.



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

Ostale operacije premikanja / Other shift operations

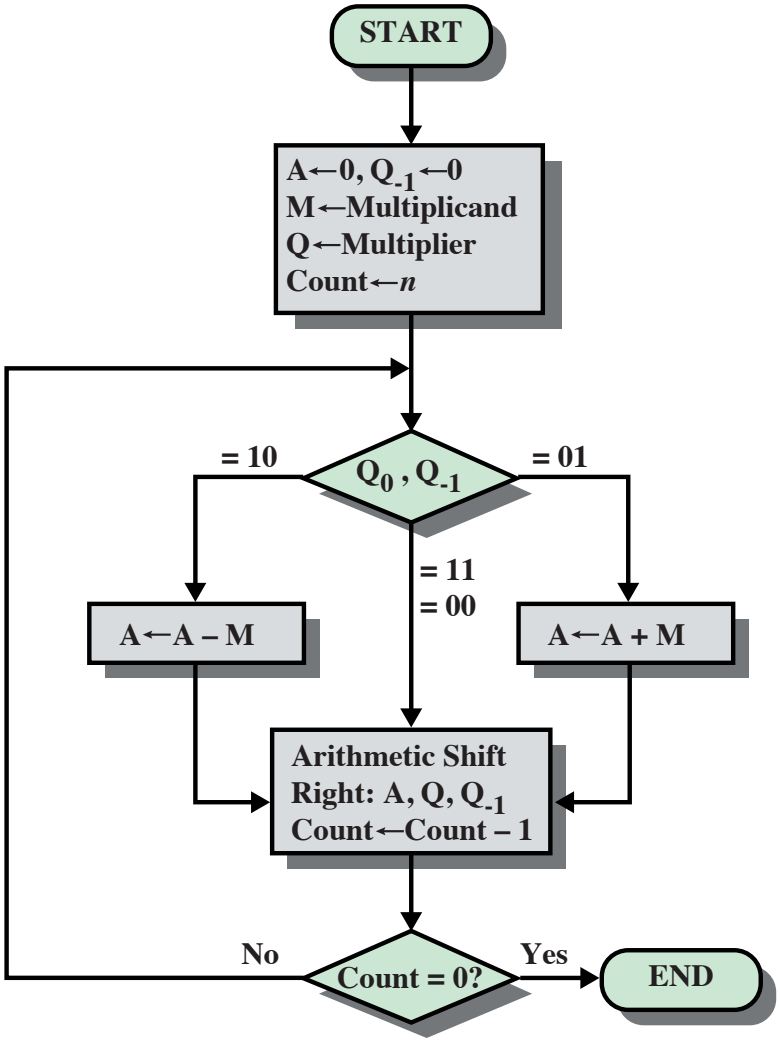
Rotacija Rotate		
Rotacija s prenosom Rotate through carry		

Množenje negativnih števil / Multiplying Negative Numbers

- Prejšnja rešitev ne dela.
- Rešitev 1
  - Pretvorimo v pozitivna števila, če je potrebno.
  - Množimo kot prej.
  - Če sta predznaka različna, negiramo rezultat.
- Rešitev 2
  - Boothov algoritem

- Previous solution 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 alghorithm



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

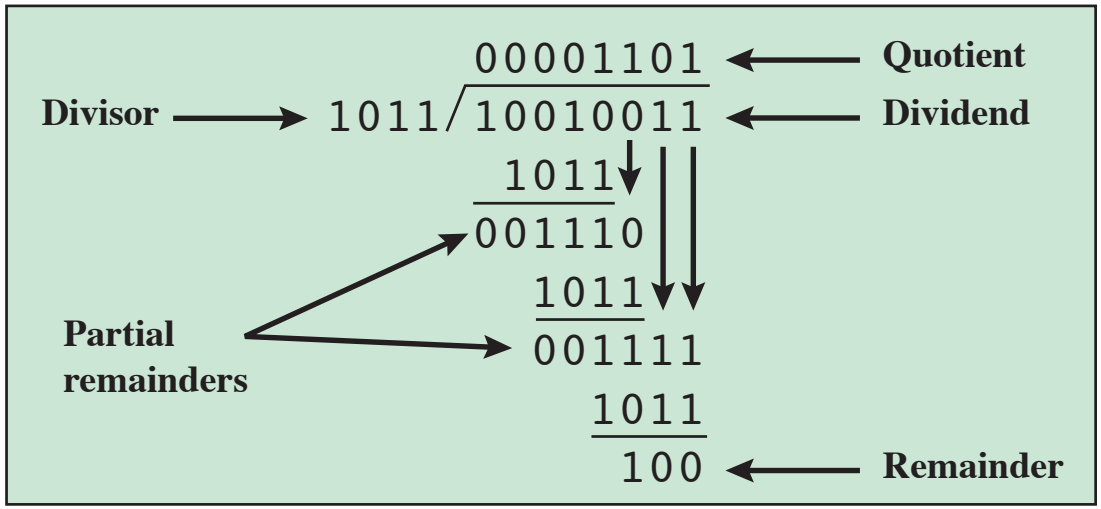
Poskusimo rešiti / Try to solve it

- Bolj kompleksno kot množenje
- Negativna števila so problematična!

