Informatika 3.
Előadás X: Bónusz

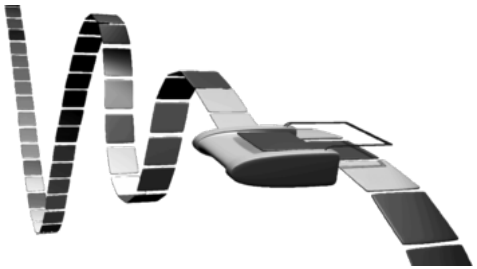
Kovács Kristóf
Wettl Ferenc előadásai alapján

Budapesti Műszaki és Gazdaságtudományi Egyetem

2024-03-26

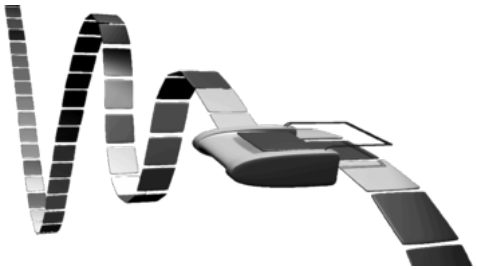
Turing-gép

- A Turing-gép egy
  $M = \langle Q, \Gamma, b, \Sigma, \delta, q_0, F \rangle$ hetes, ahol



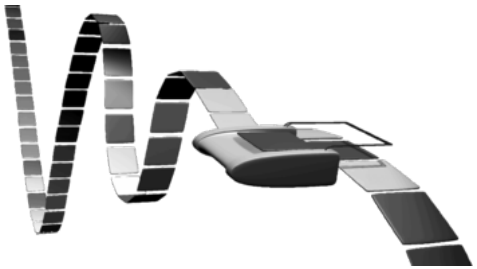
Turing-gép

- A Turing-gép egy $M = \langle Q, \Gamma, b, \Sigma, \delta, q_0, F \rangle$ hetes, ahol
- Q az 'állapotok' nem üres halmaza,



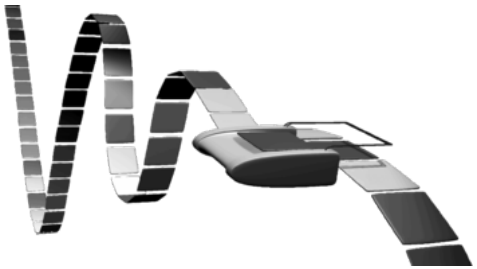
Turing-gép

- A Turing-gép egy $M = \langle Q, \Gamma, b, \Sigma, \delta, q_0, F \rangle$ hetes, ahol
- Q az 'állapotok' nem üres halmaza,
- Γ a 'szalag ábécé' véges, nem üres halmaza,



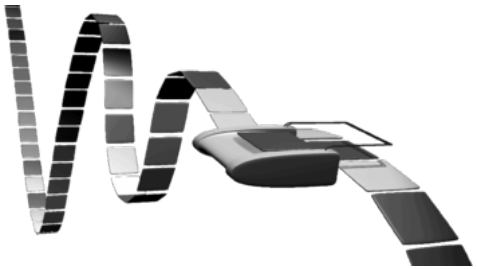
Turing-gép

- A Turing-gép egy $M = \langle Q, \Gamma, b, \Sigma, \delta, q_0, F \rangle$ hetes, ahol
- Q az 'állapotok' nem üres halmaza,
- Γ a 'szalag ábécé' véges, nem üres halmaza,
- $b \in \Gamma$ az 'üres szimbólum' (az egyetlen jel, amiből végtelen sok lehet a szalagon),



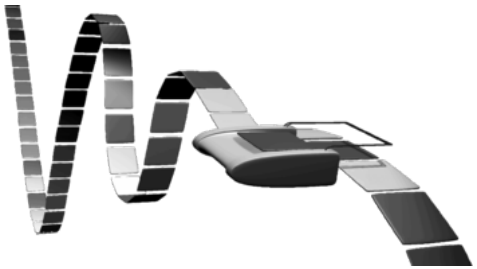
Turing-gép

- A Turing-gép egy $M = \langle Q, \Gamma, b, \Sigma, \delta, q_0, F \rangle$ hetes, ahol
- Q az 'állapotok' nem üres halmaza,
- Γ a 'szalag ábécé' véges, nem üres halmaza,
- $b \in \Gamma$ az 'üres szimbólum' (az egyetlen jel, amiből végtelen sok lehet a szalagon),
- $\Sigma \subseteq \Gamma \setminus \{b\}$ a 'bemeneti jelek' halmaza,



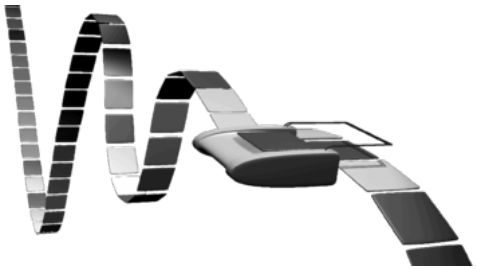
Turing-gép

- A Turing-gép egy $M = \langle Q, \Gamma, b, \Sigma, \delta, q_0, F \rangle$ hetes, ahol
- Q az 'állapotok' nem üres halmaza,
- Γ a 'szalag ábécé' véges, nem üres halmaza,
- $b \in \Gamma$ az 'üres szimbólum' (az egyetlen jel, amiből végtelen sok lehet a szalagon),
- $\Sigma \subseteq \Gamma \setminus \{b\}$ a 'bemeneti jelek' halmaza,
- $q_0 \in Q$ a 'kezdő állapot'



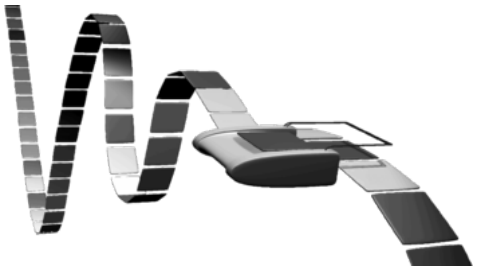
Turing-gép

- A Turing-gép egy $M = \langle Q, \Gamma, b, \Sigma, \delta, q_0, F \rangle$ hetes, ahol
- Q az 'állapotok' nem üres halmaza,
- Γ a 'szalag ábécé' véges, nem üres halmaza,
- $b \in \Gamma$ az 'üres szimbólum' (az egyetlen jel, amiből végtelen sok lehet a szalagon),
- $\Sigma \subseteq \Gamma \setminus \{b\}$ a 'bemeneti jelek' halmaza,
- $q_0 \in Q$ a 'kezdő állapot'
- $F \subseteq Q$ a 'végállapotok' halmaza (ekkor a gép leáll).



Turing-gép

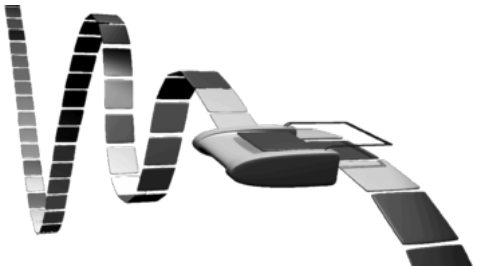
- A Turing-gép egy $M = \langle Q, \Gamma, b, \Sigma, \delta, q_0, F \rangle$ hetes, ahol
- Q az 'állapotok' nem üres halmaza,
- Γ a 'szalag ábécé' véges, nem üres halmaza,



- $b \in \Gamma$ az 'üres szimbólum' (az egyetlen jel, amiből végtelen sok lehet a szalagon),
- $\Sigma \subseteq \Gamma \setminus \{b\}$ a 'bemeneti jelek' halmaza,
- $q_0 \in Q$ a 'kezdő állapot'
- $F \subseteq Q$ a 'végállapotok' halmaza (ekkor a gép leáll).
- $\delta : (Q \setminus F) \times \Gamma \hookrightarrow Q \times \Gamma \times \{L, R\}$ az 'átviteli függvény' (ha nincs értelmezve, a gép leáll), ahol R a szalag jobbra, L balra mozgatást jelenti,

Turing-gép

- A Turing-gép egy $M = \langle Q, \Gamma, b, \Sigma, \delta, q_0, F \rangle$ hetes, ahol
- Q az 'állapotok' nem üres halmaza,
- Γ a 'szalag ábécé' véges, nem üres halmaza,

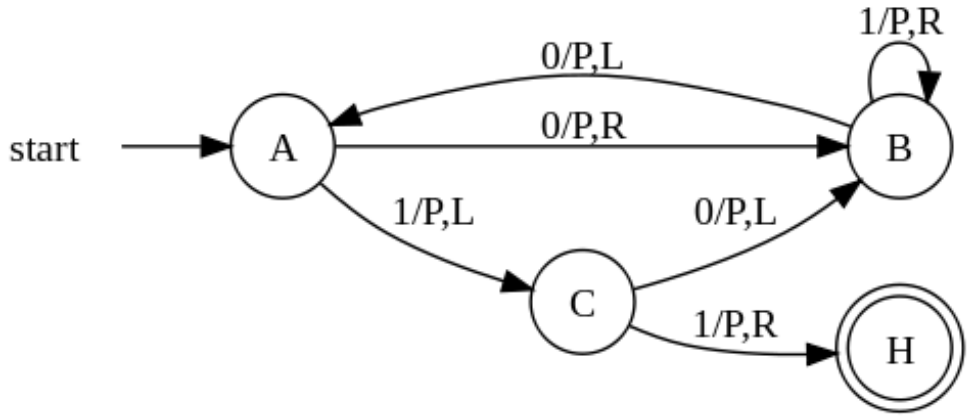


- $b \in \Gamma$ az 'üres szimbólum' (az egyetlen jel, amiből végtelen sok lehet a szalagon),
- $\Sigma \subseteq \Gamma \setminus \{b\}$ a 'bemeneti jelek' halmaza,
- $q_0 \in Q$ a 'kezdő állapot'
- $F \subseteq Q$ a 'végállapotok' halmaza (ekkor a gép leáll).
- $\delta : (Q \setminus F) \times \Gamma \hookrightarrow Q \times \Gamma \times \{L, R\}$ az 'átviteli függvény' (ha nincs értelmezve, a gép leáll), ahol R a szalag jobbra, L balra mozgatást jelenti,

H *Church–Turing-tézis*: minden formalizálható probléma, ami megoldható algoritmussal, az megoldható Turing-géppel is.

Turing gép

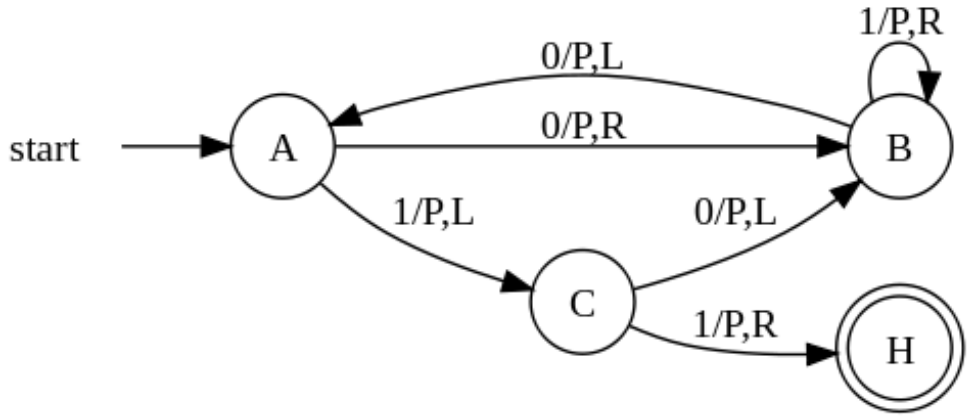
- Dolgos hód (Radó Tibor, 1962, busy beaver) az a Turing-gép, amely adott típusú Turing-gépek közül a legtöbb nem üres jelet írja egy üres szalagra, és véges lépésben leáll.



1	A	0	0	0	0	0	0	0	0	0	0	0
2	B	0	0	0	0	0	0	1	0	0	0	0
3	A	0	0	0	0	1	1	0	0	0	0	0
4	C	0	0	0	1	1	0	0	0	0	0	0
5	B	0	0	1	1	1	0	0	0	0	0	0
6	A	0	1	1	1	1	0	0	0	0	0	0
7	B	0	0	1	1	1	1	1	0	0	0	0
8	B	0	0	0	1	1	1	1	1	0	0	0
9	B	0	0	0	0	1	1	1	1	1	0	0
10	B	0	0	0	0	0	1	1	1	1	1	0
11	B	0	0	0	0	0	0	1	1	1	1	0
12	A	0	0	0	0	1	1	1	1	1	1	0
13	C	0	0	0	1	1	1	1	1	1	0	0
14	H	0	0	0	1	1	1	1	1	1	0	0

Turing gép

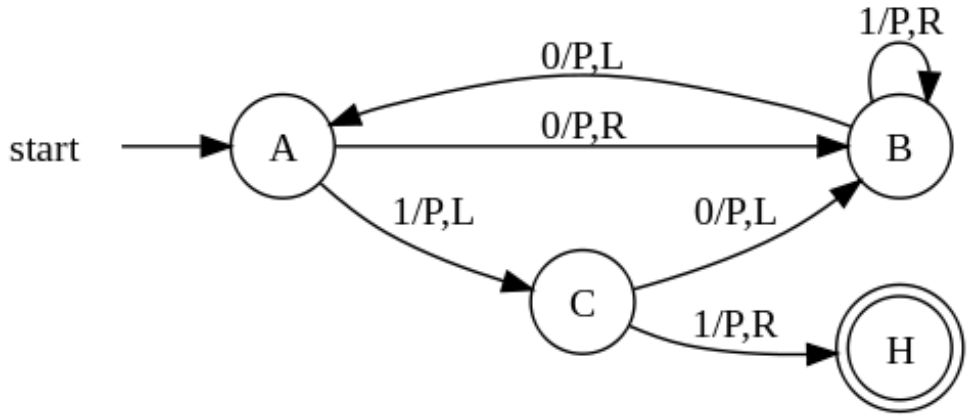
- Dolgos hód (Radó Tibor, 1962, busy beaver) az a Turing-gép, amely adott típusú Turing-gépek közül a legtöbb nem üres jelet írja egy üres szalagra, és véges lépésben leáll.
- $Q = \{A, B, C, HALT\}$



1	A	0	0	0	0	0	0	0	0	0	0	0
2	B	0	0	0	0	0	0	1	0	0	0	0
3	A	0	0	0	0	1	1	0	0	0	0	0
4	C	0	0	0	1	1	0	0	0	0	0	0
5	B	0	0	1	1	1	0	0	0	0	0	0
6	A	0	1	1	1	1	0	0	0	0	0	0
7	B	0	0	1	1	1	1	1	0	0	0	0
8	B	0	0	0	1	1	1	1	1	0	0	0
9	B	0	0	0	0	1	1	1	1	1	0	0
10	B	0	0	0	0	0	1	1	1	1	1	0
11	B	0	0	0	0	0	0	1	1	1	1	0
12	A	0	0	0	0	1	1	1	1	1	1	0
13	C	0	0	0	1	1	1	1	1	1	0	0
14	H	0	0	0	1	1	1	1	1	1	0	0

Turing gép

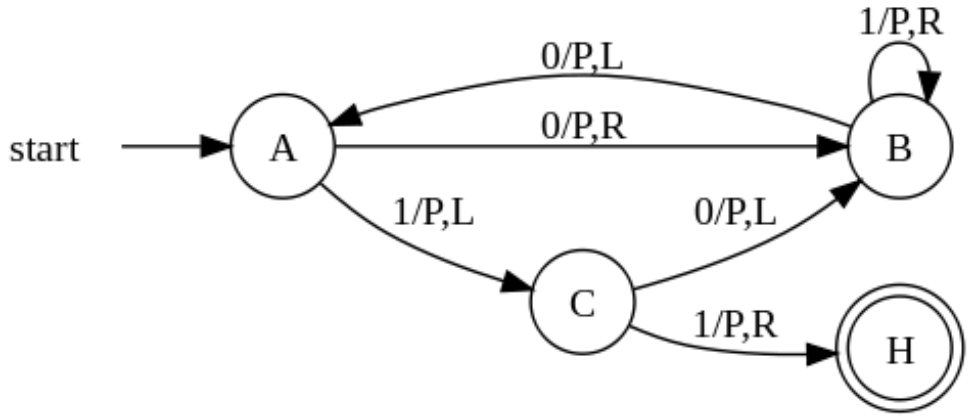
- Dolgos hód (Radó Tibor, 1962, busy beaver) az a Turing-gép, amely adott típusú Turing-gépek közül a legtöbb nem üres jelet írja egy üres szalagra, és véges lépésben leáll.
- $Q = \{A, B, C, HALT\}$
- $\Gamma = \{0, 1\}$



1	A	0 0 0 0 0	0	0 0 0 0 0 0
2	B	0 0 0 0 0	0	1 0 0 0 0 0
3	A	0 0 0 0 1	1	0 0 0 0 0 0
4	C	0 0 0 1 1	0	0 0 0 0 0 0
5	B	0 0 1 1 1	0	0 0 0 0 0 0
6	A	0 1 1 1 1	0	0 0 0 0 0 0
7	B	0 0 1 1 1	1	1 0 0 0 0 0
8	B	0 0 0 1 1	1	1 1 0 0 0 0
9	B	0 0 0 0 1	1	1 1 1 0 0 0
10	B	0 0 0 0 0	1	1 1 1 1 0 0
11	B	0 0 0 0 0	0	1 1 1 1 1 0
12	A	0 0 0 0 1	1	1 1 1 1 0 0
13	C	0 0 0 1 1	1	1 1 1 0 0 0
14	H	0 0 0 1 1	1	1 1 1 0 0 0

Turing gép

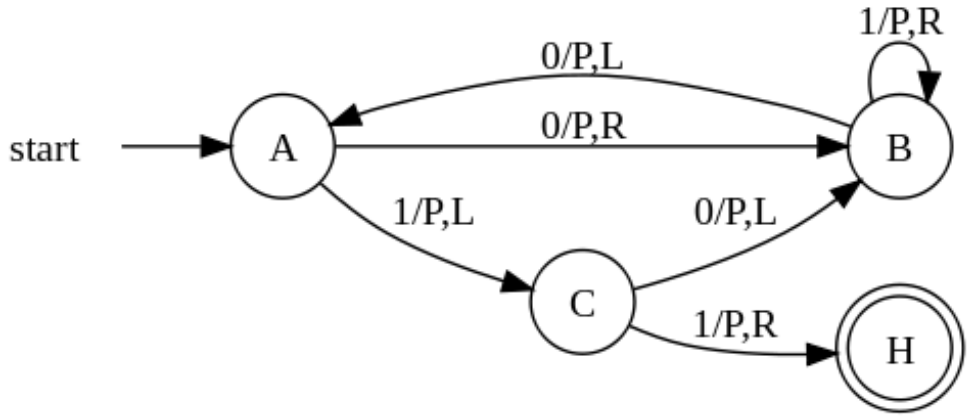
- Dolgos hód (Radó Tibor, 1962, busy beaver) az a Turing-gép, amely adott típusú Turing-gépek közül a legtöbb nem üres jelet írja egy üres szalagra, és véges lépésben leáll.
- $Q = \{A, B, C, HALT\}$
- $\Gamma = \{0, 1\}$
- $b = 0$ (az üres jel)



1	A	0	0	0	0	0	0	0	0	0	0	0
2	B	0	0	0	0	0	0	1	0	0	0	0
3	A	0	0	0	0	1	1	0	0	0	0	0
4	C	0	0	0	1	1	0	0	0	0	0	0
5	B	0	0	1	1	1	0	0	0	0	0	0
6	A	0	1	1	1	1	0	0	0	0	0	0
7	B	0	0	1	1	1	1	1	0	0	0	0
8	B	0	0	0	1	1	1	1	1	0	0	0
9	B	0	0	0	0	1	1	1	1	1	0	0
10	B	0	0	0	0	0	1	1	1	1	1	0
11	B	0	0	0	0	0	0	1	1	1	1	0
12	A	0	0	0	0	1	1	1	1	1	1	0
13	C	0	0	0	1	1	1	1	1	1	0	0
14	H	0	0	0	1	1	1	1	1	1	0	0