- More complex than multiplication
- Negative numbers are problematic!

deljitelj

delni ostanki



količnik

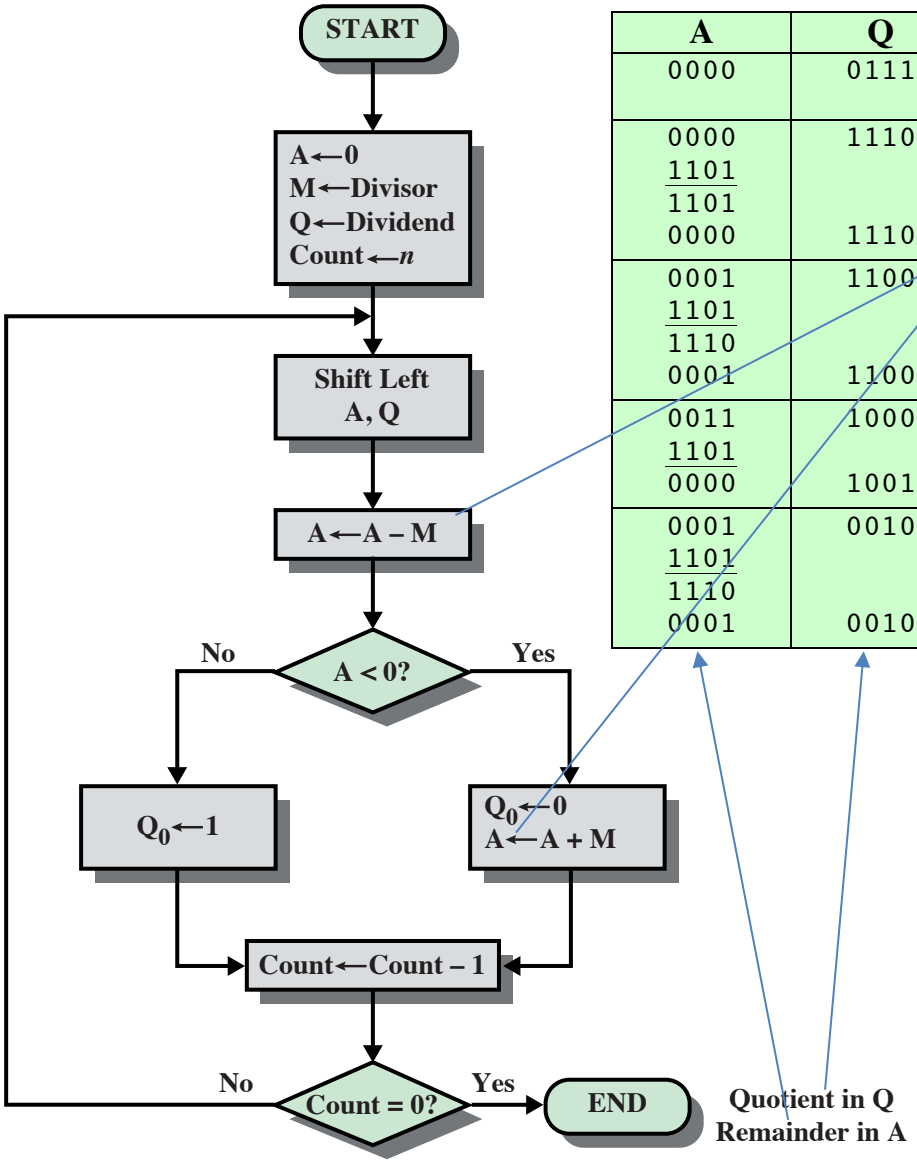
deljenec

ostanek

Division of unsigned binary numbers

Poskusimo rešiti / Try to solve it

Vzemimo 7 in 3 kot nepredznačeni binarni števili /
Take 7 and 3 as unsigned binary numbers



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

Q M

7 / 3 = 0111 / 0011

Pri deljenju v dvojiškem komplementu pretvorimo operande v nepredznačene in na koncu po potrebi upoštevamo predznaka Q in R. /
For two's complement division convert the operands in unsigned values and, at the end, account for the signs Q and R where needed.

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 (zgoraj zapisano s . v angleški notaciji)?
- Premična?
  - Kako pokazati kje se nahaja?

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

Predstavitev s fiksno vejico / Fixed-Point Representation

- Vse do sedaj predstavljene predstavitve so s fiksno vejico:

  Položaj dvojiške vejice je fiksen in se predpostavlja, da je desno od skrajne desne števke.

- Programer lahko enako predstavitev uporabi za binarne ulomke s skaliranjem števil, tako da je binarna vejica implicitno postavljena na neko drugo mesto.

- All so far presented representations are fixed-point:

  The radix point (binary point) is fixed and assumed to be to the right of the rightmost digit.

- The 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.

- Z zapisom s fiksno vejico je mogoče predstavljati obseg pozitivnih in negativnih celih in realnih števil, osredotočenih okoli 0.

- Omejitve:
  - ni mogoče predstaviti zelo velikih števil in zelo majhnih deležev (ne-celi del števila),
  - delež količnika pri deljenju dveh velikih števil se lahko izgubi.

- With a fixed-point notation it is possible to represent a range of positive and negative integers and real numbers 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.

Znanstvena predstavitev / Scientific notation

- Da bi obšli omejitev predstavitve s fiksno vejico, lahko uporabimo znanstveni zapis.
- V decimalnem sistemu pomaknemo vejico (. v spodnjem zapisu angleške notacije) na priročno mesto in uporabimo eksponent števila 10, da vemo kje se vejica nahaja.

- To get around the limitation of a fixed radix point representation we can use scientific notation.
- In decimal sytem we slide the decimal point to a convenient location and use the exponent of 10 to track the location of the point.

$$976{,}000{,}000{,}000{,}000 = 9.76 * 10^{14}$$
$$0.0000000000000976 = 9.76 * 10^{-14}$$

- Enak pristop je mogoče uporabiti pri binarnih številih.

- The same approach can be taken with binary numbers.

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

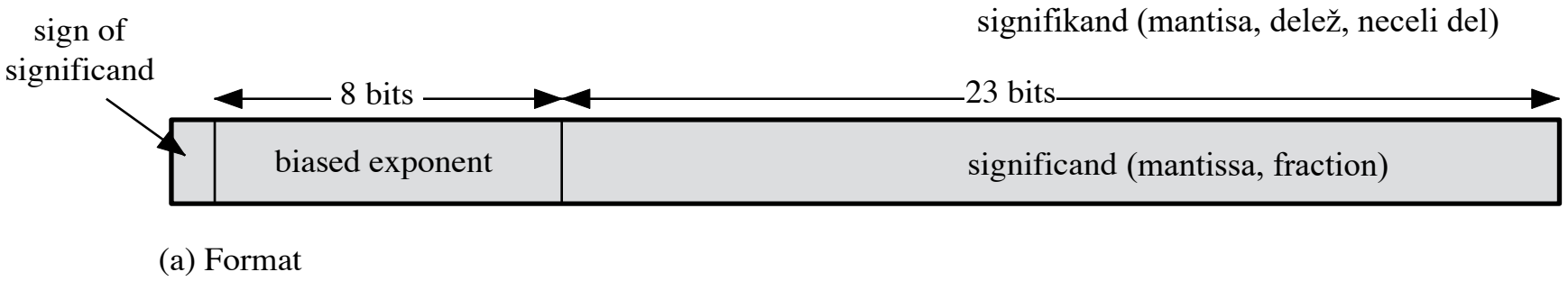
- Sign: plus or minus
- Significand S
- Exponent E

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

- Napačno poimenovanje?
  - Vejica je v resnici fiksna med eksponentom in mantiso.
- Eksponent določa položaj binarne vejice.

- Wrong naming?
  - The radix point is fixed between the exponent and the mantissa.
- 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

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

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

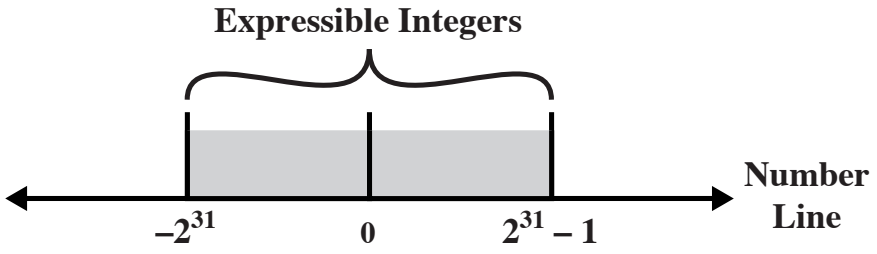
$1.1010001 \times 2^{-10100} = 0\ 01101011\ 10100010000000000000000 = 1.6328125 \times 2^{-20}$

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

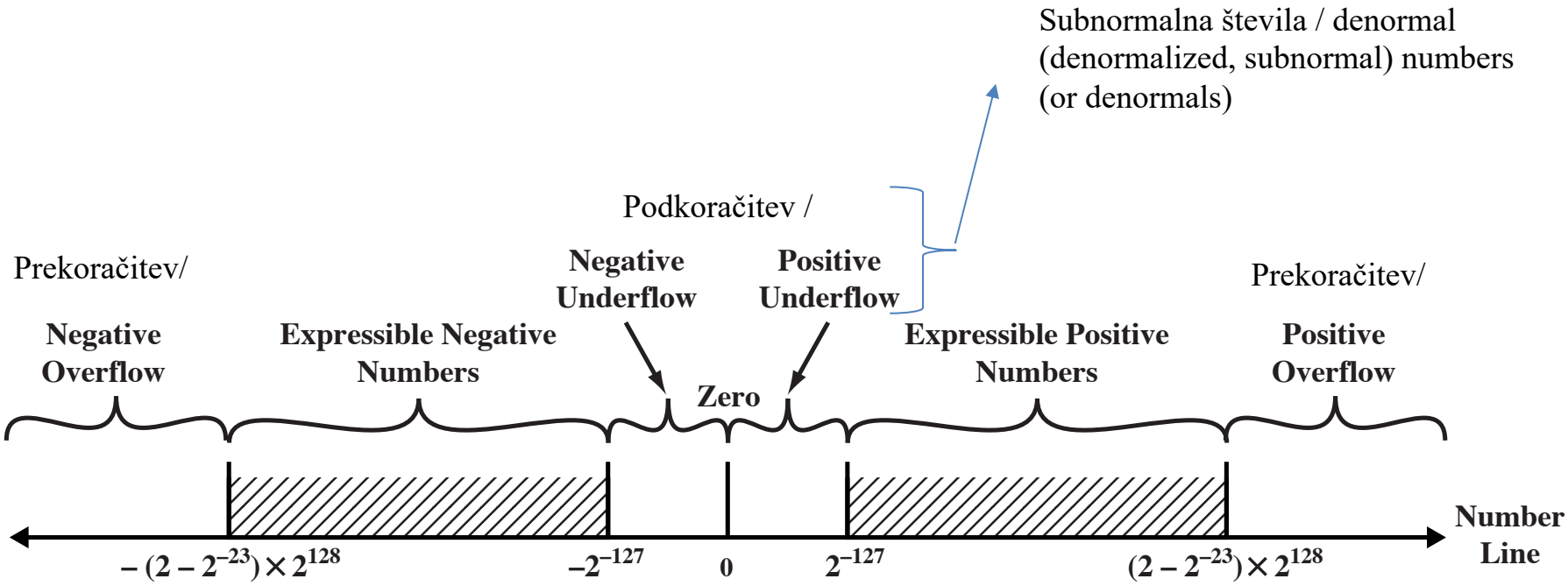
Deli predstavitve PV / Parts of the FP representation

- Eksponent je v predstavitvi z odmikom.
- V 32-bit številu v PV:
  - dolžina: 8 bitov,
  - možne vrednosti 0-255,
  - odštejemo 127 za dejansko vrednost,
  - obseg-127 do +128.
- Signifikand (mantisa):
  - 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.
- In 32-bit FP:
  - Length: 8 bits
  - Possible values 0-255
  - Subtract 127 to get the real value
  - Range-127 do +128
- Significand (mantissa):
  - FP numbers are usually normalised.
    - I.e. the exponent is aligned such that the leading bit (MSB) is 1
  - Since it is always 1, there is no need to present it.