Turing gép

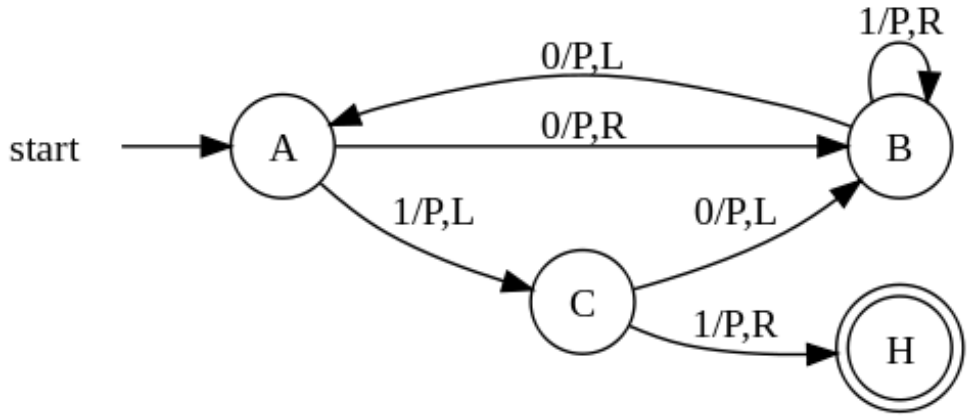
- Dolgos hód (Radó Tibor, 1962, busy beaver) az a Turing-gép, amely adott típusú Turing-gépek közül a legtöbb nem üres jelet írja egy üres szalagra, és véges lépésben leáll.
- $Q = \{A, B, C, HALT\}$
- $\Gamma = \{0, 1\}$
- $b = 0$ (az üres jel)
- $\Sigma = \{1\}$



1	A	0	0	0	0	0	0	0	0	0	0	0
2	B	0	0	0	0	0	0	1	0	0	0	0
3	A	0	0	0	0	1	1	0	0	0	0	0
4	C	0	0	0	1	1	0	0	0	0	0	0
5	B	0	0	1	1	1	0	0	0	0	0	0
6	A	0	1	1	1	1	0	0	0	0	0	0
7	B	0	0	1	1	1	1	1	0	0	0	0
8	B	0	0	0	1	1	1	1	1	0	0	0
9	B	0	0	0	0	1	1	1	1	1	0	0
10	B	0	0	0	0	0	1	1	1	1	1	0
11	B	0	0	0	0	0	0	1	1	1	1	0
12	A	0	0	0	0	1	1	1	1	1	1	0
13	C	0	0	0	1	1	1	1	1	1	0	0
14	H	0	0	0	1	1	1	1	1	1	0	0

Turing gép

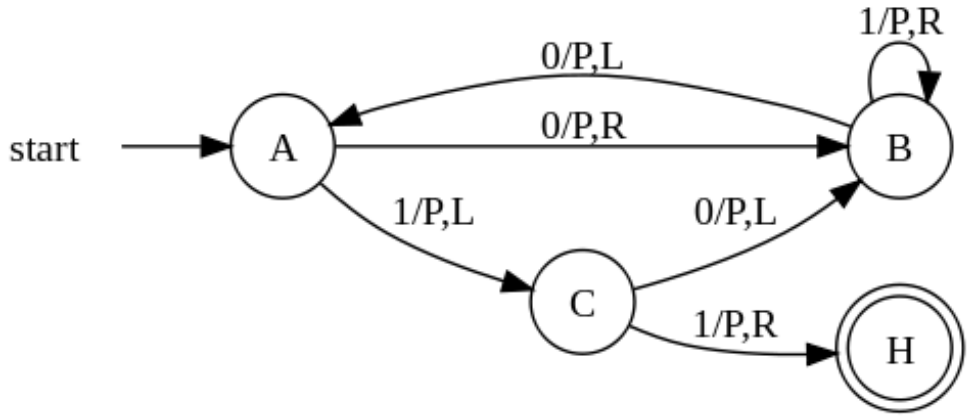
- Dolgos hód (Radó Tibor, 1962, busy beaver) az a Turing-gép, amely adott típusú Turing-gépek közül a legtöbb nem üres jelet írja egy üres szalagra, és véges lépésben leáll.
- $Q = \{A, B, C, HALT\}$
- $\Gamma = \{0, 1\}$
- $b = 0$ (az üres jel)
- $\Sigma = \{1\}$
- $q_0 = A$ (kezdő állapot)



1	A	0	0	0	0	0	0	0	0	0	0	0	0
2	B	0	0	0	0	0	0	1	0	0	0	0	0
3	A	0	0	0	0	1	1	0	0	0	0	0	0
4	C	0	0	0	1	1	0	0	0	0	0	0	0
5	B	0	0	1	1	1	0	0	0	0	0	0	0
6	A	0	1	1	1	1	0	0	0	0	0	0	0
7	B	0	0	1	1	1	1	1	0	0	0	0	0
8	B	0	0	0	1	1	1	1	1	0	0	0	0
9	B	0	0	0	0	1	1	1	1	1	0	0	0
10	B	0	0	0	0	0	1	1	1	1	1	0	0
11	B	0	0	0	0	0	0	1	1	1	1	1	0
12	A	0	0	0	0	1	1	1	1	1	1	0	0
13	C	0	0	0	1	1	1	1	1	1	0	0	0
14	H	0	0	0	1	1	1	1	1	1	0	0	0

Turing gép

- Dolgos hód (Radó Tibor, 1962, busy beaver) az a Turing-gép, amely adott típusú Turing-gépek közül a legtöbb nem üres jelet írja egy üres szalagra, és véges lépésben leáll.
- $Q = \{A, B, C, HALT\}$
- $\Gamma = \{0, 1\}$
- $b = 0$ (az üres jel)
- $\Sigma = \{1\}$
- $q_0 = A$ (kezdő állapot)
- $F = \{HALT\}$

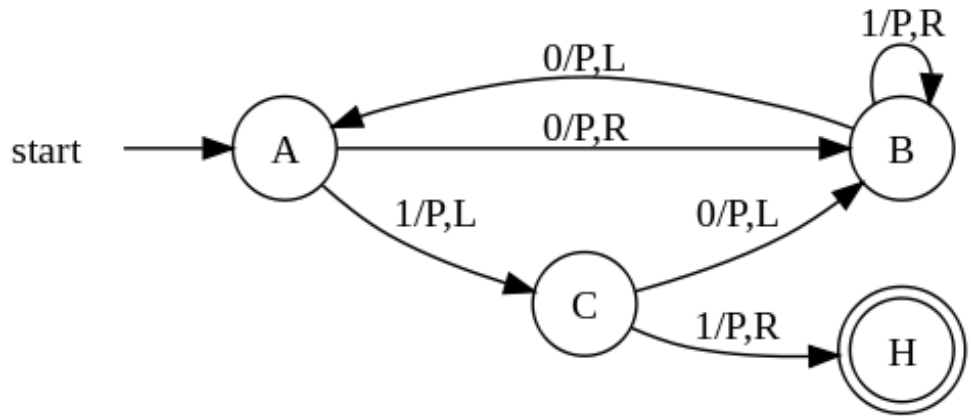


1	A	0	0	0	0	0	0	0	0	0	0	0	0	0
2	B	0	0	0	0	0	0	1	0	0	0	0	0	0
3	A	0	0	0	0	1	1	0	0	0	0	0	0	0
4	C	0	0	0	1	1	0	0	0	0	0	0	0	0
5	B	0	0	1	1	1	0	0	0	0	0	0	0	0
6	A	0	1	1	1	1	0	0	0	0	0	0	0	0
7	B	0	0	1	1	1	1	1	0	0	0	0	0	0
8	B	0	0	0	1	1	1	1	1	0	0	0	0	0
9	B	0	0	0	0	1	1	1	1	1	0	0	0	0
10	B	0	0	0	0	0	1	1	1	1	1	0	0	0
11	B	0	0	0	0	0	0	1	1	1	1	1	0	0
12	A	0	0	0	0	1	1	1	1	1	1	0	0	0
13	C	0	0	0	1	1	1	1	1	1	0	0	0	0
14	H	0	0	0	1	1	1	1	1	1	0	0	0	0

Turing gép

- Dolgos hód (Radó Tibor, 1962, busy beaver) az a Turing-gép, amely adott típusú Turing-gépek közül a legtöbb nem üres jelet írja egy üres szalagra, és véges lépésben leáll.
- $Q = \{A, B, C, HALT\}$
- $\Gamma = \{0, 1\}$
- $b = 0$ (az üres jel)
- $\Sigma = \{1\}$
- $q_0 = A$ (kezdő állapot)
- $F = \{HALT\}$
- δ táblázata:

	A	B	C
0	1RB	1LA	1LB
1	1LC	1RB	1RH



1	A	0	0	0	0	0	0	0	0	0	0	0
2	B	0	0	0	0	0	0	1	0	0	0	0
3	A	0	0	0	0	1	1	0	0	0	0	0
4	C	0	0	0	1	1	0	0	0	0	0	0
5	B	0	0	1	1	1	0	0	0	0	0	0
6	A	0	1	1	1	1	0	0	0	0	0	0
7	B	0	0	1	1	1	1	1	0	0	0	0
8	B	0	0	0	1	1	1	1	1	0	0	0
9	B	0	0	0	0	1	1	1	1	1	0	0
10	B	0	0	0	0	0	1	1	1	1	1	0
11	B	0	0	0	0	0	0	1	1	1	1	0
12	A	0	0	0	0	1	1	1	1	1	1	0
13	C	0	0	0	1	1	1	1	1	1	0	0
14	H	0	0	0	1	1	1	1	1	0	0	0

BIOS (Basic Input/Output System)

- Mi van az operációs rendszer előtt? Honnan tudja a gép, hogy honnan töltse be az oprendszert, hogy tudja használni a monitort és a billentyűzetet?

BIOS (Basic Input/Output System)

- Mi van az operációs rendszer előtt? Honnan tudja a gép, hogy honnan töltse be az oprendszert, hogy tudja használni a monitort és a billentyűzetet?
- Az első dolog ami a számítógép bekapcsolása után történik, hogy a BIOS elkezdi a működését

BIOS (Basic Input/Output System)

- Mi van az operációs rendszer előtt? Honnan tudja a gép, hogy honnan töltse be az oprendszert, hogy tudja használni a monitort és a billentyűzetet?
- Az első dolog ami a számítógép bekapcsolása után történik, hogy a BIOS elkezdi a működését
- Ez egy minimális rendszer az alaplapba építve, melynek feladata a számítógép kezdeti állapotba állítása

BIOS (Basic Input/Output System)

- Mi van az operációs rendszer előtt? Honnan tudja a gép, hogy honnan töltse be az oprendszert, hogy tudja használni a monitort és a billentyűzetet?
- Az első dolog ami a számítógép bekapcsolása után történik, hogy a BIOS elkezdi a működését
- Ez egy minimális rendszer az alaplapba építve, melynek feladata a számítógép kezdeti állapotba állítása
- A BIOS-ban tárolva van driver (szoftver mely a számítógép valamely fizikai alkatrészének a működését segíti) a csatlakoztatott billentyűzet és monitor használatára

BIOS (Basic Input/Output System)

- Mi van az operációs rendszer előtt? Honnan tudja a gép, hogy honnan töltse be az oprendszert, hogy tudja használni a monitort és a billentyűzetet?
- Az első dolog ami a számítógép bekapcsolása után történik, hogy a BIOS elkezdi a működését
- Ez egy minimális rendszer az alaplapba építve, melynek feladata a számítógép kezdeti állapotba állítása
- A BIOS-ban tárolva van driver (szoftver mely a számítógép valamely fizikai alkatrészének a működését segíti) a csatlakoztatott billentyűzet és monitor használatára
- Amikor a BIOS elindul a háttértárról még semmilyen tudása nincs

BIOS (Basic Input/Output System)

- Mi van az operációs rendszer előtt? Honnan tudja a gép, hogy honnan töltse be az oprendszert, hogy tudja használni a monitort és a billentyűzetet?
- Az első dolog ami a számítógép bekapcsolása után történik, hogy a BIOS elkezdi a működését
- Ez egy minimális rendszer az alaplapba építve, melynek feladata a számítógép kezdeti állapotba állítása
- A BIOS-ban tárolva van driver (szoftver mely a számítógép valamely fizikai alkatrészének a működését segíti) a csatlakoztatott billentyűzet és monitor használatára
- Amikor a BIOS elindul a háttértárról még semmilyen tudása nincs
- A BIOS megkeresi az első (legnagyobb prioritású) csatlakoztatott háttértárat és elkezdi az operációs rendszer betöltését

MBR (Master Boot Record)

- Az operációs rendszer betöltésének első lépése, hogy a BIOS beolvassa a megtalált háttértár első 512 byte-ját, melyen a MBR van tárolva

MBR (Master Boot Record)

- Az operációs rendszer betöltésének első lépése, hogy a BIOS beolvassa a megtalált háttértár első 512 byte-ját, melyen a MBR van tárolva
- Az MBR első része egy rövid programkód (bootstrap code), mely a számítógép indításának részleteit írja le (boots-trap=cipőhúzó, csizmahúzó)

MBR (Master Boot Record)

- Az operációs rendszer betöltésének első lépése, hogy a BIOS beolvassa a megtalált háttértár első 512 byte-ját, melyen a MBR van tárolva
- Az MBR első része egy rövid programkód (bootstrap code), mely a számítógép indításának részleteit írja le (boots-trap=cipőhúzó, csizmahúzó)
- A következő rész a partíciós tábla

MBR (Master Boot Record)

- Az operációs rendszer betöltésének első lépése, hogy a BIOS beolvassa a megtalált háttértár első 512 byte-ját, melyen a MBR van tárolva
- Az MBR első része egy rövid programkód (bootstrap code), mely a számítógép indításának részleteit írja le (boots-trap=cipőhúzó, csizmahúzó)
- A következő rész a partíciós tábla
- A harmadik és utolsó része az MBR-nak, egy mágikus számnak nevezett szám, mely minden PC-nél ugyanaz (0xAA55 = 0b1010101001010101, ahol 0xAA az utolsó bájt!), ezzel ellenőrzi a BIOS, hogy valódi MBR-t talált-e a háttértár elején, (ha nem, akkor nem indítja az operációs rendszert)

MBR (Master Boot Record)

- Az operációs rendszer betöltésének első lépése, hogy a BIOS beolvassa a megtalált háttértár első 512 byte-ját, melyen a MBR van tárolva
- Az MBR első része egy rövid programkód (bootstrap code), mely a számítógép indításának részleteit írja le (boots-trap=cipőhúzó, csizmahúzó)
- A következő rész a partíciós tábla
- A harmadik és utolsó része az MBR-nak, egy mágikus számnak nevezett szám, mely minden PC-nél ugyanaz (0xAA55 = 0b1010101001010101, ahol 0xAA az utolsó bájt!), ezzel ellenőrzi a BIOS, hogy valódi MBR-t talált-e a háttértár elején, (ha nem, akkor nem indítja az operációs rendszert)
- Egészen eddig a pontig, a számítógép indulása operációs rendszertől független

Háttértárak

- Az MBR után egy vagy több partíció van

Háttértárak

- Az MBR után egy vagy több partíció van
- Elsődleges partícióból egy háttértáron maximum 4 lehet

Háttértárak

- Az MBR után egy vagy több partíció van
- Elsődleges partícióból egy háttértáron maximum 4 lehet
- Operációs rendszert elsődleges partícióra érdemes installálni (Windowst például csak erre lehet)

Háttértár kiterjesztett partíciója

- Elsődleges partíciónak számít a max 4 szabály tekintetében, azaz vagy 4 elsődleges vagy három elsődleges és egy kiterjesztett partíció lehet a tárolón.

Háttértár kiterjesztett partíciója

- Elsődleges partíciónak számít a max 4 szabály tekintetében, azaz vagy 4 elsődleges vagy három elsődleges és egy kiterjesztett partíció lehet a tárolón.
- Tetszőleges számú logikai partíciót tartalmaz logikai partíciót tárolhat, így lehet 4 fölé növelni a lehetséges partíciók számát

Háttértár kiterjesztett partíciója

- Elsődleges partíciónak számít a max 4 szabály tekintetében, azaz vagy 4 elsődleges vagy három elsődleges és egy kiterjesztett partíció lehet a tárolón.
- Tetszőleges számú logikai partíciót tartalmaz logikai partíciót tárolhat, így lehet 4 fölé növelni a lehetséges partíciók számát
- Csak a háttártár végén helyezkedhet el, azaz utána elsődleges partíció nem jöhet

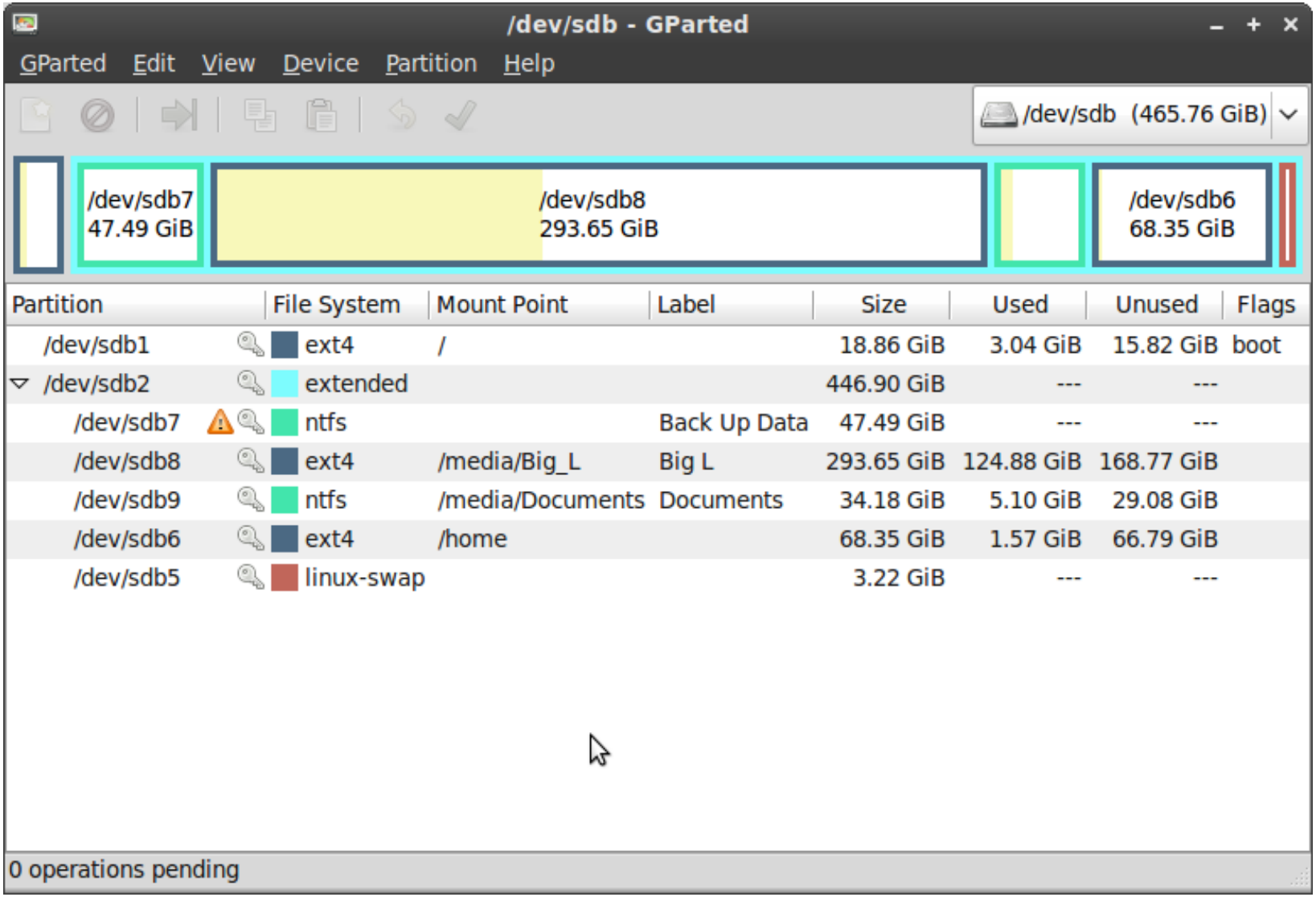
Háttértár kiterjesztett partíciója

- Elsődleges partíciónak számít a max 4 szabály tekintetében, azaz vagy 4 elsődleges vagy három elsődleges és egy kiterjesztett partíció lehet a tárolón.
- Tetszőleges számú logikai partíciót tartalmaz logikai partíciót tárolhat, így lehet 4 fölé növelni a lehetséges partíciók számát
- Csak a háttártár végén helyezkedhet el, azaz utána elsődleges partíció nem jöhet
- A windowsnak szokása telepítéskor létrehozni egy recovery partíciót, mely az operációs rendszer partíciója előtt helyezkedik el, ha elromlana az operációs rendszer, akkor ennek segítségével próbálja megjavítani magát

Háttértár kiterjesztett partíciója

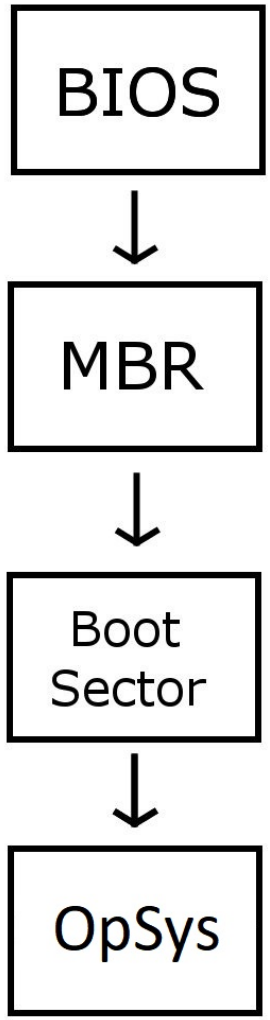
- Elsődleges partíciónak számít a max 4 szabály tekintetében, azaz vagy 4 elsődleges vagy három elsődleges és egy kiterjesztett partíció lehet a tárolón.
- Tetszőleges számú logikai partíciót tartalmaz logikai partíciót tárolhat, így lehet 4 fölé növelni a lehetséges partíciók számát
- Csak a háttártár végén helyezkedhet el, azaz utána elsődleges partíció nem jöhet
- A windowsnak szokása telepítéskor létrehozni egy recovery partíciót, mely az operációs rendszer partíciója előtt helyezkedik el, ha elromlana az operációs rendszer, akkor ennek segítségével próbálja megjavítani magát
- A linux több (általában 4) partíciót használ, egyikőjük a virtuális memória partíciója. Ide másolódik a valódi memória épp nem használt része (swapping, paging).