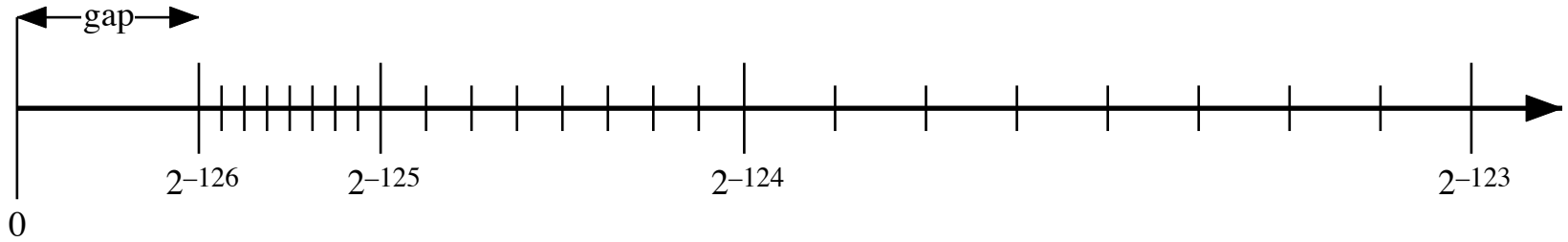


(a) Twos Complement Integers



(b) Floating-Point Numbers

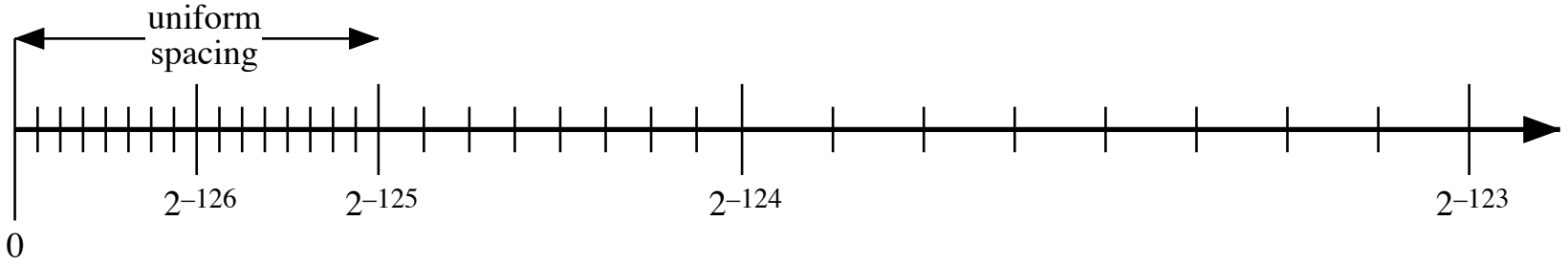
Expressible Numbers in Typical 32-Bit Formats



(a) 32-bit format without subnormal numbers

Z dodajanjem subnormalnih števil se med 0 in 2^{-126} enakomerno doda dodatnih 2^{23}-1 števil.

With the addition of subnormal numbers, an additional 2^{23} - 1 numbers are uniformly added between 0 and 2^{-126}.

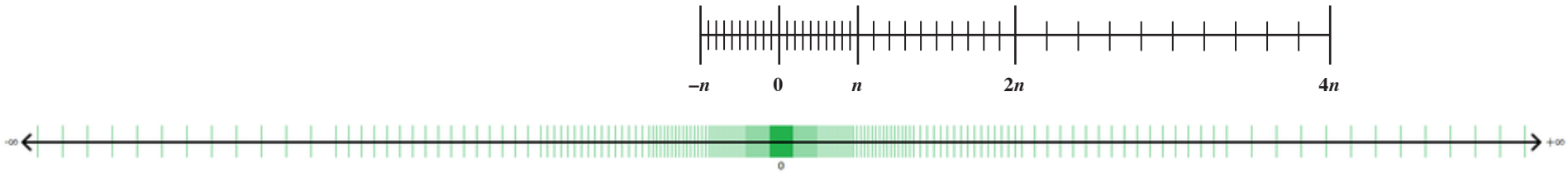


(b) 32-bit format with subnormal numbers

The Effect of IEEE 754 Subnormal Numbers

Gostota FP števil / Density pf FP numbers

- Možne vrednosti so si blizu izvora bližje, z oddaljevanjem pa se oddaljujejo ena od druge.
- The possible values get closer together near the origin and farther apart as you move away.



- Izračune je treba zaokrožiti na najbližjo vrednost, ki jo lahko zapis predsatvi.
- Kompromis med razponom in natančnostjo: več bitov za eksponent omogoča večji razpon na račun manjše natančnosti.
- Edina možnost je povečat število bitov za eksponent in mantiso.

- Calculations have to be rounded to the nearest value that the notation can represent.
- A trade-off between range and precision: a larger exponent base gives a greater range at the expense of less precision.
- Only solution is to increase the number of bits for exponent and significand.

IEEE Standard 754(-2019 ISO/IEC 60559:2020)

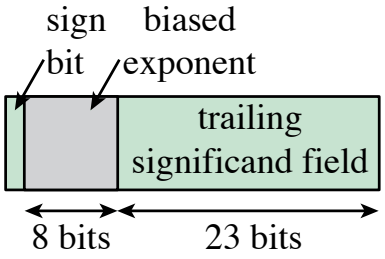
- Opredeljuje najpomembnejši prikaz s plavajočo vejico.
- Standard je bil razvit za lažjo prenosljivost programov z enega procesorja na drugega in za spodbujanje razvoja prefinjenih numerično usmerjenih programov.
- Standard je bil široko sprejet in se uporablja na skoraj vseh sodobnih procesorjih in aritmetičnih koprocesorjih.
- IEEE 754-2019 zajema binarne in decimalne predstavitve s plavajočo vejico.

- Defines most important floating-point representation.
- 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 754-2019 covers both binary and decimal floating-point representations.

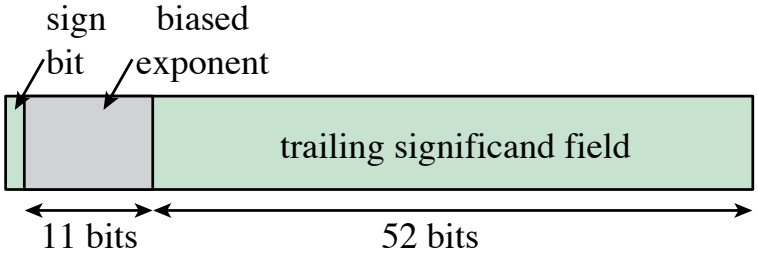
IEEE 754 Format Parameters

Name	Common name	Base	Significand bits*	Decimal digits	Exp. bits	Decimal Exp. max	Exponent bias	Exp. min	Exp. max	No. of Exp.	No. of values	Smallest positive normal number	Largest positive normal number	Smallest subnorm. magnit.
binary16	Half precision	2	10	3.31	5	4.51	$2^4-1 = 15$	−14	15	30				
binary32	Single precision	2	23	7.22	8	38.23	$2^7-1 = 127$	−126	127	254	$1.98*2^{31}$	2^{-126}	$2^{128}-2^{104}$	2^{-149}
binary64	Double precision	2	52	15.95	11	307.95	$2^{10}-1 = 1023$	−1022	1023	2046	$1.99*2^{63}$	2^{-1022}	$2^{1024}-2^{971}$	2^{-1074}
binary128	Quadruple precision	2	112	34.02	15	4931.77	$2^{14}-1 = 16383$	−16382	16383	32766	$1.99*21^{28}$	2^{-16362}	$2^{16384}-2^{16271}$	2^{-16494}
binary256	Octuple precision	2	236	71.34	19	78913.2	$2^{18}-1 = 262143$	−262142	262143	524286				
decimal32		10	7	7	7.58	96	101	−95	96	192				
decimal64		10	16	16	9.58	384	398	−383	384	768				
decimal128		10	34	34	13.58	6144	6176	−6143	6144	12288				

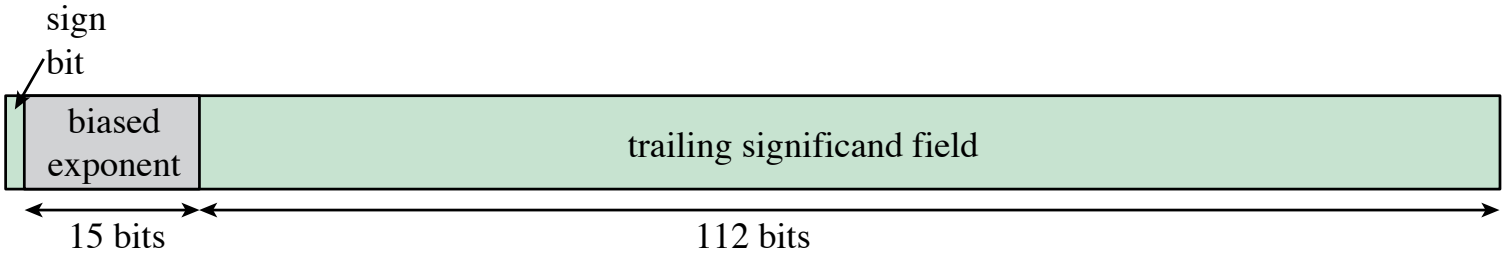
* Not including sign & implied bit. Number of fractions is $2^{\text{significand bits}}$



(a) binary32 format



(b) binary64 format



(c) binary128 format

IEEE 754-2019

Določa naslednje različne vrste formatov :

- 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 osem predstavitev s plavajočo vejico, pet binarnih in tri decimalne, katerih kodiranje določa standard in jih je mogoče uporabiti za aritmetiko.
- 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:

- 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 eight floating-point representations, five binary and three decimal, whose encodings are specified by the standard, and which can be used for arithmetic.
- Interchange format
  - A fully specified, fixed-length binary encoding that allows data interchange between different platforms and that can be used for storage.

IEEE 754-2008 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		

Dodatne oblike / Additional formats (1)

Razširjene natančne oblike

- Natančna oblika je odvisna od izvedbe, vendar standard določa določene omejitve za dolžine eksponentov in signifikantov.
- Ti formati so tipi aritmetičnih formatov ne pa tudi tipi izmenjevalnih formatov.
- Omogočajo dodatne bite v eksponentu (razširjeni obseg) in v mantisi (razširjena natančnost).
- Se uporablja za vmesne izračune
  - Zmanjšamo pretirano napako končnega rezultata, ki lahko nastane z zaokroževanjem.
  - Zmanjšamo 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.

Extended precision formats

- The exact format is implementation dependent, but the standard places certain constraints on the length of the exponent and significand.
- These formats are arithmetic format types but not interchange format types.
- Provide additional bits in the exponent (extended range) and in the significand (extended precision).
- Used for intermediate calculations
  - 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.