Példa grafikus partícionáló szoftverre



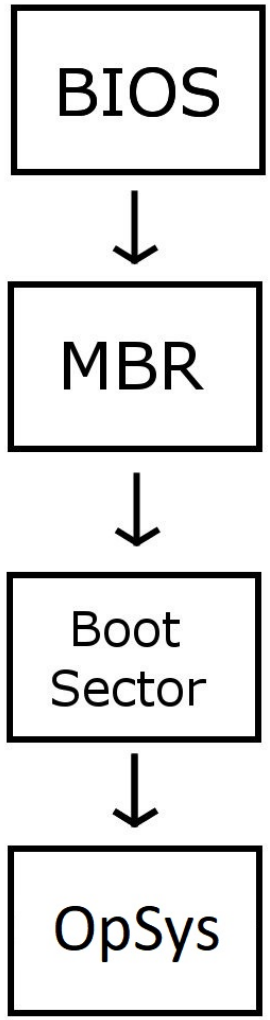
Boot Sector

- Minden elsődleges partíció elején egy Boot Sector található, ennek a pozícióját mondja meg az MBR és ez kezdi el az adott operációs rendszer indítását



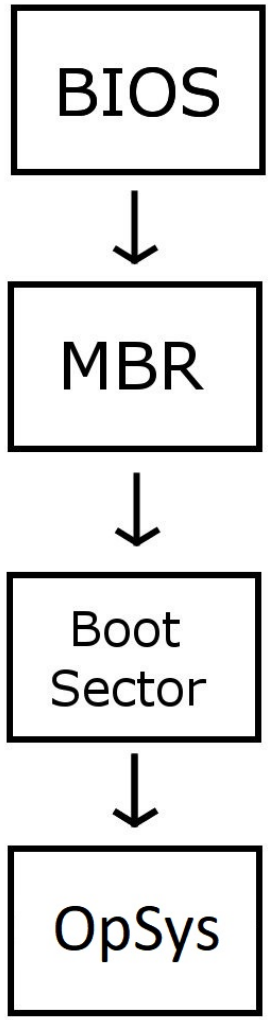
Boot Sector

- Minden elsődleges partíció elején egy Boot Sector található, ennek a pozícióját mondja meg az MBR és ez kezdi el az adott operációs rendszer indítását
- Hasonlóan az MBR-hoz ez is egy 512 byte-os rész, mely az operációs rendszer indításának módját írja le, valamint tartalmazza a mágikus számot, mint az MBR



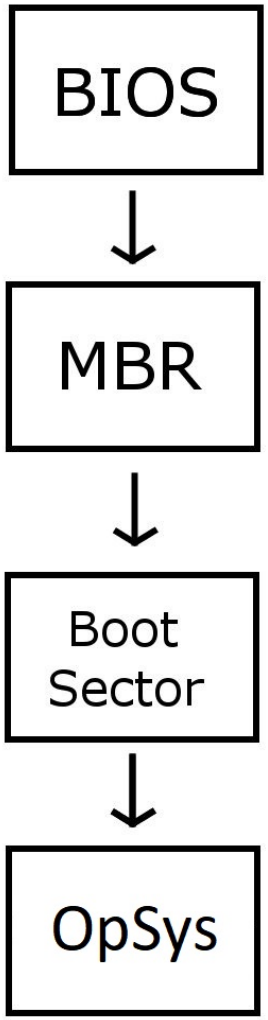
Boot Sector

- Minden elsődleges partíció elején egy Boot Sector található, ennek a pozícióját mondja meg az MBR és ez kezdi el az adott operációs rendszer indítását
- Hasonlóan az MBR-hoz ez is egy 512 byte-os rész, mely az operációs rendszer indításának módját írja le, valamint tartalmazza a mágikus számot, mint az MBR
- Linux rendszereken a Boot Sector valójában üres, és az operációs rendszer a betöltését máshogy végzi, ezáltal lehetséges logikai partícióra telepíteni linuxot



Boot Sector

- Minden elsődleges partíció elején egy Boot Sector található, ennek a pozícióját mondja meg az MBR és ez kezdi el az adott operációs rendszer indítását
- Hasonlóan az MBR-hoz ez is egy 512 byte-os rész, mely az operációs rendszer indításának módját írja le, valamint tartalmazza a mágikus számot, mint az MBR
- Linux rendszereken a Boot Sector valójában üres, és az operációs rendszer a betöltését máshogy végzi, ezáltal lehetséges logikai partícióra telepíteni linuxot
- Amikor több operációs rendszer van egy háttértáron és az MBR-ban megfelelő instrukciók vannak, lehetséges az operációs rendszerek betöltése előtt kiválasztani, hogy melyiket szeretnénk indítani



Fájlrendszer

Oprendszer	WINDOWS	LINUX	MAC	Cserélhető háttértárak
Fájlrendszer	NTFS	ext4	HFS+	FAT32 vagy NTFS

Az operációs rendszerek feladatai

- Operációs rendszer (OS, operating system): alapprogram, mely

Az operációs rendszerek feladatai

- Operációs rendszer (OS, operating system): alapprogram, mely
  - közvetlenül kezeli a hardvert (memóriát, perifériákat,...),

Az operációs rendszerek feladatai

- Operációs rendszer (OS, operating system): alapprogram, mely
  - közvetlenül kezeli a hardvert (memóriát, perifériákat,...),
  - egységes környezetet biztosít a gépen futó alkalmazásoknak,

Az operációs rendszerek feladatai

- Operációs rendszer (OS, operating system): alapprogram, mely
  - közvetlenül kezeli a hardvert (memóriát, perifériákat,...),
  - egységes környezetet biztosít a gépen futó alkalmazásoknak,
  - szervezi azok futását, (osztja a futási időt, a memóriát...)

Az operációs rendszerek feladatai

- Operációs rendszer (OS, operating system): alapprogram, mely
  - közvetlenül kezeli a hardvert (memóriát, perifériákat,...),
  - egységes környezetet biztosít a gépen futó alkalmazásoknak,
  - szervezi azok futását, (osztja a futási időt, a memóriát...)
  - gondoskodik a hibakezelésről,

Az operációs rendszerek feladatai

- Operációs rendszer (OS, operating system): alapprogram, mely
  - közvetlenül kezeli a hardvert (memóriát, perifériákat,...),
  - egységes környezetet biztosít a gépen futó alkalmazásoknak,
  - szervezi azok futását, (osztja a futási időt, a memóriát...)
  - gondoskodik a hibakezelésről,
  - kezeli az állományokat,

Az operációs rendszerek feladatai

- Operációs rendszer (OS, operating system): alapprogram, mely
  - közvetlenül kezeli a hardvert (memóriát, perifériákat,...),
  - egységes környezetet biztosít a gépen futó alkalmazásoknak,
  - szervezi azok futását, (osztja a futási időt, a memóriát...)
  - gondoskodik a hibakezelésről,
  - kezeli az állományokat,
  - gondoskodik a gép és adatainak védelméről,

Az operációs rendszerek feladatai

- Operációs rendszer (OS, operating system): alapprogram, mely
  - közvetlenül kezeli a hardvert (memóriát, perifériákat,...),
  - egységes környezetet biztosít a gépen futó alkalmazásoknak,
  - szervezi azok futását, (osztja a futási időt, a memóriát...)
  - gondoskodik a hibakezelésről,
  - kezeli az állományokat,
  - gondoskodik a gép és adatainak védelméről,
  - a történéseket naplózza...

Az operációs rendszerek feladatai

- Operációs rendszer (OS, operating system): alapprogram, mely
  - közvetlenül kezeli a hardvert (memóriát, perifériákat,...),
  - egységes környezetet biztosít a gépen futó alkalmazásoknak,
  - szervezi azok futását, (osztja a futási időt, a memóriát...)
  - gondoskodik a hibakezelésről,
  - kezeli az állományokat,
  - gondoskodik a gép és adatainak védelméről,
  - a történéseket naplózza...
- Az OS a rendszerprogramok közé tartozik.

Az operációs rendszerek feladatai

- Operációs rendszer (OS, operating system): alapprogram, mely
  - közvetlenül kezeli a hardvert (memóriát, perifériákat,...),
  - egységes környezetet biztosít a gépen futó alkalmazásoknak,
  - szervezi azok futását, (osztja a futási időt, a memóriát...)
  - gondoskodik a hibakezelésről,
  - kezeli az állományokat,
  - gondoskodik a gép és adatainak védelméről,
  - a történéseket naplózza...
- Az OS a rendszerprogramok közé tartozik.
- Rendszerprogramok még a segédprogramok (utility), melyek konfigurálják, analizálják, optimalizálják, karban tartják a számítógépet. Pl. antivírus, archiváló, backup, adattömörítő, adatszinkronizáló, titkosító, verziókövető programok (revision control), diszk kezelő (elemző, ellenőrző, tisztító, defregmentáló...), állománykezelő (törlés, mozgatás, másolás...), hálózati programok, rendszermonitor,...

Operációs rendszerek típusai

- egy felhasználós, több felhasználós (single-, multi-user)

Operációs rendszerek típusai

- egy felhasználós, több felhasználós (single-, multi-user)
- egy feladatos, több feladatos (single-, multi-tasking)

Operációs rendszerek típusai

- egy felhasználós, több felhasználós (single-, multi-user)
- egy feladatos, több feladatos (single-, multi-tasking)
- elosztott (több gép egynek tűnik),

Operációs rendszerek típusai

- egy felhasználós, több felhasználós (single-, multi-user)
- egy feladatos, több feladatos (single-, multi-tasking)
- elosztott (több gép egynek tűnik),
- beágyazott (kis gépekbe, korlátozott erőforrásokkal)

Operációs rendszerek típusai

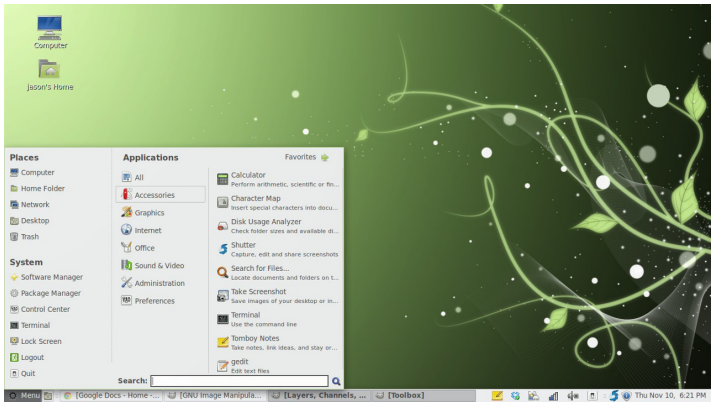
- egy felhasználós, több felhasználós (single-, multi-user)
- egy feladatos, több feladatos (single-, multi-tasking)
- elosztott (több gép egynek tűnik),
- beágyazott (kis gépekbe, korlátozott erőforrásokkal)
- feladata szerint: személyi, szerver,. . .

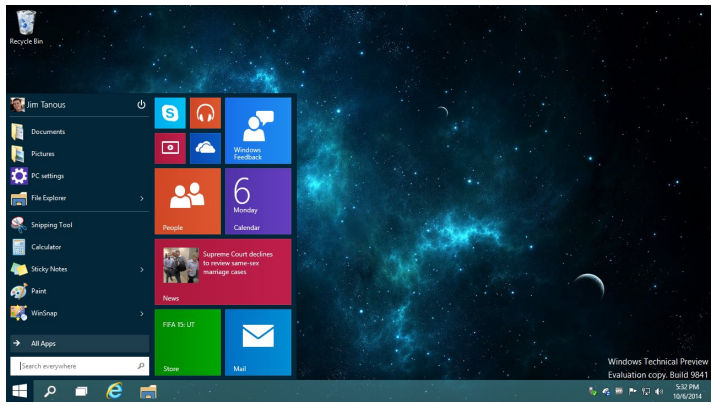
Operációs rendszerek típusai

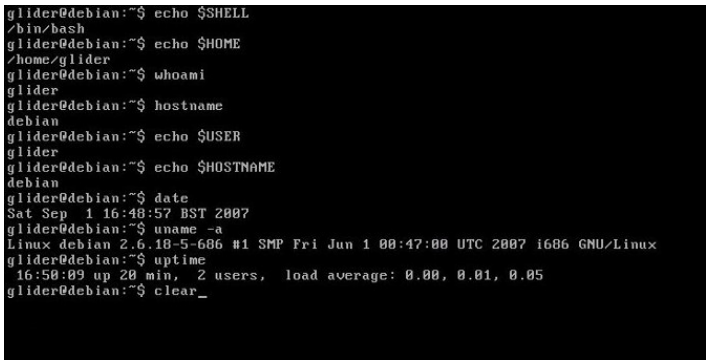
- egy felhasználós, több felhasználós (single-, multi-user)
- egy feladatos, több feladatos (single-, multi-tasking)
- elosztott (több gép egynek tűnik),
- beágyazott (kis gépekbe, korlátozott erőforrásokkal)
- feladata szerint: személyi, szerver,. . .
- a címzésre használt szóhossz szerint 32- vagy 64 bites (maguk a processzorok is vagy 32 vagy 64 bitesek, azaz leegyszerűsítve 32 biten tárolt vagy 64 biten tárolt számokkal számolnak)

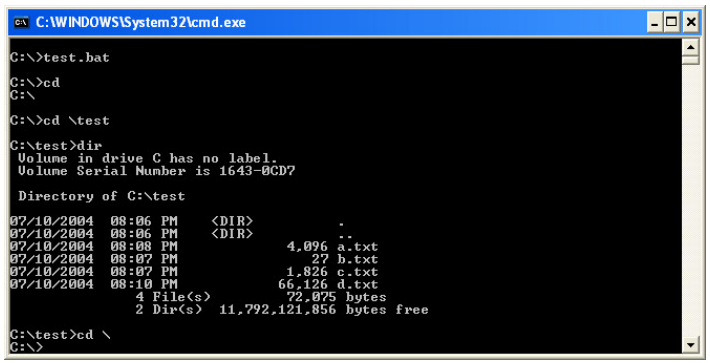
Operációs rendszerek két fontos része

- Kernel: a hardver feletti kontroll alapszintjét biztosítja, szervezi az erőforrásokhoz való hozzáférést a programok közt.



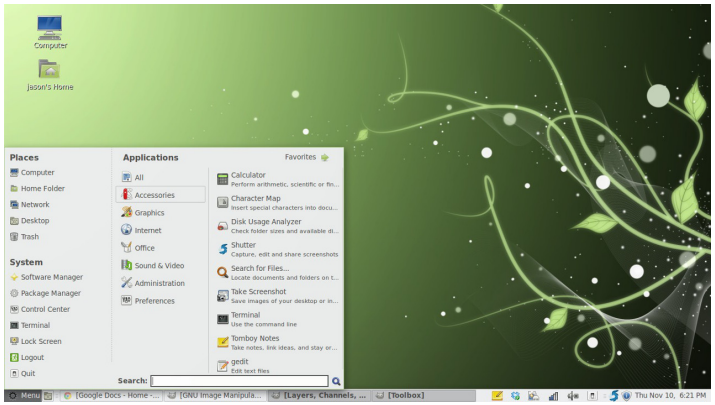


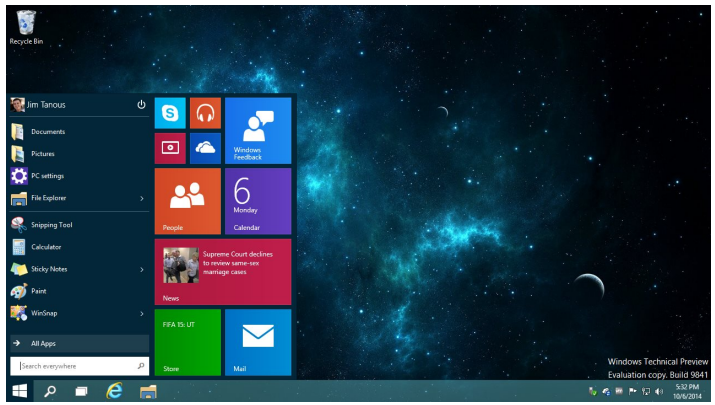




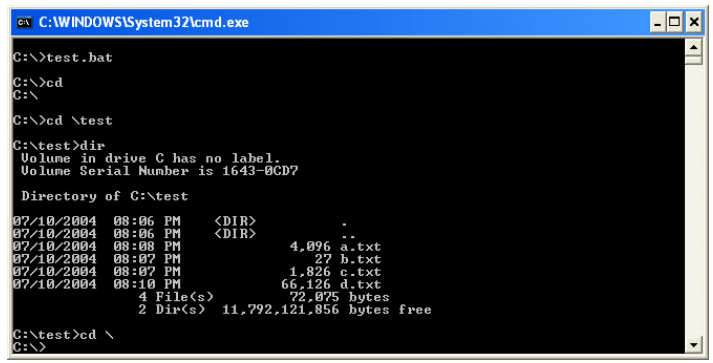
Operációs rendszerek két fontos része

- Kernel: a hardver feletti kontroll alapszintjét biztosítja, szervezi az erőforrásokhoz való hozzáférést a programok közt.
- Shell (burok, héj): a felhasználói felület a rendszerhez. Lehet karakteres, grafikus. (A Linuxban jól elkülönül a kerneltől, a Windowsban nem.)