Dodatne oblike / Additional formats (2)

Razširljive natančne oblike

- Natančnost in obseg sta določena od in pod nadzorom uporabnika.
- Lahko se uporablja za vmesne izračune, vendar standardni ne omejujejo ne oblike ne dolžine.

Extendable precision form

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

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	–∞
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 < 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 ≠ 0	$2^{e-126}(0.f)$
negative subnormal	1	0	f ≠ 0	$-2^{e-126}(0.f)$

(a) binary32 format

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 ≠ 0	$2_{e-1022}(0.f)$
negative subnormal	1	0	f ≠ 0	$-2_{e-1022}(0.f)$

(a) binary64 format

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	–∞
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	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 ≠ 0	$2_{e-16383}(0.f)$
negative subnormal	1	0	f ≠ 0	$-2_{e-16383}(0.f)$

(a) binary128 format

FP Aritmetične operacije / FP Arithmetic Operations

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

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 eksponenta: npr.-200 <-127
- Prekoračitev significanda
  - Se lahko reši z zaokroževanjem
- Podkoračitev significanda
  - Se lahko reši s poravnavo

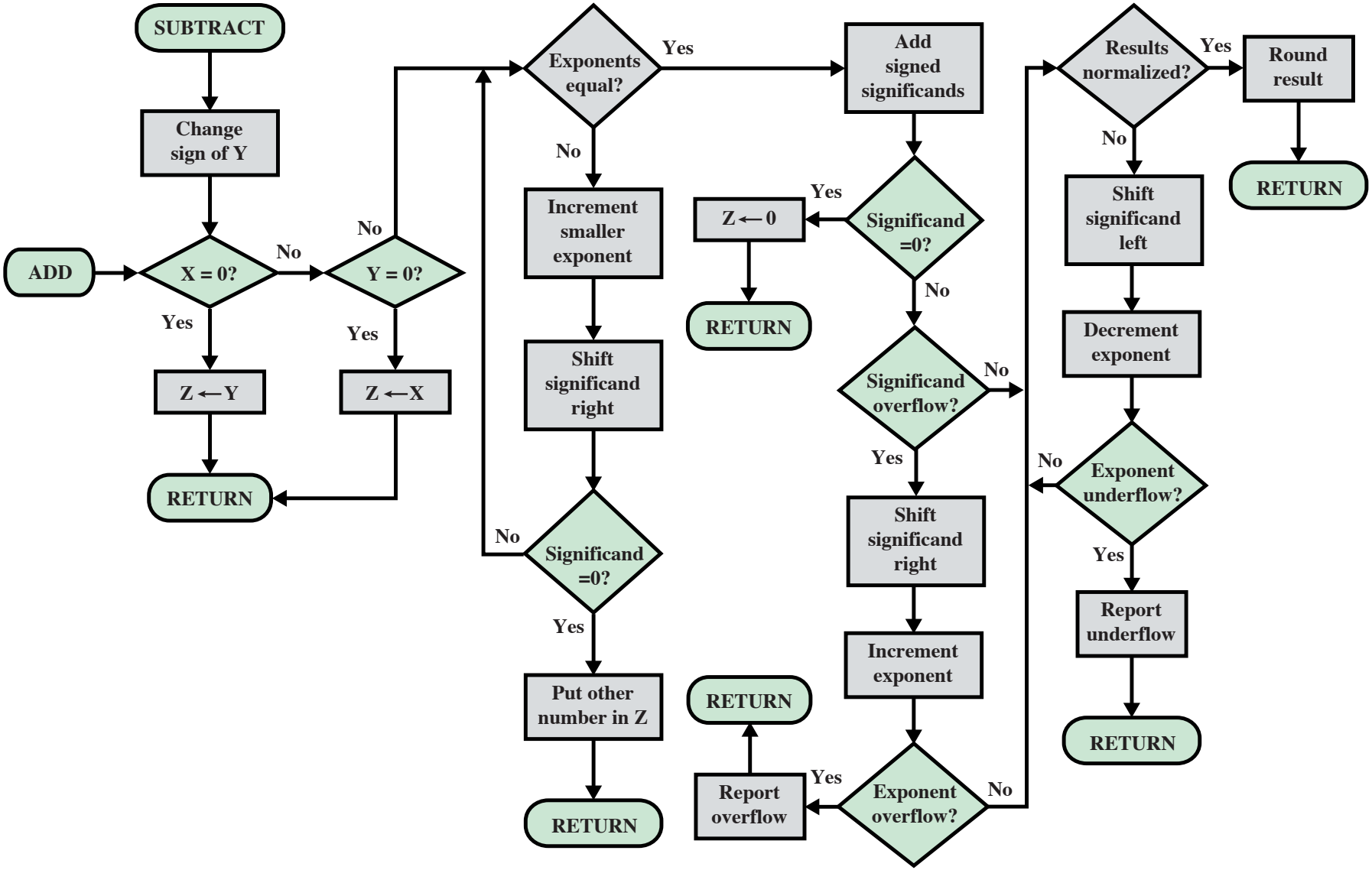
A floating-point operation may produce also one of these conditions:

- Exponent overflow: + ∞ or - ∞.
- Exponent underflow: e.g. -200 < -127
- Significand underflow
  - May be solved with rounding
- Significand overflow
  - May be solved by realignment

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

- Preverimo ali gre za ničlo.
- Poravnamo mantisi s popravljanjem eksponenta.
- Mantisi seštejemo ali odštejemo.
- Normaliziramo rezultat.

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

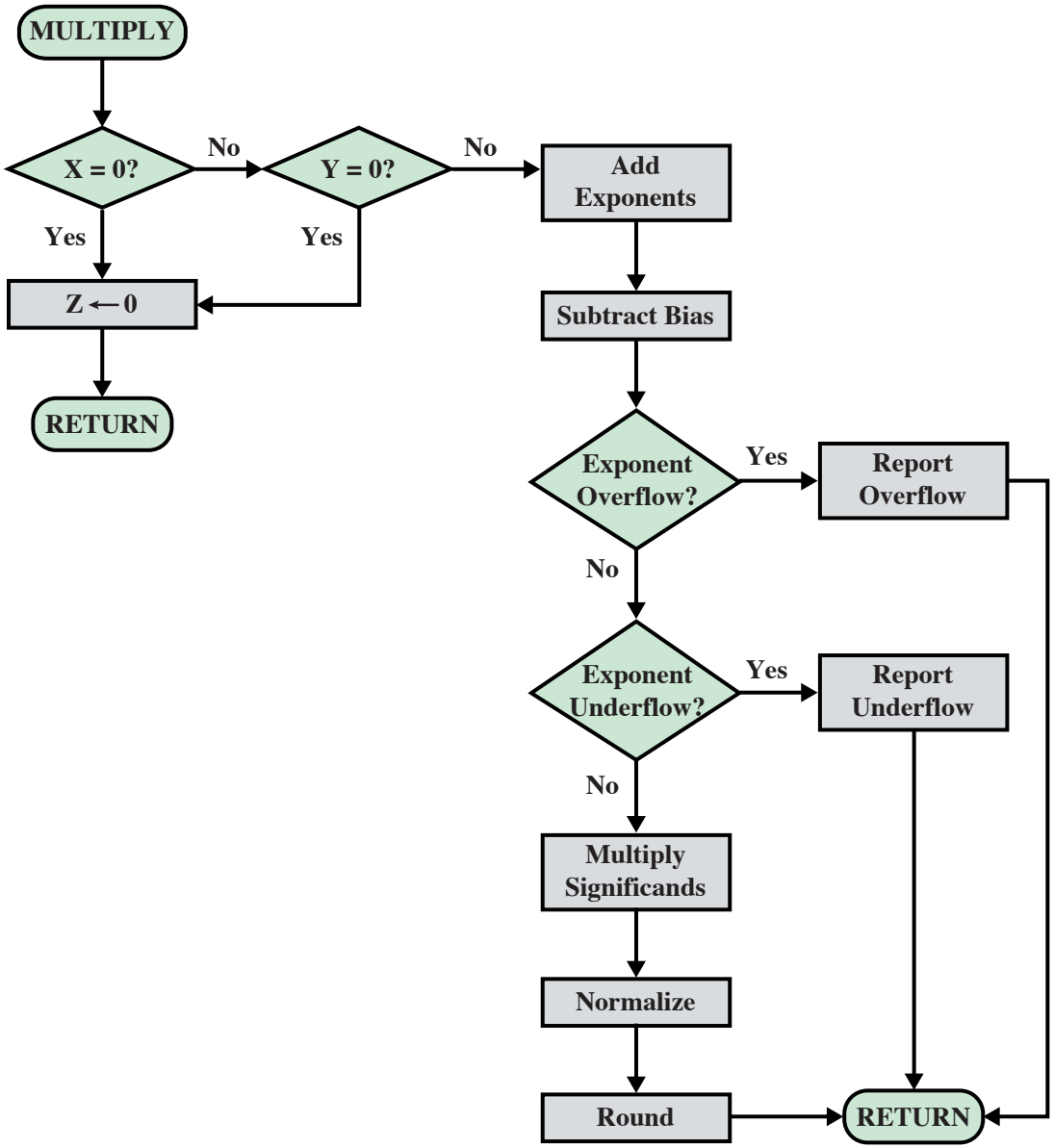


Floating-Point Addition and Subtraction (Z ← X ± Y)

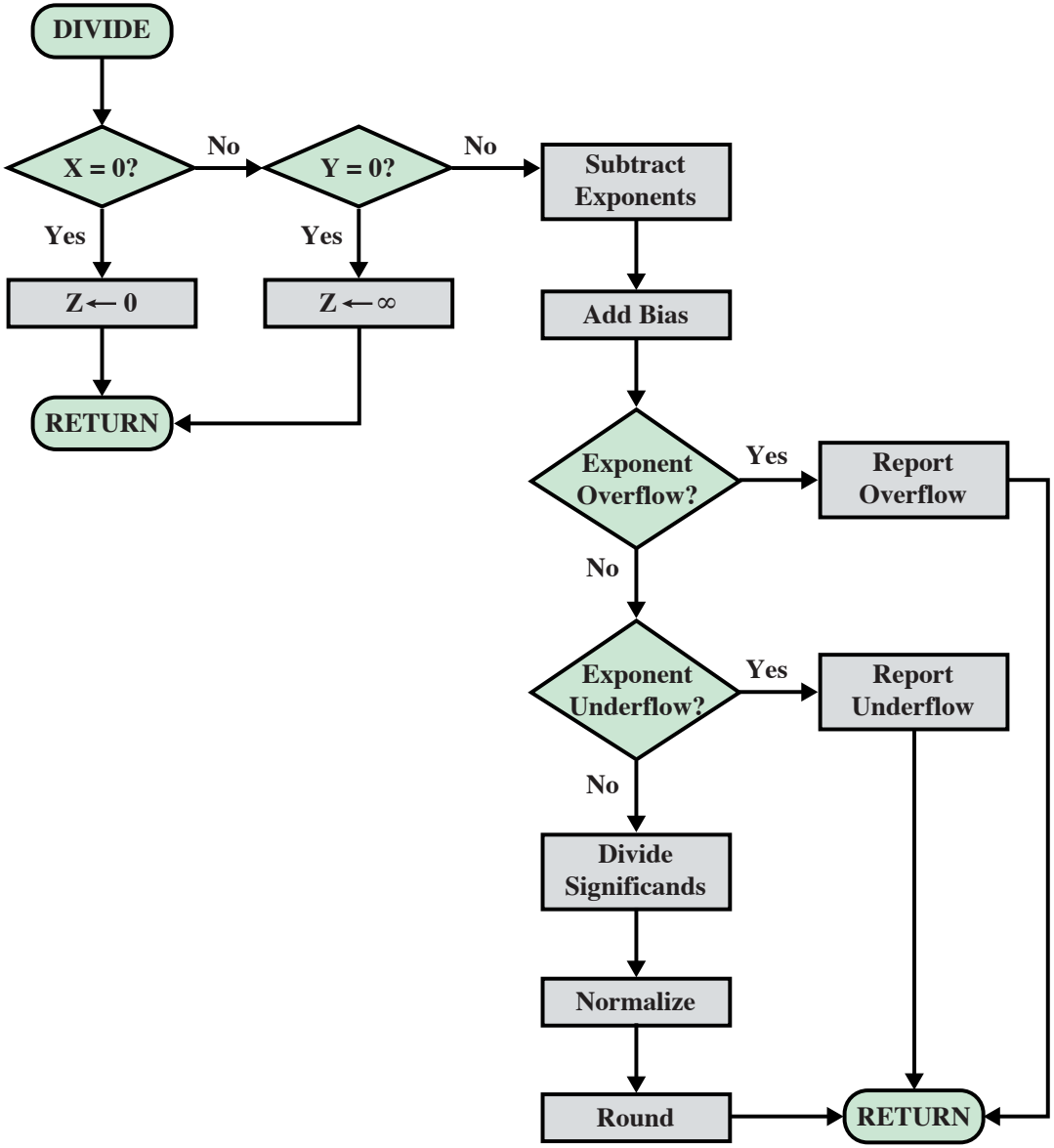
FP aritmetika × in ÷ / FP arithmetic of × and ÷