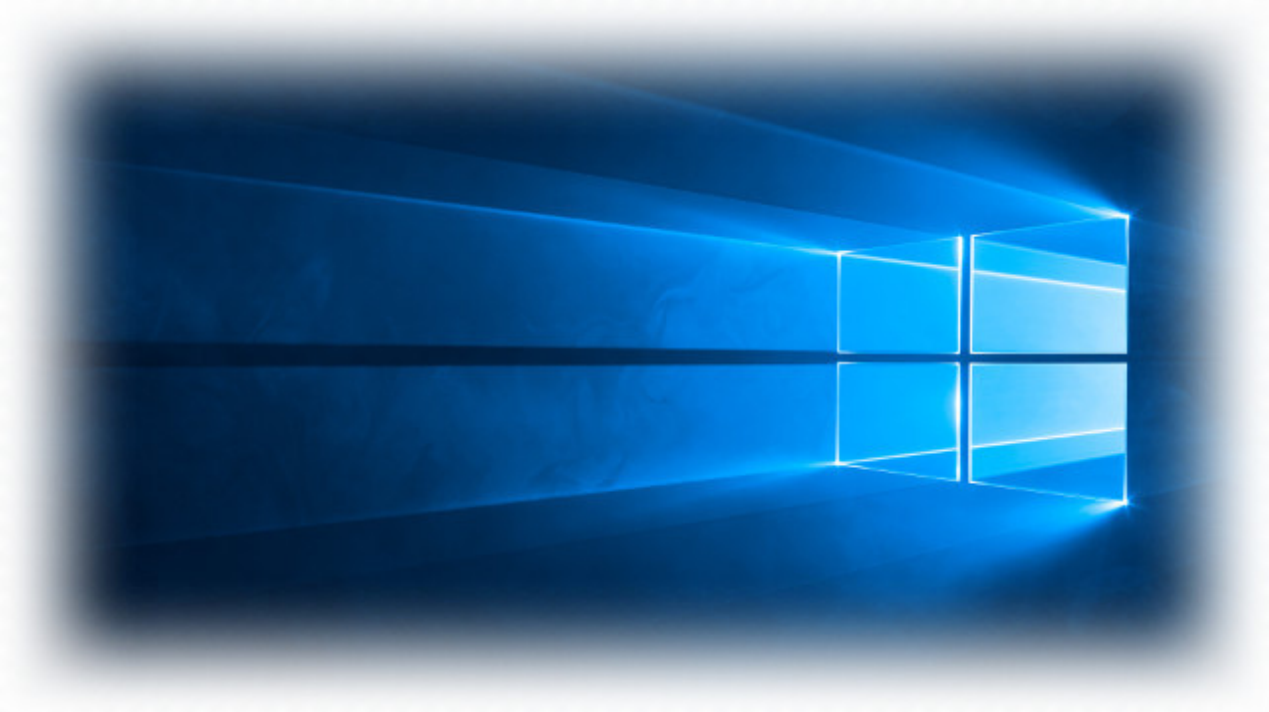




```
glider@debian:~$ echo $SHELL
/bin/bash
glider@debian:~$ echo $HOME
/home/glider
glider@debian:~$ whoami
glider
glider@debian:~$ hostname
debian
glider@debian:~$ echo $USER
glider
glider@debian:~$ echo $HOSTNAME
debian
glider@debian:~$ date
Sat Sep  1 16:48:57 BST 2007
glider@debian:~$ uname -a
Linux debian 2.6.18-5-686 #1 SMP Fri Jun 1 00:47:00 UTC 2007 i686 GNU/Linux
glider@debian:~$ uptime
 16:50:09 up 20 min,  2 users,  load average: 0.00, 0.01, 0.05
glider@debian:~$ clear_
```

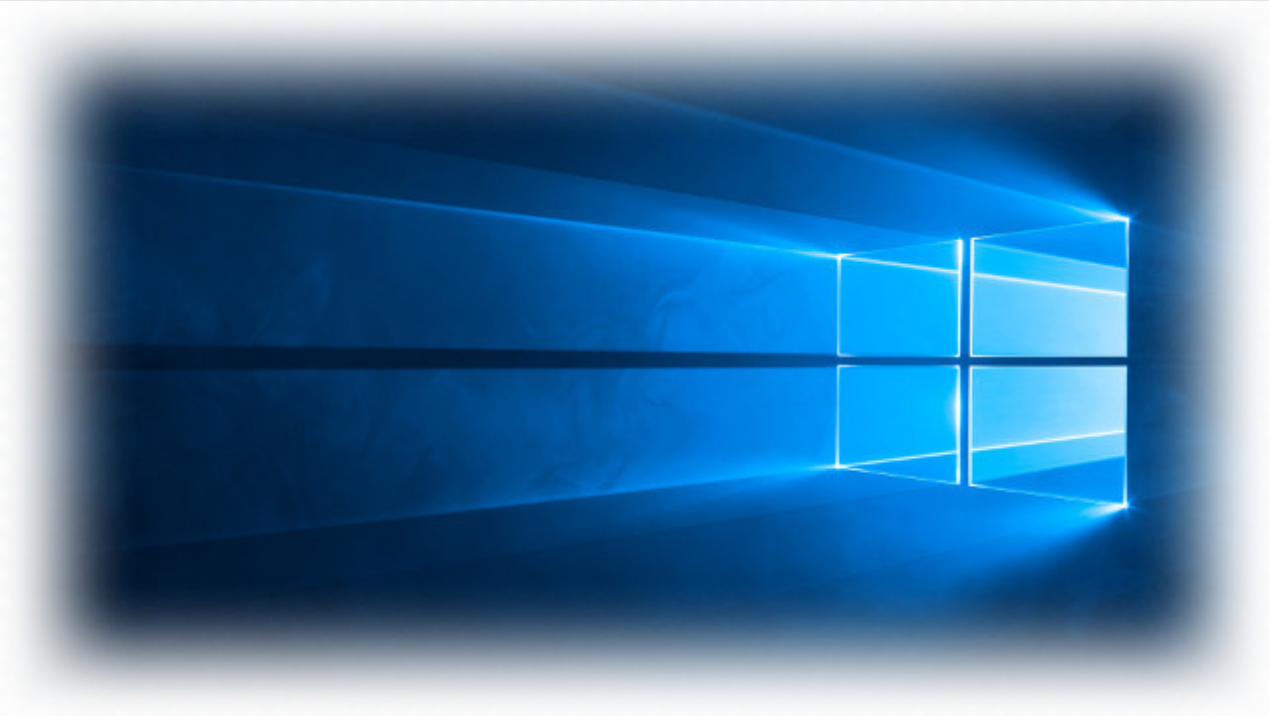


Windows összefoglaló



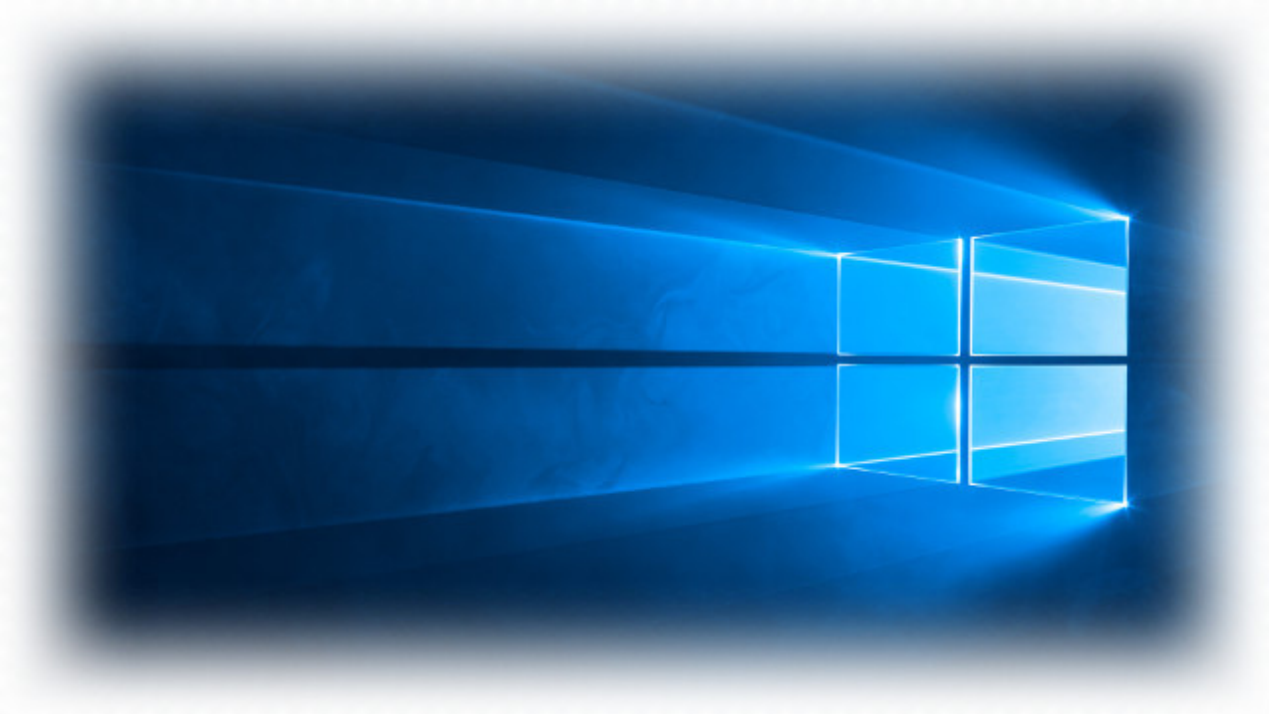
- Használt fájlrendszer: NTFS

Windows összefoglaló



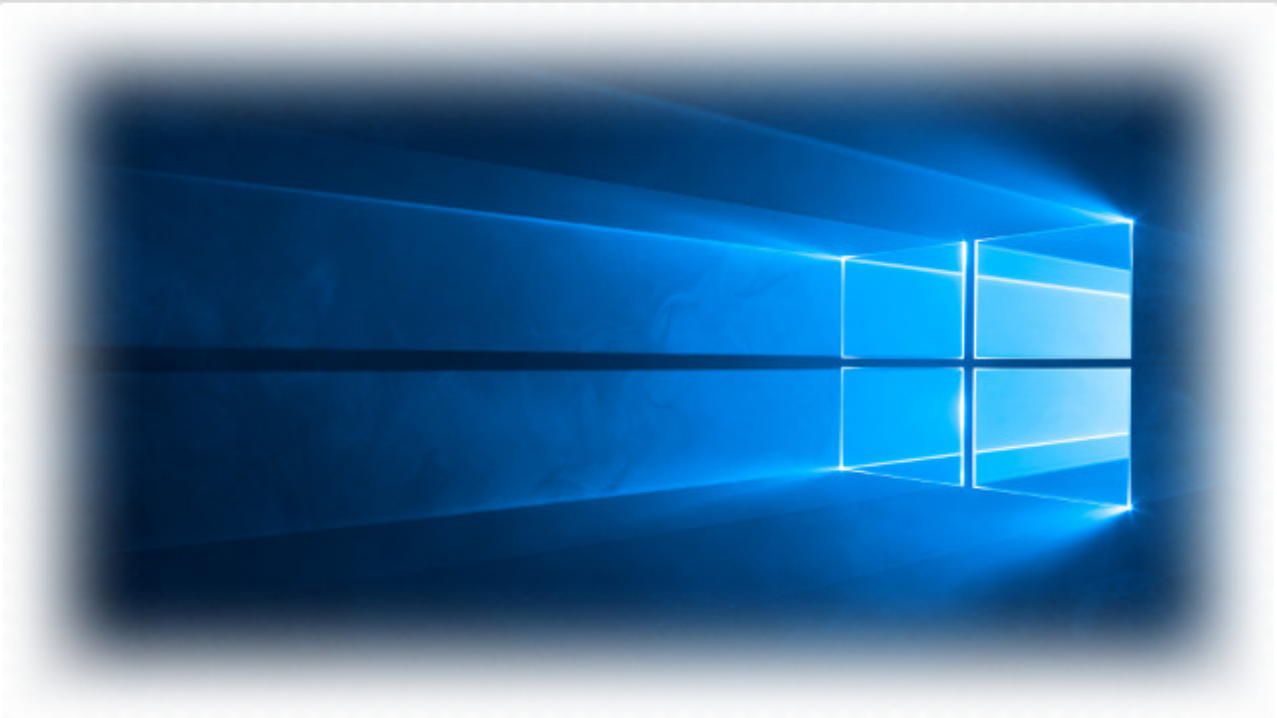
- Használt fájlrendszer: NTFS
- Forráskód: zárt

Windows összefoglaló



- Használt fájlrendszer: NTFS
- Forráskód: zárt
- PC-k nagy részén ez fut

Windows összefoglaló



- Használt fájlrendszer: NTFS
- Forráskód: zárt
- PC-k nagy részén ez fut
- Szakaszosan fejlődik, mindig van egy aktívan fejlesztett ág (pl Windows 10), míg a régebbiek már csak apró javításokat kapnak (pl Windows 7, 8.1), vagy már megszűnt a támogatásuk (pl Windows XP)

Linux összefoglaló

- Használt fájlrendszer: ext4

Linux összefoglaló

- Használt fájlrendszer: ext4
- Forráskód: nyílt

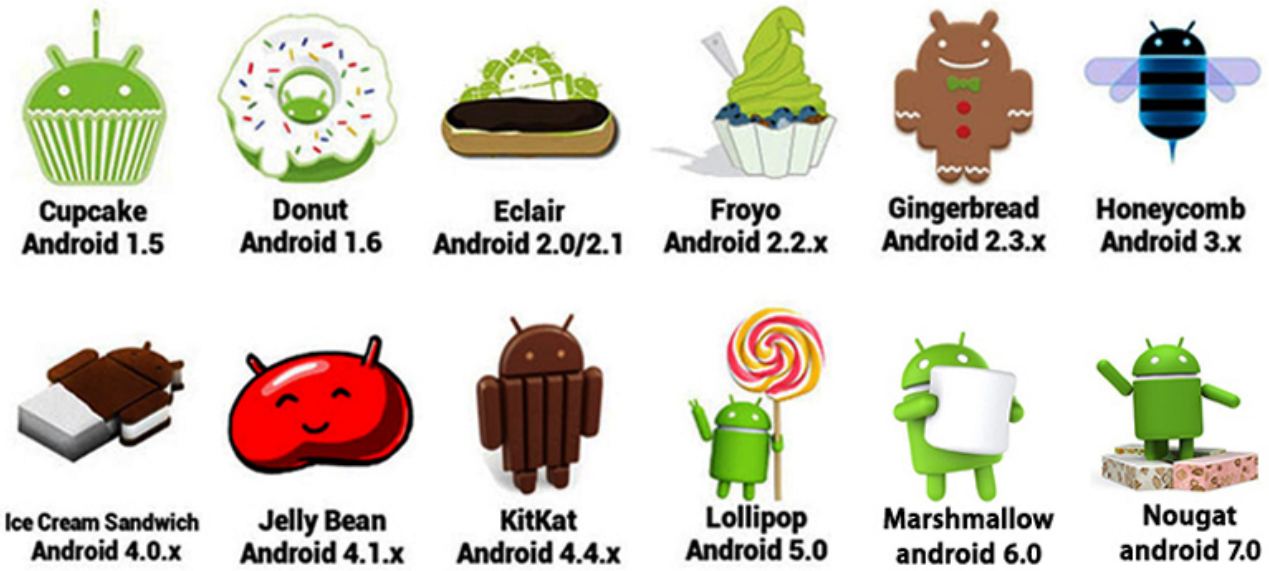
Linux összefoglaló

- Használt fájlrendszer: ext4
- Forráskód: nyílt
- Szervereken ez a legelterjedtebb, de PC-ken is használatos

Linux összefoglaló

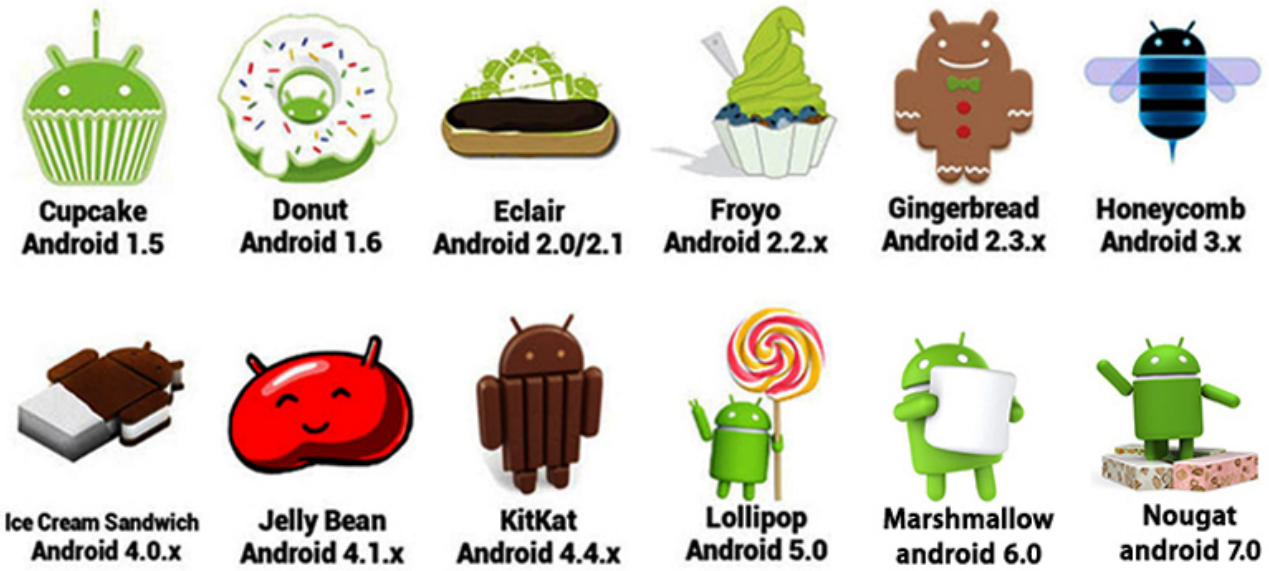
- Használt fájlrendszer: ext4
- Forráskód: nyílt
- Szervereken ez a legelterjedtebb, de PC-ken is használatos
- Több ágon folyik a fejlesztése, sokfajta disztribúció, vannak erősen kutatás és munka orientáltak (pl SUSE), és vannak felhasználóbarátak (pl Linux Mint, Ubuntu)

Android összefoglaló



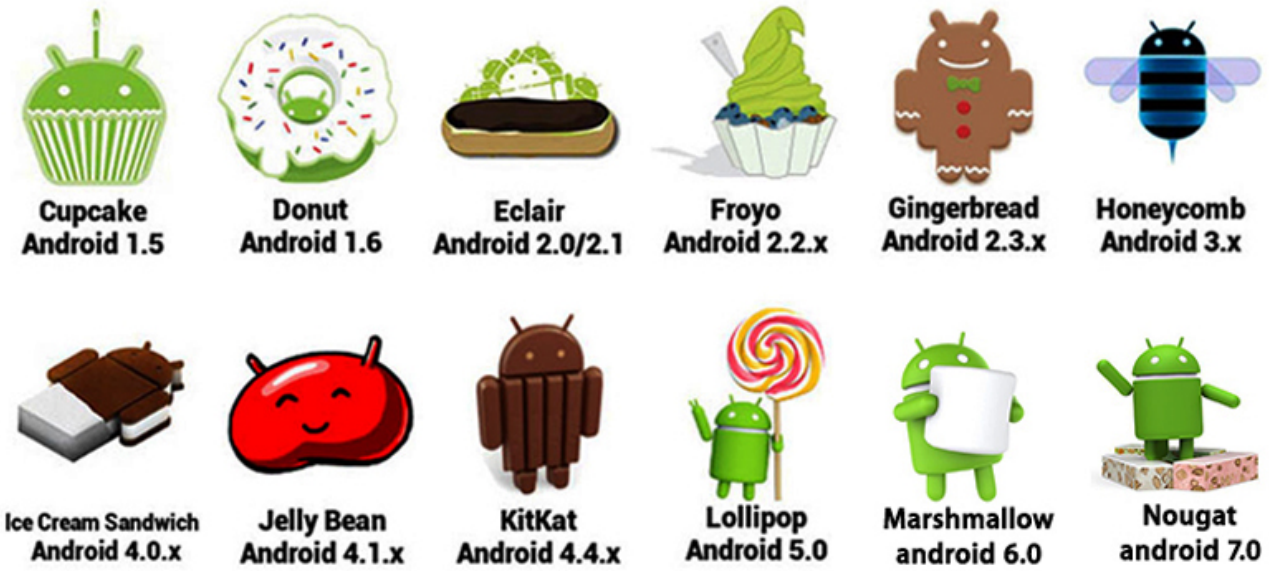
- Használt fájlrendszer: változó, flash háttértárra optimalizált: yaffs2, vfat (SD-kártyán), (Samsung: Flash-Friendly File System f2fs),. . .

Android összefoglaló



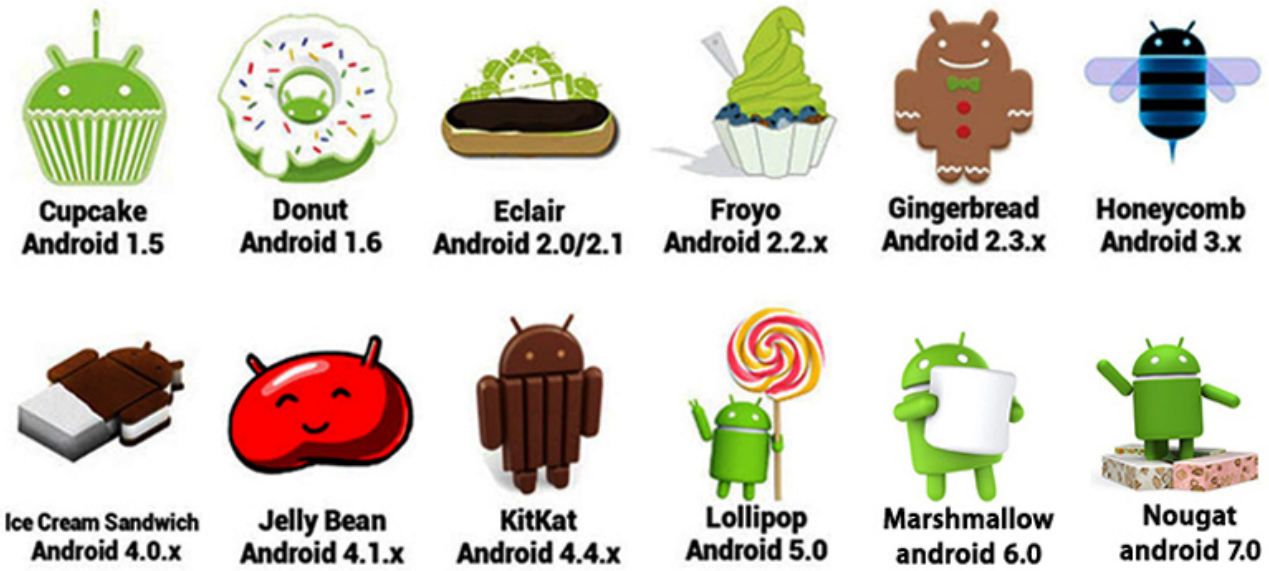
- Használt fájlrendszer: változó, flash háttértárra optimalizált: yaffs2, vfat (SD-kártyán), (Samsung: Flash-Friendly File System f2fs),. . .
- Forráskód: nyílt

Android összefoglaló



- Használt fájlrendszer: változó, flash háttértárra optimalizált: yaffs2, vfat (SD-kártyán), (Samsung: Flash-Friendly File System f2fs),. . .
- Forráskód: nyílt
- Megjelent: 2008 szeptember 23

Android összefoglaló



- Használt fájlrendszer: változó, flash háttértárra optimalizált: yaffs2, vfat (SD-kártyán), (Samsung: Flash-Friendly File System f2fs),. . .
- Forráskód: nyílt
- Megjelent: 2008 szeptember 23
- Céleszközök: telefon, tablet, karóra, TV, autó,. . .

Hálózat – IP cím

- Az internetre kötött gépek azonosítására szolgáló cím az IP cím (IP address), ami

Hálózat – IP cím

- Az internetre kötött gépek azonosítására szolgáló cím az IP cím (IP address), ami
  - IPv4 szabvány: `nnn.nnn.nnn.nnn` alakú (32 bites, 4 db 8-bites szám decimális alakban) – 2015 nyarán kifogyott

Hálózat – IP cím

- Az internetre kötött gépek azonosítására szolgáló cím az IP cím (IP address), ami
  - IPv4 szabvány: `nnn.nnn.nnn.nnn` alakú (32 bites, 4 db 8-bites szám decimális alakban) – 2015 nyarán kifogyott
  - IPv6 szabvány: xxxx:xxxx:xxxx:xxxx:xxxx:xxxx:xxxx:xxxx alakú (128 bit, 8 db 16 bites hexadecimálisan ábrázolt szám)

Hálózat – IP cím

- Az internetre kötött gépek azonosítására szolgáló cím az IP cím (IP address), ami
  - IPv4 szabvány: `nnn.nnn.nnn.nnn` alakú (32 bites, 4 db 8-bites szám decimális alakban) – 2015 nyarán kifogyott
  - IPv6 szabvány: xxxx:xxxx:xxxx:xxxx:xxxx:xxxx:xxxx:xxxx alakú (128 bit, 8 db 16 bites hexadecimálisan ábrázolt szám)

gép		IP cím	honnan tudom meg?
belső hálózat		172.17.148.238	ifconfig (WIN ipconfig)
		192.168.xxx.xxx	Reserved IP addresses
kifelé	IPv4:	152.66.83.241	http://miazipcimem.hu/
			http://www.howtofindmyipaddress.com/
	IPv6:	2001:738:2001:2010:891b:efb:2b36:5447	
			http://whatismyipaddress.com/
szerver		152.66.83.17	ping leibniz.math.bme.hu

A ping

- A ping egy rendszerprogram (utility), mely eldönti, hogy egy adatcsomag hibátlanul eljut-e a megadott IP címre.

```
C:\Users\Tofi>ping bme.hu

Pinging bme.hu [152.66.115.203] with 32 bytes of data:
Reply from 152.66.115.203: bytes=32 time=66ms TTL=52
Reply from 152.66.115.203: bytes=32 time=69ms TTL=52
Reply from 152.66.115.203: bytes=32 time=73ms TTL=52
Reply from 152.66.115.203: bytes=32 time=62ms TTL=52

Ping statistics for 152.66.115.203:
    Packets: Sent = 4, Received = 4, Lost = 0 (0% loss),
Approximate round trip times in milli-seconds:
    Minimum = 62ms, Maximum = 73ms, Average = 67ms

C:\Users\Tofi>_
```

A ping

- A ping egy rendszerprogram (utility), mely eldönti, hogy egy adatcsomag hibátlanul eljut-e a megadott IP címre.
- Ha a ping parancs után nem IP cím áll, hanem egy név, a DNS (Domain Name System) szolgátatással megtudja, hogy a szerver nevéhez (host name) milyen IP-cím tartozik, majd egy PING üzenetet küld a címre.

```
C:\Users\Tofi>ping bme.hu

Pinging bme.hu [152.66.115.203] with 32 bytes of data:
Reply from 152.66.115.203: bytes=32 time=66ms TTL=52
Reply from 152.66.115.203: bytes=32 time=69ms TTL=52
Reply from 152.66.115.203: bytes=32 time=73ms TTL=52
Reply from 152.66.115.203: bytes=32 time=62ms TTL=52

Ping statistics for 152.66.115.203:
    Packets: Sent = 4, Received = 4, Lost = 0 (0% loss),
Approximate round trip times in milli-seconds:
    Minimum = 62ms, Maximum = 73ms, Average = 67ms

C:\Users\Tofi>_
```

A ping

- A ping egy rendszerprogram (utility), mely eldönti, hogy egy adatcsomag hibátlanul eljut-e a megadott IP címre.
- Ha a ping parancs után nem IP cím áll, hanem egy név, a DNS (Domain Name System) szolgátatással megtudja, hogy a szerver nevéhez (host name) milyen IP-cím tartozik, majd egy PING üzenetet küld a címre.
- PING means "Send a packet to a computer and wait for its return (Packet INternet Groper)" (groper – molesztáló)

```
C:\Users\Tofi>ping bme.hu

Pinging bme.hu [152.66.115.203] with 32 bytes of data:
Reply from 152.66.115.203: bytes=32 time=66ms TTL=52
Reply from 152.66.115.203: bytes=32 time=69ms TTL=52
Reply from 152.66.115.203: bytes=32 time=73ms TTL=52
Reply from 152.66.115.203: bytes=32 time=62ms TTL=52

Ping statistics for 152.66.115.203:
    Packets: Sent = 4, Received = 4, Lost = 0 (0% loss),
Approximate round trip times in milli-seconds:
    Minimum = 62ms, Maximum = 73ms, Average = 67ms

C:\Users\Tofi>_
```

Bináris számok

Konverzió 2-es számrendszerből 10-es számrendszerbe:

$$b_n b_{n-1} \dots b_1 b_0 . b_{-1} \dots b_{-m} = \sum_{i=-m}^{n} b_i 2^i .$$

Bináris számok

Konverzió 2-es számrendszerből 10-es számrendszerbe:

$$b_n b_{n-1} \dots b_1 b_0 . b_{-1} \dots b_{-m} = \sum_{i=-m}^{n} b_i 2^i .$$

Például $110.101_2 =$

Bináris számok

Konverzió 2-es számrendszerből 10-es számrendszerbe:

$$b_n b_{n-1} \dots b_1 b_0 . b_{-1} \dots b_{-m} = \sum_{i=-m}^{n} b_i 2^i .$$

Például $110.101_2 = 6.625$

Bináris számok

Konverzió 2-es számrendszerből 10-es számrendszerbe:

$$b_n b_{n-1} \dots b_1 b_0 . b_{-1} \dots b_{-m} = \sum_{i=-m}^{n} b_i 2^i .$$

Például $110.101_2 = 6.625$

Konverzió 10-es számrendszerből 2-es számrendszerbe

- egészek esetén ismételt 2-vel való maradékos osztás,

Bináris számok

Konverzió 2-es számrendszerből 10-es számrendszerbe:

$$b_n b_{n-1} \dots b_1 b_0 . b_{-1} \dots b_{-m} = \sum_{i=-m}^{n} b_i 2^i .$$

Például $110.101_2 = 6.625$

Konverzió 10-es számrendszerből 2-es számrendszerbe

- egészek esetén ismételt 2-vel való maradékos osztás,
- törtrész esetén ismételt 2-vel való szorzás.

Bináris számok

Konverzió 2-es számrendszerből 10-es számrendszerbe:

$$b_n b_{n-1} \dots b_1 b_0 . b_{-1} \dots b_{-m} = \sum_{i=-m}^{n} b_i 2^i .$$

Például $110.101_2 = 6.625$

Konverzió 10-es számrendszerből 2-es számrendszerbe

- egészek esetén ismételt 2-vel való maradékos osztás,
- törtrész esetén ismételt 2-vel való szorzás.

Bináris számok

Konverzió 2-es számrendszerből 10-es számrendszerbe:

$$b_n b_{n-1} \dots b_1 b_0 . b_{-1} \dots b_{-m} = \sum_{i=-m}^{n} b_i 2^i .$$

Például $110.101_2 = 6.625$

Konverzió 10-es számrendszerből 2-es számrendszerbe

- egészek esetén ismételt 2-vel való maradékos osztás,
- törtrész esetén ismételt 2-vel való szorzás.

Például 106 bináris alakja:

Bináris számok

Konverzió 2-es számrendszerből 10-es számrendszerbe:

$$b_n b_{n-1} \dots b_1 b_0 . b_{-1} \dots b_{-m} = \sum_{i=-m}^{n} b_i 2^i .$$

Például $110.101_2 = 6.625$

Konverzió 10-es számrendszerből 2-es számrendszerbe

- egészek esetén ismételt 2-vel való maradékos osztás,
- törtrész esetén ismételt 2-vel való szorzás.

Például 106 bináris alakja:

$106 = 2 \cdot 53 + 0$

Bináris számok

Konverzió 2-es számrendszerből 10-es számrendszerbe:

$$b_n b_{n-1} \dots b_1 b_0 . b_{-1} \dots b_{-m} = \sum_{i=-m}^{n} b_i 2^i .$$

Például $110.101_2 = 6.625$

Konverzió 10-es számrendszerből 2-es számrendszerbe

- egészek esetén ismételt 2-vel való maradékos osztás,
- törtrész esetén ismételt 2-vel való szorzás.

Például 106 bináris alakja:

$106 = 2 \cdot 53 + 0 \rightarrow 0$

Bináris számok

Konverzió 2-es számrendszerből 10-es számrendszerbe:

$$b_n b_{n-1} \dots b_1 b_0 . b_{-1} \dots b_{-m} = \sum_{i=-m}^{n} b_i 2^i .$$

Például $110.101_2 = 6.625$

Konverzió 10-es számrendszerből 2-es számrendszerbe

- egészek esetén ismételt 2-vel való maradékos osztás,
- törtrész esetén ismételt 2-vel való szorzás.

Például 106 bináris alakja:

$106 = 2 \cdot 53 + 0 \rightarrow 0$

$53 = 2 \cdot 26 + 1$

Bináris számok

Konverzió 2-es számrendszerből 10-es számrendszerbe:

$$b_n b_{n-1} \dots b_1 b_0 . b_{-1} \dots b_{-m} = \sum_{i=-m}^{n} b_i 2^i .$$

Például $110.101_2 = 6.625$

Konverzió 10-es számrendszerből 2-es számrendszerbe

- egészek esetén ismételt 2-vel való maradékos osztás,
- törtrész esetén ismételt 2-vel való szorzás.

Például 106 bináris alakja:

$106 = 2 \cdot 53 + 0 \rightarrow 0$

$53 = 2 \cdot 26 + 1 \rightarrow 1$

Bináris számok

Konverzió 2-es számrendszerből 10-es számrendszerbe:

$$b_n b_{n-1} \dots b_1 b_0 . b_{-1} \dots b_{-m} = \sum_{i=-m}^{n} b_i 2^i .$$

Például $110.101_2 = 6.625$

Konverzió 10-es számrendszerből 2-es számrendszerbe

- egészek esetén ismételt 2-vel való maradékos osztás,
- törtrész esetén ismételt 2-vel való szorzás.

Például 106 bináris alakja:

$106 = 2 \cdot 53 + 0 \rightarrow 0$

$53 = 2 \cdot 26 + 1 \rightarrow 1$

$26 = 2 \cdot 13 + 0$

Bináris számok

Konverzió 2-es számrendszerből 10-es számrendszerbe:

$$b_n b_{n-1} \dots b_1 b_0 . b_{-1} \dots b_{-m} = \sum_{i=-m}^{n} b_i 2^i .$$

Például $110.101_2 = 6.625$

Konverzió 10-es számrendszerből 2-es számrendszerbe

- egészek esetén ismételt 2-vel való maradékos osztás,
- törtrész esetén ismételt 2-vel való szorzás.

Például 106 bináris alakja:

$106 = 2 \cdot 53 + 0 \rightarrow 0$

$53 = 2 \cdot 26 + 1 \rightarrow 1$

$26 = 2 \cdot 13 + 0 \rightarrow 0$

Bináris számok

Konverzió 2-es számrendszerből 10-es számrendszerbe:

$$b_n b_{n-1} \dots b_1 b_0 . b_{-1} \dots b_{-m} = \sum_{i=-m}^{n} b_i 2^i .$$

Például $110.101_2 = 6.625$

Konverzió 10-es számrendszerből 2-es számrendszerbe

- egészek esetén ismételt 2-vel való maradékos osztás,
- törtrész esetén ismételt 2-vel való szorzás.

Például 106 bináris alakja:

$106 = 2 \cdot 53 + 0 \rightarrow 0$

$53 = 2 \cdot 26 + 1 \rightarrow 1$

$26 = 2 \cdot 13 + 0 \rightarrow 0$

$13 = 2 \cdot \ \ 6 + 1$

Bináris számok

Konverzió 2-es számrendszerből 10-es számrendszerbe:

$$b_n b_{n-1} \dots b_1 b_0 . b_{-1} \dots b_{-m} = \sum_{i=-m}^{n} b_i 2^i .$$

Például $110.101_2 = 6.625$

Konverzió 10-es számrendszerből 2-es számrendszerbe

- egészek esetén ismételt 2-vel való maradékos osztás,
- törtrész esetén ismételt 2-vel való szorzás.

Például 106 bináris alakja:

$106 = 2 \cdot 53 + 0 \rightarrow 0$

$53 = 2 \cdot 26 + 1 \rightarrow 1$

$26 = 2 \cdot 13 + 0 \rightarrow 0$

$13 = 2 \cdot \ \ 6 + 1 \rightarrow 1$

Bináris számok

Konverzió 2-es számrendszerből 10-es számrendszerbe:

$$b_n b_{n-1} \dots b_1 b_0 . b_{-1} \dots b_{-m} = \sum_{i=-m}^{n} b_i 2^i .$$

Például $110.101_2 = 6.625$

Konverzió 10-es számrendszerből 2-es számrendszerbe

- egészek esetén ismételt 2-vel való maradékos osztás,
- törtrész esetén ismételt 2-vel való szorzás.

Például 106 bináris alakja:

$$\begin{aligned} 106 &= 2 \cdot 53 + 0 \to 0 \\ 53 &= 2 \cdot 26 + 1 \to 1 \\ 26 &= 2 \cdot 13 + 0 \to 0 \\ 13 &= 2 \cdot \ 6 + 1 \to 1 \\ 6 &= 2 \cdot \ 3 + 0 \end{aligned}$$

Bináris számok

Konverzió 2-es számrendszerből 10-es számrendszerbe:

$$b_n b_{n-1} \dots b_1 b_0 . b_{-1} \dots b_{-m} = \sum_{i=-m}^{n} b_i 2^i .$$

Például $110.101_2 = 6.625$

Konverzió 10-es számrendszerből 2-es számrendszerbe

- egészek esetén ismételt 2-vel való maradékos osztás,
- törtrész esetén ismételt 2-vel való szorzás.

Például 106 bináris alakja:

$$\begin{aligned} 106 &= 2 \cdot 53 + 0 \to 0 \\ 53 &= 2 \cdot 26 + 1 \to 1 \\ 26 &= 2 \cdot 13 + 0 \to 0 \\ 13 &= 2 \cdot \ 6 + 1 \to 1 \\ 6 &= 2 \cdot \ 3 + 0 \to 0 \end{aligned}$$

Bináris számok

Konverzió 2-es számrendszerből 10-es számrendszerbe:

$$b_n b_{n-1} \dots b_1 b_0 . b_{-1} \dots b_{-m} = \sum_{i=-m}^{n} b_i 2^i .$$

Például $110.101_2 = 6.625$

Konverzió 10-es számrendszerből 2-es számrendszerbe

- egészek esetén ismételt 2-vel való maradékos osztás,
- törtrész esetén ismételt 2-vel való szorzás.

Például 106 bináris alakja:

$$\begin{aligned}
106 &= 2 \cdot 53 + 0 \to 0 \\
53 &= 2 \cdot 26 + 1 \to 1 \\
26 &= 2 \cdot 13 + 0 \to 0 \\
13 &= 2 \cdot 6 + 1 \to 1 \\
6 &= 2 \cdot 3 + 0 \to 0 \\
3 &= 2 \cdot 1 + 1
\end{aligned}$$

Bináris számok

Konverzió 2-es számrendszerből 10-es számrendszerbe:

$$b_n b_{n-1} \dots b_1 b_0 . b_{-1} \dots b_{-m} = \sum_{i=-m}^{n} b_i 2^i .$$

Például $110.101_2 = 6.625$

Konverzió 10-es számrendszerből 2-es számrendszerbe

- egészek esetén ismételt 2-vel való maradékos osztás,
- törtrész esetén ismételt 2-vel való szorzás.

Például 106 bináris alakja:

$$\begin{aligned}
106 &= 2 \cdot 53 + 0 \to 0 \\
53 &= 2 \cdot 26 + 1 \to 1 \\
26 &= 2 \cdot 13 + 0 \to 0 \\
13 &= 2 \cdot 6 + 1 \to 1 \\
6 &= 2 \cdot 3 + 0 \to 0 \\
3 &= 2 \cdot 1 + 1 \to 1
\end{aligned}$$

Bináris számok

Konverzió 2-es számrendszerből 10-es számrendszerbe:

$$b_n b_{n-1} \dots b_1 b_0 . b_{-1} \dots b_{-m} = \sum_{i=-m}^{n} b_i 2^i .$$

Például $110.101_2 = 6.625$

Konverzió 10-es számrendszerből 2-es számrendszerbe

- egészek esetén ismételt 2-vel való maradékos osztás,
- törtrész esetén ismételt 2-vel való szorzás.

Például 106 bináris alakja:

$$\begin{aligned}
106 &= 2 \cdot 53 + 0 \to 0 \\
53 &= 2 \cdot 26 + 1 \to 1 \\
26 &= 2 \cdot 13 + 0 \to 0 \\
13 &= 2 \cdot 6 + 1 \to 1 \\
6 &= 2 \cdot 3 + 0 \to 0 \\
3 &= 2 \cdot 1 + 1 \to 1 \\
1 &= 2 \cdot 0 + 1
\end{aligned}$$

Bináris számok

Konverzió 2-es számrendszerből 10-es számrendszerbe:

$$b_n b_{n-1} \dots b_1 b_0 . b_{-1} \dots b_{-m} = \sum_{i=-m}^{n} b_i 2^i .$$

Például $110.101_2 = 6.625$

Konverzió 10-es számrendszerből 2-es számrendszerbe

- egészek esetén ismételt 2-vel való maradékos osztás,
- törtrész esetén ismételt 2-vel való szorzás.