- Preverimo, če je kateri od operandov enak nič.
- Eksponenta se seštejeta ali odštejeta.
- Mantisi se zmnožita ali delita.
- Število se normalizira.
- Nato se zaokroži.
- Vmesni rezultat mora biti shranjen v pomnilniku dvojne dolžine.

- Check for zero.
- The exponent is summed or subtracted.
- Significands are multiplied or divided.
- The number is normalised.
- 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

Odštevanje števil zelo podobnih vrednosti substract numbers very close in value

$x = 1.00\ldots00 * 2^1$ and $y = 1.11\ldots11 * 2^0$

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

(a) Binary example, without guard bits

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

(c) Hexadecimal example, without guard bits

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

(b) Binary example, with guard bits

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

(d) Hexadecimal example, with guard bits

Natančnost / Precision Considerations

Rezultat katere koli operacije s signifikanti se običajno shrani v daljši register. Ko rezultat ponovno shranimo v format s plavajočo vejico, je treba dodatne bite odstraniti. To je zaokroževanje.

Standardni pristopi IEEE:

- Zaokrožitev do najbližje:
  - Rezultat je zaokrožen na najbližjo predstavljivo številko. Kaj pa, če je na polovici?
- Zaokrožitev proti + ∞:
  - Rezultat je zaokrožen proti pozitivni neskončnosti.
- Zaokrožitev proti -∞:
  - Rezultat je zaokrožen navzdol proti negativni neskončnosti.
- Zaokrožitev proti 0:
  - Rezultat se zaokroži proti nič.

The result of any operation on the significands is generally stored in a longer register. When the result is put back into the floating-point format, the extra bits must be eliminated. This is rounding.

IEEE rounding approaches:

- Round to nearest:
  - The result is rounded to the nearest representable number. What if it is in the middle?
- 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

Zaokrožitev proti + in *- neskončnosti* je 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.
  - Če končne točke niso predstavljive, se intervalne končne točke zaokrožijo navzdol oziroma navzgor.
- Širina intervala označuje natančnost rezultata.
  - Če je razpon med zgornjo in spodnjo mejo dovolj ozek, smo dobili dovolj natančen rezultat.

Rounding to plus and *minus infinity is* 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.
  - If the endpoints are not representable then the interval endpoints are rounded down and up respectively.
- The width of the interval indicates the accuracy of the result.
  - 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 745 – neskončnost / IEEE 754 - 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:

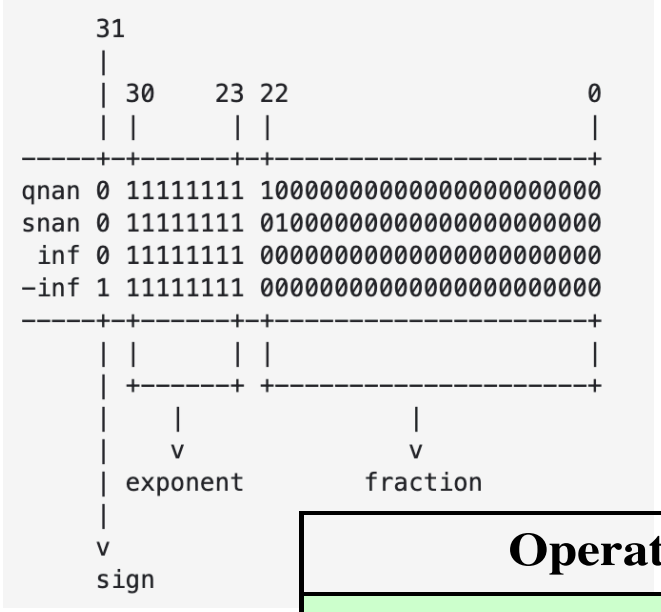
```
-∞ < (every finite number) < +∞
```

Primeri:

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

For example:

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



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 $x < 0$

Vrpašanja / Questions