Például 106 bináris alakja:

$$\begin{aligned}
106 &= 2 \cdot 53 + 0 \to 0 \\
53 &= 2 \cdot 26 + 1 \to 1 \\
26 &= 2 \cdot 13 + 0 \to 0 \\
13 &= 2 \cdot 6 + 1 \to 1 \\
6 &= 2 \cdot 3 + 0 \to 0 \\
3 &= 2 \cdot 1 + 1 \to 1 \\
1 &= 2 \cdot 0 + 1 \to 1
\end{aligned}$$

Bináris számok

Konverzió 2-es számrendszerből 10-es számrendszerbe:

$$b_n b_{n-1} \dots b_1 b_0 . b_{-1} \dots b_{-m} = \sum_{i=-m}^{n} b_i 2^i .$$

Például $110.101_2 = 6.625$

Konverzió 10-es számrendszerből 2-es számrendszerbe

- egészek esetén ismételt 2-vel való maradékos osztás,
- törtrész esetén ismételt 2-vel való szorzás.

Például 106 bináris alakja:

$$\begin{aligned}
106 &= 2 \cdot 53 + 0 \to 0 \\
53 &= 2 \cdot 26 + 1 \to 1 \\
26 &= 2 \cdot 13 + 0 \to 0 \\
13 &= 2 \cdot 6 + 1 \to 1 \\
6 &= 2 \cdot 3 + 0 \to 0 \\
3 &= 2 \cdot 1 + 1 \to 1 \\
1 &= 2 \cdot 0 + 1 \to 1
\end{aligned}$$

tehát a bináris alak 1101010.

Bináris számok

Konverzió 2-es számrendszerből 10-es számrendszerbe:

$$b_n b_{n-1} \dots b_1 b_0 . b_{-1} \dots b_{-m} = \sum_{i=-m}^{n} b_i 2^i .$$

Például $110.101_2 = 6.625$

Konverzió 10-es számrendszerből 2-es számrendszerbe

- egészek esetén ismételt 2-vel való maradékos osztás,
- törtrész esetén ismételt 2-vel való szorzás.

Például 106 bináris alakja:

$$\begin{aligned} 106 &= 2 \cdot 53 + 0 \to 0 \\ 53 &= 2 \cdot 26 + 1 \to 1 \\ 26 &= 2 \cdot 13 + 0 \to 0 \\ 13 &= 2 \cdot 6 + 1 \to 1 \\ 6 &= 2 \cdot 3 + 0 \to 0 \\ 3 &= 2 \cdot 1 + 1 \to 1 \\ 1 &= 2 \cdot 0 + 1 \to 1 \end{aligned}$$

106	2
53	0

tehát a bináris alak 1101010.

Bináris számok

Konverzió 2-es számrendszerből 10-es számrendszerbe:

$$b_n b_{n-1} \dots b_1 b_0 . b_{-1} \dots b_{-m} = \sum_{i=-m}^{n} b_i 2^i .$$

Például $110.101_2 = 6.625$

Konverzió 10-es számrendszerből 2-es számrendszerbe

- egészek esetén ismételt 2-vel való maradékos osztás,
- törtrész esetén ismételt 2-vel való szorzás.

Például 106 bináris alakja:

$$\begin{aligned}
106 &= 2 \cdot 53 + 0 \to 0 \\
53 &= 2 \cdot 26 + 1 \to 1 \\
26 &= 2 \cdot 13 + 0 \to 0 \\
13 &= 2 \cdot 6 + 1 \to 1 \\
6 &= 2 \cdot 3 + 0 \to 0 \\
3 &= 2 \cdot 1 + 1 \to 1 \\
1 &= 2 \cdot 0 + 1 \to 1
\end{aligned}$$

106	2
53	0
26	1

tehát a bináris alak 1101010.

Bináris számok

Konverzió 2-es számrendszerből 10-es számrendszerbe:

$$b_n b_{n-1} \dots b_1 b_0 . b_{-1} \dots b_{-m} = \sum_{i=-m}^{n} b_i 2^i .$$

Például $110.101_2 = 6.625$

Konverzió 10-es számrendszerből 2-es számrendszerbe

- egészek esetén ismételt 2-vel való maradékos osztás,
- törtrész esetén ismételt 2-vel való szorzás.

Például 106 bináris alakja:

$$\begin{aligned}
106 &= 2 \cdot 53 + 0 \to 0 \\
53 &= 2 \cdot 26 + 1 \to 1 \\
26 &= 2 \cdot 13 + 0 \to 0 \\
13 &= 2 \cdot 6 + 1 \to 1 \\
6 &= 2 \cdot 3 + 0 \to 0 \\
3 &= 2 \cdot 1 + 1 \to 1 \\
1 &= 2 \cdot 0 + 1 \to 1
\end{aligned}$$

106	2
53	0
26	1
13	0

tehát a bináris alak 1101010.

Bináris számok

Konverzió 2-es számrendszerből 10-es számrendszerbe:

$$b_n b_{n-1} \dots b_1 b_0 . b_{-1} \dots b_{-m} = \sum_{i=-m}^{n} b_i 2^i .$$

Például $110.101_2 = 6.625$

Konverzió 10-es számrendszerből 2-es számrendszerbe

- egészek esetén ismételt 2-vel való maradékos osztás,
- törtrész esetén ismételt 2-vel való szorzás.

Például 106 bináris alakja:

$$\begin{aligned}
106 &= 2 \cdot 53 + 0 \to 0 \\
53 &= 2 \cdot 26 + 1 \to 1 \\
26 &= 2 \cdot 13 + 0 \to 0 \\
13 &= 2 \cdot 6 + 1 \to 1 \\
6 &= 2 \cdot 3 + 0 \to 0 \\
3 &= 2 \cdot 1 + 1 \to 1 \\
1 &= 2 \cdot 0 + 1 \to 1
\end{aligned}$$

106	2
53	0
26	1
13	0
6	1

tehát a bináris alak 1101010.

Bináris számok

Konverzió 2-es számrendszerből 10-es számrendszerbe:

$$b_n b_{n-1} \dots b_1 b_0 . b_{-1} \dots b_{-m} = \sum_{i=-m}^{n} b_i 2^i .$$

Például $110.101_2 = 6.625$

Konverzió 10-es számrendszerből 2-es számrendszerbe

- egészek esetén ismételt 2-vel való maradékos osztás,
- törtrész esetén ismételt 2-vel való szorzás.

Például 106 bináris alakja:

$$\begin{aligned}
106 &= 2 \cdot 53 + 0 \to 0 \\
53 &= 2 \cdot 26 + 1 \to 1 \\
26 &= 2 \cdot 13 + 0 \to 0 \\
13 &= 2 \cdot 6 + 1 \to 1 \\
6 &= 2 \cdot 3 + 0 \to 0 \\
3 &= 2 \cdot 1 + 1 \to 1 \\
1 &= 2 \cdot 0 + 1 \to 1
\end{aligned}$$

106	2
53	0
26	1
13	0
6	1
3	0

tehát a bináris alak 1101010.

Bináris számok

Konverzió 2-es számrendszerből 10-es számrendszerbe:

$$b_n b_{n-1} \dots b_1 b_0 . b_{-1} \dots b_{-m} = \sum_{i=-m}^{n} b_i 2^i .$$

Például $110.101_2 = 6.625$

Konverzió 10-es számrendszerből 2-es számrendszerbe

- egészek esetén ismételt 2-vel való maradékos osztás,
- törtrész esetén ismételt 2-vel való szorzás.

Például 106 bináris alakja:

$$\begin{aligned}
106 &= 2 \cdot 53 + 0 \to 0 \\
53 &= 2 \cdot 26 + 1 \to 1 \\
26 &= 2 \cdot 13 + 0 \to 0 \\
13 &= 2 \cdot 6 + 1 \to 1 \\
6 &= 2 \cdot 3 + 0 \to 0 \\
3 &= 2 \cdot 1 + 1 \to 1 \\
1 &= 2 \cdot 0 + 1 \to 1
\end{aligned}$$

106	2
53	0
26	1
13	0
6	1
3	0
1	1

tehát a bináris alak 1101010.

Bináris számok

Konverzió 2-es számrendszerből 10-es számrendszerbe:

$$b_n b_{n-1} \dots b_1 b_0 . b_{-1} \dots b_{-m} = \sum_{i=-m}^{n} b_i 2^i .$$

Például $110.101_2 = 6.625$

Konverzió 10-es számrendszerből 2-es számrendszerbe

- egészek esetén ismételt 2-vel való maradékos osztás,
- törtrész esetén ismételt 2-vel való szorzás.

Például 106 bináris alakja:

$$\begin{aligned}
106 &= 2 \cdot 53 + 0 \to 0 \\
53 &= 2 \cdot 26 + 1 \to 1 \\
26 &= 2 \cdot 13 + 0 \to 0 \\
13 &= 2 \cdot 6 + 1 \to 1 \\
6 &= 2 \cdot 3 + 0 \to 0 \\
3 &= 2 \cdot 1 + 1 \to 1 \\
1 &= 2 \cdot 0 + 1 \to 1
\end{aligned}$$

tehát a bináris alak 1101010.

106	2
53	0
26	1
13	0
6	1
3	0
1	1
0	1

Bináris számok

Konverzió 2-es számrendszerből 10-es számrendszerbe:

$$b_n b_{n-1} \dots b_1 b_0 . b_{-1} \dots b_{-m} = \sum_{i=-m}^{n} b_i 2^i .$$

Például $110.101_2 = 6.625$

Konverzió 10-es számrendszerből 2-es számrendszerbe

- egészek esetén ismételt 2-vel való maradékos osztás,
- törtrész esetén ismételt 2-vel való szorzás.

Például 106 bináris alakja:

$$\begin{aligned}
106 &= 2 \cdot 53 + 0 \to 0 \\
53 &= 2 \cdot 26 + 1 \to 1 \\
26 &= 2 \cdot 13 + 0 \to 0 \\
13 &= 2 \cdot 6 + 1 \to 1 \\
6 &= 2 \cdot 3 + 0 \to 0 \\
3 &= 2 \cdot 1 + 1 \to 1 \\
1 &= 2 \cdot 0 + 1 \to 1
\end{aligned}$$

tehát a bináris alak 1101010.

106	2
53	0
26	1
13	0
6	1
3	0
1	1
0	1

Example

Hogyan konvertálunk tizedes törtet binárissá?

Example

Hogyan konvertálunk tizedes törtet binárissá? Pl. írjuk fel 0.3 bináris alakjának tizedespont utáni első 6 jegyét!

Example

Hogyan konvertálunk tizedes törtet binárissá? Pl. írjuk fel 0.3 bináris alakjának tizedespont utáni első 6 jegyét!

Megoldás: A tizedespont utáni jegyek jelentése bináris esetben $1/2$, $1/4, \ldots, 1/2^n, \ldots$. Pl. a bináris 0.1011001 számot mindig 2-vel szorozva az eredmény egész része rendre 1, 0, 1, 1, 0, 0, 1.

Example

Hogyan konvertálunk tizedes törtet binárissá? Pl. írjuk fel 0.3 bináris alakjának tizedespont utáni első 6 jegyét!

Megoldás: A tizedespont utáni jegyek jelentése bináris esetben $1/2$, $1/4, \ldots, 1/2^n, \ldots$. Pl. a bináris 0.1011001 számot mindig 2-vel szorozva az eredmény egész része rendre 1, 0, 1, 1, 0, 0, 1. Ezt használva:
$0.3 \cdot 2 = 0.6$

Example

Hogyan konvertálunk tizedes törtet binárissá? Pl. írjuk fel 0.3 bináris alakjának tizedespont utáni első 6 jegyét!

Megoldás: A tizedespont utáni jegyek jelentése bináris esetben $1/2$, $1/4, \ldots, 1/2^n, \ldots$. Pl. a bináris 0.1011001 számot mindig 2-vel szorozva az eredmény egész része rendre 1, 0, 1, 1, 0, 0, 1. Ezt használva:
$0.3 \cdot 2 = 0.6 \rightarrow 0$

Example

Hogyan konvertálunk tizedes törtet binárissá? Pl. írjuk fel 0.3 bináris alakjának tizedespont utáni első 6 jegyét!

Megoldás: A tizedespont utáni jegyek jelentése bináris esetben $1/2$, $1/4, \ldots, 1/2^n, \ldots$. Pl. a bináris 0.1011001 számot mindig 2-vel szorozva az eredmény egész része rendre 1, 0, 1, 1, 0, 0, 1. Ezt használva:

$0.3 \cdot 2 = 0.6 \rightarrow 0$

$0.6 \cdot 2 = 1.2$

Example

Hogyan konvertálunk tizedes törtet binárissá? Pl. írjuk fel 0.3 bináris alakjának tizedespont utáni első 6 jegyét!

Megoldás: A tizedespont utáni jegyek jelentése bináris esetben $1/2$, $1/4, \ldots, 1/2^n, \ldots$. Pl. a bináris 0.1011001 számot mindig 2-vel szorozva az eredmény egész része rendre 1, 0, 1, 1, 0, 0, 1. Ezt használva:

$0.3 \cdot 2 = 0.6 \to 0$

$0.6 \cdot 2 = 1.2 \to 1$

Example

Hogyan konvertálunk tizedes törtet binárissá? Pl. írjuk fel 0.3 bináris alakjának tizedespont utáni első 6 jegyét!

Megoldás: A tizedespont utáni jegyek jelentése bináris esetben $1/2$, $1/4, \ldots, 1/2^n, \ldots$. Pl. a bináris 0.1011001 számot mindig 2-vel szorozva az eredmény egész része rendre 1, 0, 1, 1, 0, 0, 1. Ezt használva:

$0.3 \cdot 2 = 0.6 \rightarrow 0$

$0.6 \cdot 2 = 1.2 \rightarrow 1$

$0.2 \cdot 2 = 0.4$

Example

Hogyan konvertálunk tizedes törtet binárissá? Pl. írjuk fel 0.3 bináris alakjának tizedespont utáni első 6 jegyét!

Megoldás: A tizedespont utáni jegyek jelentése bináris esetben $1/2$, $1/4, \ldots, 1/2^n, \ldots$. Pl. a bináris 0.1011001 számot mindig 2-vel szorozva az eredmény egész része rendre 1, 0, 1, 1, 0, 0, 1. Ezt használva:

$0.3 \cdot 2 = 0.6 \to 0$

$0.6 \cdot 2 = 1.2 \to 1$

$0.2 \cdot 2 = 0.4 \to 0$

Example

Hogyan konvertálunk tizedes törtet binárissá? Pl. írjuk fel 0.3 bináris alakjának tizedespont utáni első 6 jegyét!

Megoldás: A tizedespont utáni jegyek jelentése bináris esetben $1/2$, $1/4, \ldots, 1/2^n, \ldots$. Pl. a bináris 0.1011001 számot mindig 2-vel szorozva az eredmény egész része rendre 1, 0, 1, 1, 0, 0, 1. Ezt használva:

$0.3 \cdot 2 = 0.6 \rightarrow 0$

$0.6 \cdot 2 = 1.2 \rightarrow 1$

$0.2 \cdot 2 = 0.4 \rightarrow 0$

$0.4 \cdot 2 = 0.8$

Example

Hogyan konvertálunk tizedes törtet binárissá? Pl. írjuk fel 0.3 bináris alakjának tizedespont utáni első 6 jegyét!

Megoldás: A tizedespont utáni jegyek jelentése bináris esetben $1/2$, $1/4, \ldots, 1/2^n, \ldots$. Pl. a bináris 0.1011001 számot mindig 2-vel szorozva az eredmény egész része rendre 1, 0, 1, 1, 0, 0, 1. Ezt használva:

$0.3 \cdot 2 = 0.6 \to 0$

$0.6 \cdot 2 = 1.2 \to 1$

$0.2 \cdot 2 = 0.4 \to 0$

$0.4 \cdot 2 = 0.8 \to 0$

Example

Hogyan konvertálunk tizedes törtet binárissá? Pl. írjuk fel 0.3 bináris alakjának tizedespont utáni első 6 jegyét!

Megoldás: A tizedespont utáni jegyek jelentése bináris esetben $1/2$, $1/4, \ldots, 1/2^n, \ldots$. Pl. a bináris 0.1011001 számot mindig 2-vel szorozva az eredmény egész része rendre 1, 0, 1, 1, 0, 0, 1. Ezt használva:

$0.3 \cdot 2 = 0.6 \rightarrow 0$

$0.6 \cdot 2 = 1.2 \rightarrow 1$

$0.2 \cdot 2 = 0.4 \rightarrow 0$

$0.4 \cdot 2 = 0.8 \rightarrow 0$

$0.8 \cdot 2 = 1.6$

Example

Hogyan konvertálunk tizedes törtet binárissá? Pl. írjuk fel 0.3 bináris alakjának tizedespont utáni első 6 jegyét!

Megoldás: A tizedespont utáni jegyek jelentése bináris esetben $1/2$, $1/4, \ldots, 1/2^n, \ldots$. Pl. a bináris 0.1011001 számot mindig 2-vel szorozva az eredmény egész része rendre 1, 0, 1, 1, 0, 0, 1. Ezt használva:

$0.3 \cdot 2 = 0.6 \to 0$

$0.6 \cdot 2 = 1.2 \to 1$

$0.2 \cdot 2 = 0.4 \to 0$

$0.4 \cdot 2 = 0.8 \to 0$

$0.8 \cdot 2 = 1.6 \to 1$

Example

Hogyan konvertálunk tizedes törtet binárissá? Pl. írjuk fel 0.3 bináris alakjának tizedespont utáni első 6 jegyét!

Megoldás: A tizedespont utáni jegyek jelentése bináris esetben $1/2$, $1/4, \ldots, 1/2^n, \ldots$. Pl. a bináris 0.1011001 számot mindig 2-vel szorozva az eredmény egész része rendre 1, 0, 1, 1, 0, 0, 1. Ezt használva:

$0.3 \cdot 2 = 0.6 \to 0$

$0.6 \cdot 2 = 1.2 \to 1$

$0.2 \cdot 2 = 0.4 \to 0$

$0.4 \cdot 2 = 0.8 \to 0$

$0.8 \cdot 2 = 1.6 \to 1$

$0.6 \cdot 2 = 1.2$

Example

Hogyan konvertálunk tizedes törtet binárissá? Pl. írjuk fel 0.3 bináris alakjának tizedespont utáni első 6 jegyét!

Megoldás: A tizedespont utáni jegyek jelentése bináris esetben $1/2$, $1/4, \ldots, 1/2^n, \ldots$. Pl. a bináris 0.1011001 számot mindig 2-vel szorozva az eredmény egész része rendre 1, 0, 1, 1, 0, 0, 1. Ezt használva:

$0.3 \cdot 2 = 0.6 \to 0$
$0.6 \cdot 2 = 1.2 \to 1$
$0.2 \cdot 2 = 0.4 \to 0$
$0.4 \cdot 2 = 0.8 \to 0$
$0.8 \cdot 2 = 1.6 \to 1$
$0.6 \cdot 2 = 1.2 \to 1$

Example

Hogyan konvertálunk tizedes törtet binárissá? Pl. írjuk fel 0.3 bináris alakjának tizedespont utáni első 6 jegyét!

Megoldás: A tizedespont utáni jegyek jelentése bináris esetben $1/2$, $1/4, \ldots, 1/2^n, \ldots$. Pl. a bináris 0.1011001 számot mindig 2-vel szorozva az eredmény egész része rendre 1, 0, 1, 1, 0, 0, 1. Ezt használva:

$0.3 \cdot 2 = 0.6 \to 0$
$0.6 \cdot 2 = 1.2 \to 1$
$0.2 \cdot 2 = 0.4 \to 0$
$0.4 \cdot 2 = 0.8 \to 0$
$0.8 \cdot 2 = 1.6 \to 1$
$0.6 \cdot 2 = 1.2 \to 1$

Azaz 0.3 bináris alakja 0.010011, sőt az is látszik, hogy a végtelen bináris alak: $0.0\dot{1}00\dot{1}$.

Example

Hogyan konvertálunk tizedes törtet binárissá? Pl. írjuk fel 0.3 bináris alakjának tizedespont utáni első 6 jegyét!

Megoldás: A tizedespont utáni jegyek jelentése bináris esetben $1/2$, $1/4, \ldots, 1/2^n, \ldots$. Pl. a bináris 0.1011001 számot mindig 2-vel szorozva az eredmény egész része rendre 1, 0, 1, 1, 0, 0, 1. Ezt használva:

$0.3 \cdot 2 = 0.6 \rightarrow 0$
$0.6 \cdot 2 = 1.2 \rightarrow 1$
$0.2 \cdot 2 = 0.4 \rightarrow 0$
$0.4 \cdot 2 = 0.8 \rightarrow 0$
$0.8 \cdot 2 = 1.6 \rightarrow 1$
$0.6 \cdot 2 = 1.2 \rightarrow 1$

0.3	2
0.6	0

Azaz 0.3 bináris alakja 0.010011, sőt az is látszik, hogy a végtelen bináris alak: $0.0\dot{1}00\dot{1}1$.

Example

Hogyan konvertálunk tizedes törtet binárissá? Pl. írjuk fel 0.3 bináris alakjának tizedespont utáni első 6 jegyét!

Megoldás: A tizedespont utáni jegyek jelentése bináris esetben $1/2$, $1/4, \ldots, 1/2^n, \ldots$. Pl. a bináris 0.1011001 számot mindig 2-vel szorozva az eredmény egész része rendre 1, 0, 1, 1, 0, 0, 1. Ezt használva:

$0.3 \cdot 2 = 0.6 \rightarrow 0$
$0.6 \cdot 2 = 1.2 \rightarrow 1$
$0.2 \cdot 2 = 0.4 \rightarrow 0$
$0.4 \cdot 2 = 0.8 \rightarrow 0$
$0.8 \cdot 2 = 1.6 \rightarrow 1$
$0.6 \cdot 2 = 1.2 \rightarrow 1$

0.3	2
0.6	0
1.2	1

Azaz 0.3 bináris alakja 0.010011, sőt az is látszik, hogy a végtelen bináris alak: $0.0\dot{1}00\dot{1}1$.

Example

Hogyan konvertálunk tizedes törtet binárissá? Pl. írjuk fel 0.3 bináris alakjának tizedespont utáni első 6 jegyét!

Megoldás: A tizedespont utáni jegyek jelentése bináris esetben $1/2$, $1/4, \ldots, 1/2^n, \ldots$. Pl. a bináris 0.1011001 számot mindig 2-vel szorozva az eredmény egész része rendre 1, 0, 1, 1, 0, 0, 1. Ezt használva:

$0.3 \cdot 2 = 0.6 \to 0$
$0.6 \cdot 2 = 1.2 \to 1$
$0.2 \cdot 2 = 0.4 \to 0$
$0.4 \cdot 2 = 0.8 \to 0$
$0.8 \cdot 2 = 1.6 \to 1$
$0.6 \cdot 2 = 1.2 \to 1$

0.3	2
0.6	0
1.2	1
0.4	0

Azaz 0.3 bináris alakja 0.010011, sőt az is látszik, hogy a végtelen bináris alak: $0.0\dot{1}00\dot{1}$.

Example

Hogyan konvertálunk tizedes törtet binárissá? Pl. írjuk fel 0.3 bináris alakjának tizedespont utáni első 6 jegyét!

Megoldás: A tizedespont utáni jegyek jelentése bináris esetben $1/2$, $1/4, \ldots, 1/2^n, \ldots$. Pl. a bináris 0.1011001 számot mindig 2-vel szorozva az eredmény egész része rendre 1, 0, 1, 1, 0, 0, 1. Ezt használva:

$0.3 \cdot 2 = 0.6 \to 0$
$0.6 \cdot 2 = 1.2 \to 1$
$0.2 \cdot 2 = 0.4 \to 0$
$0.4 \cdot 2 = 0.8 \to 0$
$0.8 \cdot 2 = 1.6 \to 1$
$0.6 \cdot 2 = 1.2 \to 1$

0.3	2
0.6	0
1.2	1
0.4	0
0.8	0

Azaz 0.3 bináris alakja 0.010011, sőt az is látszik, hogy a végtelen bináris alak: $0.0\dot{1}00\dot{1}$.

Example

Hogyan konvertálunk tizedes törtet binárissá? Pl. írjuk fel 0.3 bináris alakjának tizedespont utáni első 6 jegyét!

Megoldás: A tizedespont utáni jegyek jelentése bináris esetben $1/2$, $1/4, \ldots, 1/2^n, \ldots$. Pl. a bináris 0.1011001 számot mindig 2-vel szorozva az eredmény egész része rendre 1, 0, 1, 1, 0, 0, 1. Ezt használva:

$0.3 \cdot 2 = 0.6 \rightarrow 0$
$0.6 \cdot 2 = 1.2 \rightarrow 1$
$0.2 \cdot 2 = 0.4 \rightarrow 0$
$0.4 \cdot 2 = 0.8 \rightarrow 0$
$0.8 \cdot 2 = 1.6 \rightarrow 1$
$0.6 \cdot 2 = 1.2 \rightarrow 1$

Azaz 0.3 bináris alakja 0.010011, sőt az is látszik, hogy a végtelen bináris alak: $0.0\dot{1}00\dot{1}$.

0.3	2
0.6	0
1.2	1
0.4	0
0.8	0
1.6	1

Example

Hogyan konvertálunk tizedes törtet binárissá? Pl. írjuk fel 0.3 bináris alakjának tizedespont utáni első 6 jegyét!

Megoldás: A tizedespont utáni jegyek jelentése bináris esetben $1/2$, $1/4, \ldots, 1/2^n, \ldots$. Pl. a bináris 0.1011001 számot mindig 2-vel szorozva az eredmény egész része rendre 1, 0, 1, 1, 0, 0, 1. Ezt használva:

$0.3 \cdot 2 = 0.6 \to 0$
$0.6 \cdot 2 = 1.2 \to 1$
$0.2 \cdot 2 = 0.4 \to 0$
$0.4 \cdot 2 = 0.8 \to 0$
$0.8 \cdot 2 = 1.6 \to 1$
$0.6 \cdot 2 = 1.2 \to 1$

Azaz 0.3 bináris alakja 0.010011, sőt az is látszik, hogy a végtelen bináris alak: $0.0\dot{1}00\dot{1}$.

0.3	2
0.6	0
1.2	1
0.4	0
0.8	0
1.6	1
1.2	1

Example

Hogyan konvertálunk tizedes törtet binárissá? Pl. írjuk fel 0.3 bináris alakjának tizedespont utáni első 6 jegyét!

Megoldás: A tizedespont utáni jegyek jelentése bináris esetben $1/2$, $1/4, \ldots, 1/2^n, \ldots$. Pl. a bináris 0.1011001 számot mindig 2-vel szorozva az eredmény egész része rendre 1, 0, 1, 1, 0, 0, 1. Ezt használva:

$0.3 \cdot 2 = 0.6 \to 0$
$0.6 \cdot 2 = 1.2 \to 1$
$0.2 \cdot 2 = 0.4 \to 0$
$0.4 \cdot 2 = 0.8 \to 0$
$0.8 \cdot 2 = 1.6 \to 1$
$0.6 \cdot 2 = 1.2 \to 1$

Azaz 0.3 bináris alakja 0.010011, sőt az is látszik, hogy a végtelen bináris alak: $0.0\dot{1}00\dot{1}$.

0.3	2
0.6	0
1.2	1
0.4	0
0.8	0
1.6	1
1.2	1

Hexadecimális számrendszer

Hexadecimális (16-os számrendszerbeli) számok:

bin	hex	bin	hex
0000	0	1000	8
0001	1	1001	9
0010	2	1010	A
0011	3	1011	B
0100	4	1100	C
0101	5	1101	D
0110	6	1110	E
0111	7	1111	F

Hexadecimális számrendszer

Hexadecimális (16-os számrendszerbeli) számok:

bin	hex	bin	hex
0000	0	1000	8
0001	1	1001	9
0010	2	1010	A
0011	3	1011	B
0100	4	1100	C
0101	5	1101	D
0110	6	1110	E
0111	7	1111	F

Például $0011\,1100\ 1111\,1010 = 0x3CFA$.

Egyes komplemens számábrázolás

Egyes komplemens számábrázolás *n*-biten: az első bit az előjel.

Egyes komplemens számábrázolás

Egyes komplemens számábrázolás *n*-biten: az első bit az előjel. Az ábrázolható számok tartománya: $-2^{n-1}+1$-től $2^{n-1}-1$-ig.

Egyes komplemens számábrázolás

Egyes komplemens számábrázolás *n*-biten: az első bit az előjel. Az ábrázolható számok tartománya: $-2^{n-1}+1$-től $2^{n-1}-1$-ig.
Például 4 biten: -7-től 7-ig ábrázolhatók a számok.

$1001 \rightarrow -1$

$1100 \rightarrow -4$

$1111 \rightarrow -7$

$1000 \rightarrow -0$

$0000 \rightarrow +0$

Egyes komplemens számábrázolás

Egyes komplemens számábrázolás n-biten: az első bit az előjel. Az ábrázolható számok tartománya: $-2^{n-1}+1$-től $2^{n-1}-1$-ig.
Például 4 biten: -7-től 7-ig ábrázolhatók a számok.
$1001 \to -1$
$1100 \to -4$
$1111 \to -7$
$1000 \to -0$
$0000 \to +0$
Hátránya: van $+0$ és -0.

Kettes komplemens számábrázolás

Kettes komplemens számábrázolás n-biten: előjelbites számábrázolást keresünk, ahol nincs $+0$ és -0.

$$\bar{x} = \begin{cases} x & \text{ha } x \text{ nem negatív,} \\ 2^n - |x| & \text{ha } x \text{ negatív.} \end{cases}$$

Kettes komplemens számábrázolás

Kettes komplemens számábrázolás n-biten: előjelbites számábrázolást keresünk, ahol nincs $+0$ és -0.

$$\bar{x} = \begin{cases} x & \text{ha } x \text{ nem negatív,} \\ 2^n - |x| & \text{ha } x \text{ negatív.} \end{cases}$$

$2^n - |x|$ kiszámítása n-bites szavak közti bitműveletekkel: $|x|$ bitenkénti komplemense $+ 1$, ugyanis
$2^n - |x| = (2^n - 1) - |x| + 1 = 11\ldots1_2 - |x| + 1.$

Kettes komplemens számábrázolás

Kettes komplemens számábrázolás n-biten: előjelbites számábrázolást keresünk, ahol nincs $+0$ és -0.

$$\bar{x} = \begin{cases} x & \text{ha } x \text{ nem negatív,} \\ 2^n - |x| & \text{ha } x \text{ negatív.} \end{cases}$$

$2^n - |x|$ kiszámítása n-bites szavak közti bitműveletekkel: $|x|$ bitenkénti komplemense $+ 1$, ugyanis
$2^n - |x| = (2^n - 1) - |x| + 1 = 11\ldots1_2 - |x| + 1$. Mivel
$|x| = 2^n - (2^n - |x|)$, ezért x értékének meghatározása $\bar{x}$-ből ugyanígy történik, azaz ha az első bit egyes, $|x|$ értéke $= \bar{x}$ komplemense $+ 1$.

Kettes komplemens számábrázolás

Kettes komplemens számábrázolás n-biten: előjelbites számábrázolást keresünk, ahol nincs $+0$ és -0.

$$\bar{x} = \begin{cases} x & \text{ha } x \text{ nem negatív,} \\ 2^n - |x| & \text{ha } x \text{ negatív.} \end{cases}$$

$2^n - |x|$ kiszámítása n-bites szavak közti bitműveletekkel: $|x|$ bitenkénti komplemense $+ 1$, ugyanis
$2^n - |x| = (2^n - 1) - |x| + 1 = 11\ldots1_2 - |x| + 1$. Mivel
$|x| = 2^n - (2^n - |x|)$, ezért x értékének meghatározása $\bar{x}$-ből ugyanígy történik, azaz ha az első bit egyes, $|x|$ értéke $= \bar{x}$ komplemense $+ 1$.
-1 alakja

Kettes komplemens számábrázolás

Kettes komplemens számábrázolás n-biten: előjelbites számábrázolást keresünk, ahol nincs $+0$ és -0.

$$\bar{x} = \begin{cases} x & \text{ha } x \text{ nem negatív,} \\ 2^n - |x| & \text{ha } x \text{ negatív.} \end{cases}$$

$2^n - |x|$ kiszámítása n-bites szavak közti bitműveletekkel: $|x|$ bitenkénti komplemense $+ 1$, ugyanis
$2^n - |x| = (2^n - 1) - |x| + 1 = 11\ldots1_2 - |x| + 1$. Mivel
$|x| = 2^n - (2^n - |x|)$, ezért x értékének meghatározása $\bar{x}$-ből ugyanígy történik, azaz ha az első bit egyes, $|x|$ értéke $= \bar{x}$ komplemense $+ 1$.
-1 alakja $11\ldots11_2$.

Kettes komplemens számábrázolás

Kettes komplemens számábrázolás n-biten: előjelbites számábrázolást keresünk, ahol nincs $+0$ és -0.

$$\bar{x} = \begin{cases} x & \text{ha } x \text{ nem negatív,} \\ 2^n - |x| & \text{ha } x \text{ negatív.} \end{cases}$$

$2^n - |x|$ kiszámítása n-bites szavak közti bitműveletekkel: $|x|$ bitenkénti komplemense $+ 1$, ugyanis
$2^n - |x| = (2^n - 1) - |x| + 1 = 11\ldots1_2 - |x| + 1$. Mivel
$|x| = 2^n - (2^n - |x|)$, ezért x értékének meghatározása $\bar{x}$-ből ugyanígy történik, azaz ha az első bit egyes, $|x|$ értéke $= \bar{x}$ komplemense $+ 1$.
-1 alakja $11\ldots11_2$. -2 alakja

Kettes komplemens számábrázolás

Kettes komplemens számábrázolás n-biten: előjelbites számábrázolást keresünk, ahol nincs $+0$ és -0.

$$\bar{x} = \begin{cases} x & \text{ha } x \text{ nem negatív,} \\ 2^n - |x| & \text{ha } x \text{ negatív.} \end{cases}$$

$2^n - |x|$ kiszámítása n-bites szavak közti bitműveletekkel: $|x|$ bitenkénti komplemense $+ 1$, ugyanis
$2^n - |x| = (2^n - 1) - |x| + 1 = 11\ldots1_2 - |x| + 1$. Mivel
$|x| = 2^n - (2^n - |x|)$, ezért x értékének meghatározása $\bar{x}$-ből ugyanígy történik, azaz ha az első bit egyes, $|x|$ értéke $= \bar{x}$ komplemense $+ 1$.
-1 alakja $11\ldots11_2$. -2 alakja $11\ldots10_2$.

Kettes komplemens számábrázolás

Kettes komplemens számábrázolás n-biten: előjelbites számábrázolást keresünk, ahol nincs $+0$ és -0.

$$\bar{x} = \begin{cases} x & \text{ha } x \text{ nem negatív,} \\ 2^n - |x| & \text{ha } x \text{ negatív.} \end{cases}$$

$2^n - |x|$ kiszámítása n-bites szavak közti bitműveletekkel: $|x|$ bitenkénti komplemense $+ 1$, ugyanis
$2^n - |x| = (2^n - 1) - |x| + 1 = 11\ldots1_2 - |x| + 1$. Mivel
$|x| = 2^n - (2^n - |x|)$, ezért x értékének meghatározása $\bar{x}$-ből ugyanígy történik, azaz ha az első bit egyes, $|x|$ értéke $= \bar{x}$ komplemense $+ 1$.
-1 alakja $11\ldots11_2$. -2 alakja $11\ldots10_2$. -3 alakja

Kettes komplemens számábrázolás

Kettes komplemens számábrázolás n-biten: előjelbites számábrázolást keresünk, ahol nincs $+0$ és -0.

$$\bar{x} = \begin{cases} x & \text{ha } x \text{ nem negatív,} \\ 2^n - |x| & \text{ha } x \text{ negatív.} \end{cases}$$

$2^n - |x|$ kiszámítása n-bites szavak közti bitműveletekkel: $|x|$ bitenkénti komplemense $+ 1$, ugyanis
$2^n - |x| = (2^n - 1) - |x| + 1 = 11\ldots1_2 - |x| + 1$. Mivel
$|x| = 2^n - (2^n - |x|)$, ezért x értékének meghatározása $\bar{x}$-ből ugyanígy történik, azaz ha az első bit egyes, $|x|$ értéke $= \bar{x}$ komplemense $+ 1$.
-1 alakja $11\ldots11_2$. -2 alakja $11\ldots10_2$. -3 alakja $11\ldots01_2$.

Kettes komplemens számábrázolás

Kettes komplemens számábrázolás n-biten: előjelbites számábrázolást keresünk, ahol nincs $+0$ és -0.

$$\bar{x} = \begin{cases} x & \text{ha } x \text{ nem negatív,} \\ 2^n - |x| & \text{ha } x \text{ negatív.} \end{cases}$$

$2^n - |x|$ kiszámítása n-bites szavak közti bitműveletekkel: $|x|$ bitenkénti komplemense $+ 1$, ugyanis
$2^n - |x| = (2^n - 1) - |x| + 1 = 11\ldots1_2 - |x| + 1$. Mivel
$|x| = 2^n - (2^n - |x|)$, ezért x értékének meghatározása $\bar{x}$-ből ugyanígy történik, azaz ha az első bit egyes, $|x|$ értéke $= \bar{x}$ komplemense $+ 1$.
-1 alakja $11\ldots11_2$. -2 alakja $11\ldots10_2$. -3 alakja $11\ldots01_2$.

Example

legyen $n = 4$, $x = -5$: $-5 \rightarrow$

Kettes komplemens számábrázolás

Kettes komplemens számábrázolás n-biten: előjelbites számábrázolást keresünk, ahol nincs $+0$ és -0.

$$\bar{x} = \begin{cases} x & \text{ha } x \text{ nem negatív,} \\ 2^n - |x| & \text{ha } x \text{ negatív.} \end{cases}$$

$2^n - |x|$ kiszámítása n-bites szavak közti bitműveletekkel: $|x|$ bitenkénti komplemense $+ 1$, ugyanis
$2^n - |x| = (2^n - 1) - |x| + 1 = 11\ldots1_2 - |x| + 1$. Mivel
$|x| = 2^n - (2^n - |x|)$, ezért x értékének meghatározása $\bar{x}$-ből ugyanígy történik, azaz ha az első bit egyes, $|x|$ értéke $= \bar{x}$ komplemense $+ 1$.
-1 alakja $11\ldots11_2$. -2 alakja $11\ldots10_2$. -3 alakja $11\ldots01_2$.

Example

legyen $n = 4$, $x = -5$: $-5 \rightarrow \bar{x} = 16 - 5$

Kettes komplemens számábrázolás

Kettes komplemens számábrázolás n-biten: előjelbites számábrázolást keresünk, ahol nincs $+0$ és -0.

$$\bar{x} = \begin{cases} x & \text{ha } x \text{ nem negatív,} \\ 2^n - |x| & \text{ha } x \text{ negatív.} \end{cases}$$

$2^n - |x|$ kiszámítása n-bites szavak közti bitműveletekkel: $|x|$ bitenkénti komplemense $+ 1$, ugyanis
$2^n - |x| = (2^n - 1) - |x| + 1 = 11\ldots1_2 - |x| + 1$. Mivel
$|x| = 2^n - (2^n - |x|)$, ezért x értékének meghatározása $\bar{x}$-ből ugyanígy történik, azaz ha az első bit egyes, $|x|$ értéke $= \bar{x}$ komplemense $+ 1$.
-1 alakja $11\ldots11_2$. -2 alakja $11\ldots10_2$. -3 alakja $11\ldots01_2$.

Example

legyen $n = 4$, $x = -5$: $-5 \to \bar{x} = 16 - 5 = 11$

Kettes komplemens számábrázolás

Kettes komplemens számábrázolás n-biten: előjelbites számábrázolást keresünk, ahol nincs $+0$ és -0.

$$\bar{x} = \begin{cases} x & \text{ha } x \text{ nem negatív,} \\ 2^n - |x| & \text{ha } x \text{ negatív.} \end{cases}$$

$2^n - |x|$ kiszámítása n-bites szavak közti bitműveletekkel: $|x|$ bitenkénti komplemense $+ 1$, ugyanis
$2^n - |x| = (2^n - 1) - |x| + 1 = 11\ldots1_2 - |x| + 1$. Mivel
$|x| = 2^n - (2^n - |x|)$, ezért x értékének meghatározása $\bar{x}$-ből ugyanígy történik, azaz ha az első bit egyes, $|x|$ értéke $= \bar{x}$ komplemense $+ 1$.
-1 alakja $11\ldots11_2$. -2 alakja $11\ldots10_2$. -3 alakja $11\ldots01_2$.

Example

legyen $n = 4$, $x = -5$: $-5 \to \bar{x} = 16 - 5 = 11 = 1011_2$

Kettes komplemens számábrázolás

Kettes komplemens számábrázolás n-biten: előjelbites számábrázolást keresünk, ahol nincs $+0$ és -0.

$$\bar{x} = \begin{cases} x & \text{ha } x \text{ nem negatív,} \\ 2^n - |x| & \text{ha } x \text{ negatív.} \end{cases}$$

$2^n - |x|$ kiszámítása n-bites szavak közti bitműveletekkel: $|x|$ bitenkénti komplemense + 1, ugyanis
$2^n - |x| = (2^n - 1) - |x| + 1 = 11\ldots1_2 - |x| + 1$. Mivel
$|x| = 2^n - (2^n - |x|)$, ezért x értékének meghatározása $\bar{x}$-ből ugyanígy történik, azaz ha az első bit egyes, $|x|$ értéke $= \bar{x}$ komplemense $+1$.
-1 alakja $11\ldots11_2$. -2 alakja $11\ldots10_2$. -3 alakja $11\ldots01_2$.

Example

legyen $n = 4$, $x = -5$: $-5 \to \bar{x} = 16 - 5 = 11 = 1011_2$
bitműveletekkel: $x = -5 \to |x| = 5$

Kettes komplemens számábrázolás

Kettes komplemens számábrázolás n-biten: előjelbites számábrázolást keresünk, ahol nincs $+0$ és -0.

$$\bar{x} = \begin{cases} x & \text{ha } x \text{ nem negatív,} \\ 2^n - |x| & \text{ha } x \text{ negatív.} \end{cases}$$

$2^n - |x|$ kiszámítása n-bites szavak közti bitműveletekkel: $|x|$ bitenkénti komplemense $+ 1$, ugyanis
$2^n - |x| = (2^n - 1) - |x| + 1 = 11\ldots1_2 - |x| + 1$. Mivel
$|x| = 2^n - (2^n - |x|)$, ezért x értékének meghatározása $\bar{x}$-ből ugyanígy történik, azaz ha az első bit egyes, $|x|$ értéke $= \bar{x}$ komplemense $+ 1$.
-1 alakja $11\ldots11_2$. -2 alakja $11\ldots10_2$. -3 alakja $11\ldots01_2$.

Example

legyen $n = 4$, $x = -5$: $-5 \to \bar{x} = 16 - 5 = 11 = 1011_2$
bitműveletekkel: $x = -5 \to |x| = 5 \to 0101_2$

Kettes komplemens számábrázolás

Kettes komplemens számábrázolás n-biten: előjelbites számábrázolást keresünk, ahol nincs $+0$ és -0.

$$\bar{x} = \begin{cases} x & \text{ha } x \text{ nem negatív,} \\ 2^n - |x| & \text{ha } x \text{ negatív.} \end{cases}$$

$2^n - |x|$ kiszámítása n-bites szavak közti bitműveletekkel: $|x|$ bitenkénti komplemense $+ 1$, ugyanis
$2^n - |x| = (2^n - 1) - |x| + 1 = 11\ldots1_2 - |x| + 1$. Mivel
$|x| = 2^n - (2^n - |x|)$, ezért x értékének meghatározása $\bar{x}$-ből ugyanígy történik, azaz ha az első bit egyes, $|x|$ értéke $= \bar{x}$ komplemense $+ 1$.
-1 alakja $11\ldots11_2$. -2 alakja $11\ldots10_2$. -3 alakja $11\ldots01_2$.

Example

legyen $n = 4$, $x = -5$: $-5 \to \bar{x} = 16 - 5 = 11 = 1011_2$
bitműveletekkel: $x = -5 \to |x| = 5 \to 0101_2 \to \bar{x} = 1010_2 + 1_2 = 1011_2$

Kettes komplemens számábrázolás

Kettes komplemens számábrázolás n-biten: előjelbites számábrázolást keresünk, ahol nincs $+0$ és -0.

$$\bar{x} = \begin{cases} x & \text{ha } x \text{ nem negatív,} \\ 2^n - |x| & \text{ha } x \text{ negatív.} \end{cases}$$

$2^n - |x|$ kiszámítása n-bites szavak közti bitműveletekkel: $|x|$ bitenkénti komplemense $+ 1$, ugyanis
$2^n - |x| = (2^n - 1) - |x| + 1 = 11\ldots1_2 - |x| + 1$. Mivel
$|x| = 2^n - (2^n - |x|)$, ezért x értékének meghatározása $\bar{x}$-ből ugyanígy történik, azaz ha az első bit egyes, $|x|$ értéke $= \bar{x}$ komplemense $+ 1$.
-1 alakja $11\ldots11_2$. -2 alakja $11\ldots10_2$. -3 alakja $11\ldots01_2$.

Example

legyen $n = 4$, $x = -5$: $-5 \to \bar{x} = 16 - 5 = 11 = 1011_2$
bitműveletekkel: $x = -5 \to |x| = 5 \to 0101_2 \to \bar{x} = 1010_2 + 1_2 = 1011_2$
Visszaalakítás: $\bar{x} = 1011_2$

Kettes komplemens számábrázolás

Kettes komplemens számábrázolás n-biten: előjelbites számábrázolást keresünk, ahol nincs $+0$ és -0.

$$\bar{x} = \begin{cases} x & \text{ha } x \text{ nem negatív,} \\ 2^n - |x| & \text{ha } x \text{ negatív.} \end{cases}$$

$2^n - |x|$ kiszámítása n-bites szavak közti bitműveletekkel: $|x|$ bitenkénti komplemense $+ 1$, ugyanis
$2^n - |x| = (2^n - 1) - |x| + 1 = 11\ldots1_2 - |x| + 1$. Mivel
$|x| = 2^n - (2^n - |x|)$, ezért x értékének meghatározása $\bar{x}$-ből ugyanígy történik, azaz ha az első bit egyes, $|x|$ értéke $= \bar{x}$ komplemense $+ 1$.
-1 alakja $11\ldots11_2$. -2 alakja $11\ldots10_2$. -3 alakja $11\ldots01_2$.

Example

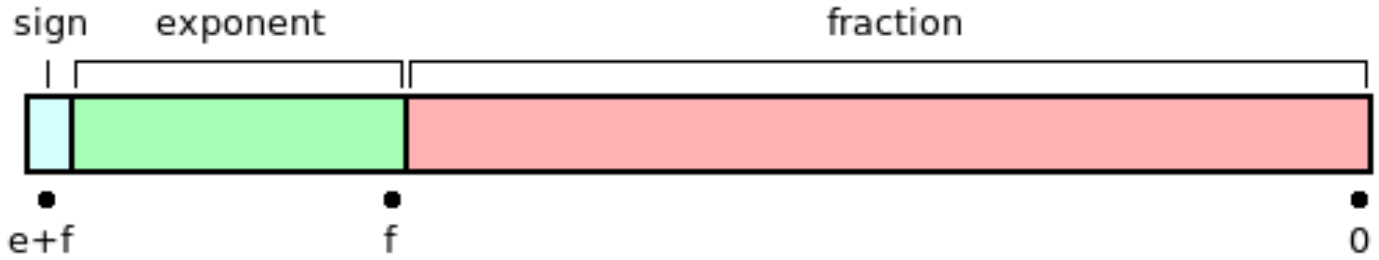
legyen $n = 4$, $x = -5$: $-5 \to \bar{x} = 16 - 5 = 11 = 1011_2$
bitműveletekkel: $x = -5 \to |x| = 5 \to 0101_2 \to \bar{x} = 1010_2 + 1_2 = 1011_2$
Visszaalakítás: $\bar{x} = 1011_2 \to x = 0100_2 + 1_2 = 0101_2 = 5$.

Lebegőpontos számábrázolás

IEEE 754-2008, ISO/IEC/IEEE 60559:2011

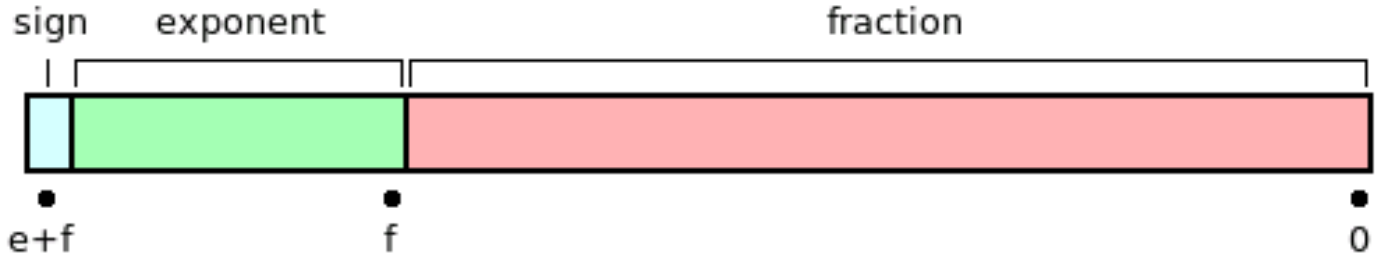
Lebegőpontos számábrázolás

IEEE 754-2008, ISO/IEC/IEEE 60559:2011



Lebegőpontos számábrázolás

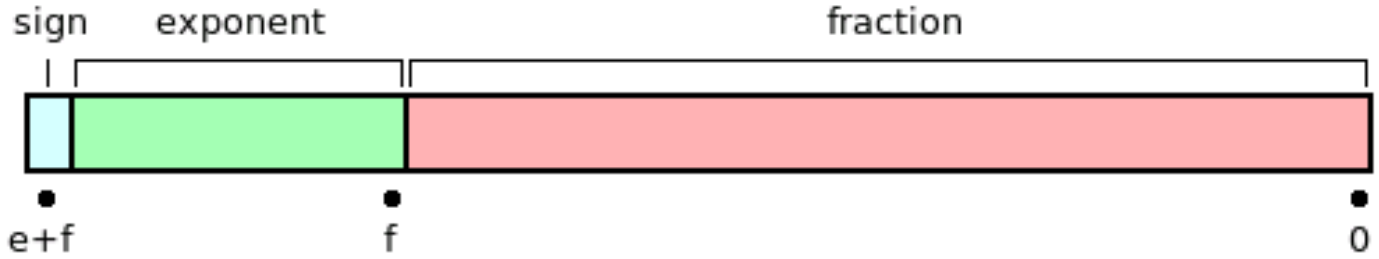
IEEE 754-2008, ISO/IEC/IEEE 60559:2011



	s=előjel	*e*=kitevő	mantissza	összesen	eltolás (bias)
szimpla	1	8	23	32	127 (01111111)
dupla	1	11	52	64	1023 (01111111111)

Lebegőpontos számábrázolás

IEEE 754-2008, ISO/IEC/IEEE 60559:2011

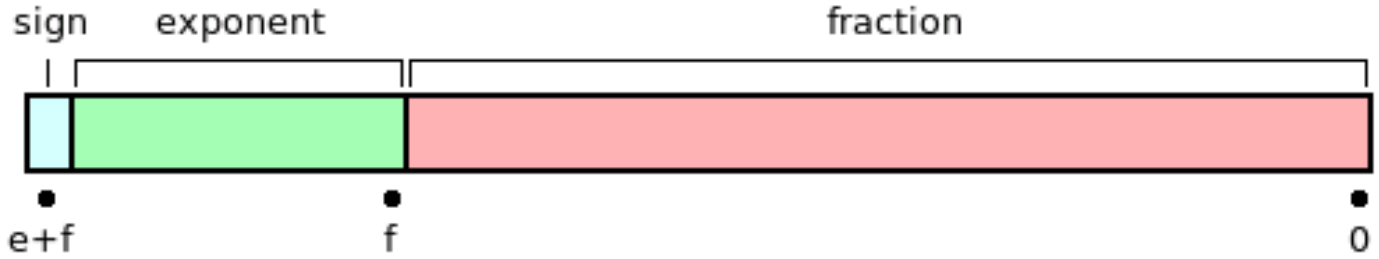


	s=előjel	e=kitevő	mantissza	összesen	eltolás (bias)
szimpla	1	8	23	32	127 (01111111)
dupla	1	11	52	64	1023 (01111111111)

szimpla: $(-1)^s(1.b_{22}b_{21}\ldots b_0)_2 \cdot 2^{e-127} = \left(1+\sum_{i=1}^{23} b_{23-i}2^{-i}\right)\cdot 2^{e-127}$

Lebegőpontos számábrázolás

IEEE 754-2008, ISO/IEC/IEEE 60559:2011



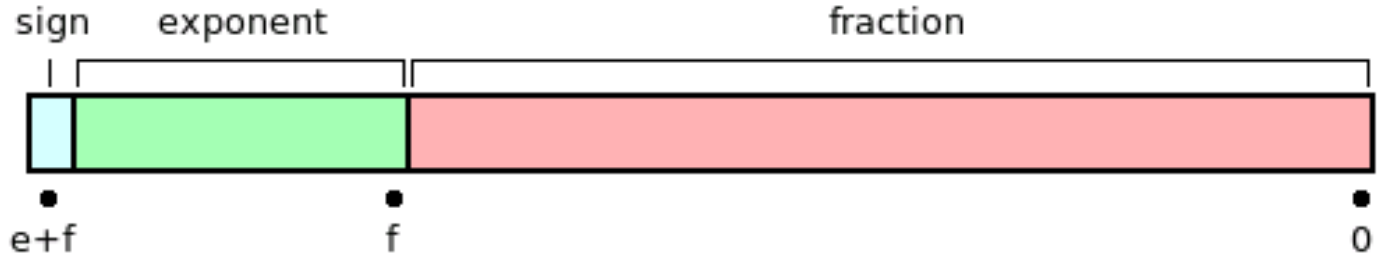
	s=előjel	e=kitevő	mantissza	összesen	eltolás (bias)
szimpla	1	8	23	32	127 (01111111)
dupla	1	11	52	64	1023 (01111111111)

szimpla: $(-1)^s(1.b_{22}b_{21}\ldots b_0)_2 \cdot 2^{e-127} = \left(1+\sum_{i=1}^{23} b_{23-i}2^{-i}\right)\cdot 2^{e-127}$

dupla: $(-1)^s(1.b_{51}b_{50}\ldots b_0)_2 \cdot 2^{e-1023} = \left(1+\sum_{i=1}^{52} b_{52-i}2^{-i}\right)\cdot 2^{e-1023}$

Lebegőpontos számábrázolás

IEEE 754-2008, ISO/IEC/IEEE 60559:2011



	s=előjel	e=kitevő	mantissza	összesen	eltolás (bias)
szimpla	1	8	23	32	127 (01111111)
dupla	1	11	52	64	1023 (01111111111)

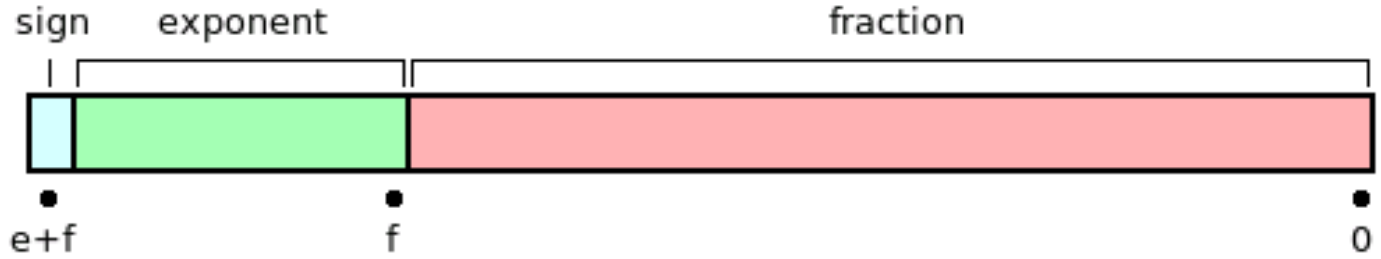
szimpla: $(-1)^s(1.b_{22}b_{21}\ldots b_0)_2 \cdot 2^{e-127} = \left(1+\sum_{i=1}^{23} b_{23-i}2^{-i}\right)\cdot 2^{e-127}$

dupla: $(-1)^s(1.b_{51}b_{50}\ldots b_0)_2 \cdot 2^{e-1023} = \left(1+\sum_{i=1}^{52} b_{52-i}2^{-i}\right)\cdot 2^{e-1023}$

Például dupla pontosság esetén $2^{52} = 4\,503\,599\,627\,370\,496$ és $2^{53} = 9\,007\,199\,254\,740\,992$ között csak az egészek vannak pontosan reprezentálva.

Lebegőpontos számábrázolás

IEEE 754-2008, ISO/IEC/IEEE 60559:2011



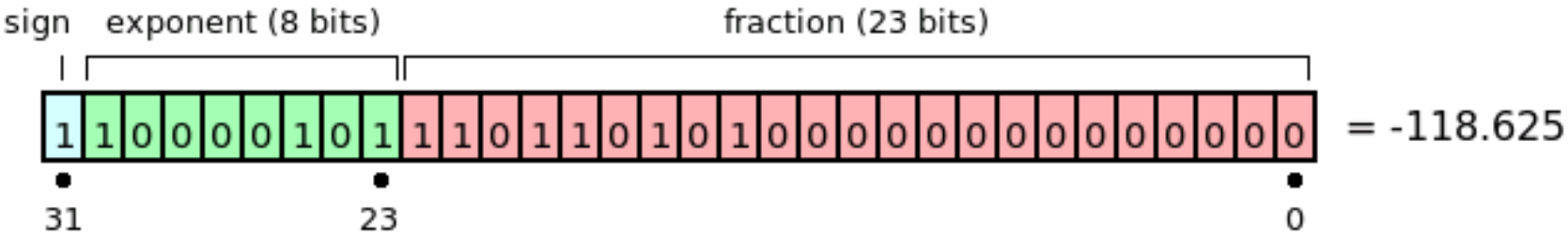
	s=előjel	e=kitevő	mantissza	összesen	eltolás (bias)
szimpla	1	8	23	32	127 (01111111)
dupla	1	11	52	64	1023 (01111111111)

szimpla: $(-1)^s(1.b_{22}b_{21}\ldots b_0)_2 \cdot 2^{e-127} = \left(1+\sum_{i=1}^{23} b_{23-i}2^{-i}\right)\cdot 2^{e-127}$

dupla: $(-1)^s(1.b_{51}b_{50}\ldots b_0)_2 \cdot 2^{e-1023} = \left(1+\sum_{i=1}^{52} b_{52-i}2^{-i}\right)\cdot 2^{e-1023}$

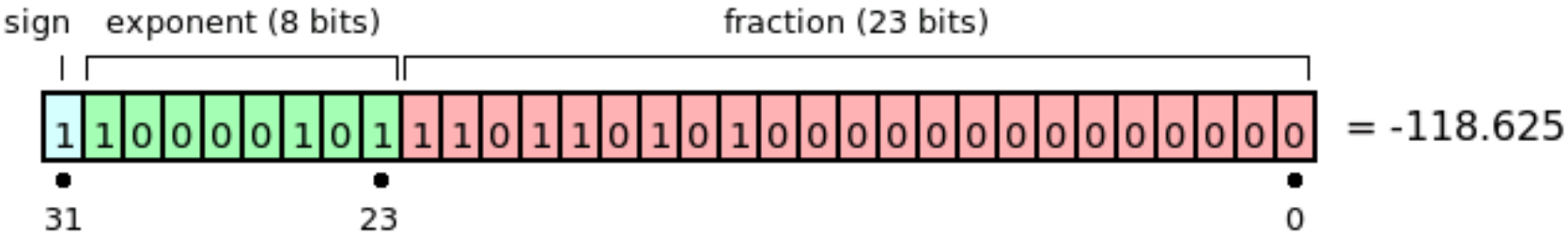
Például dupla pontosság esetén $2^{52} = 4\,503\,599\,627\,370\,496$ és $2^{53} = 9\,007\,199\,254\,740\,992$ között csak az egészek vannak pontosan reprezentálva. 2^{53} és 2^{54} között csak a páros egészek...

Előjel, kitevő, mantissza



előjel (sign) 1 $\rightarrow$ negatív

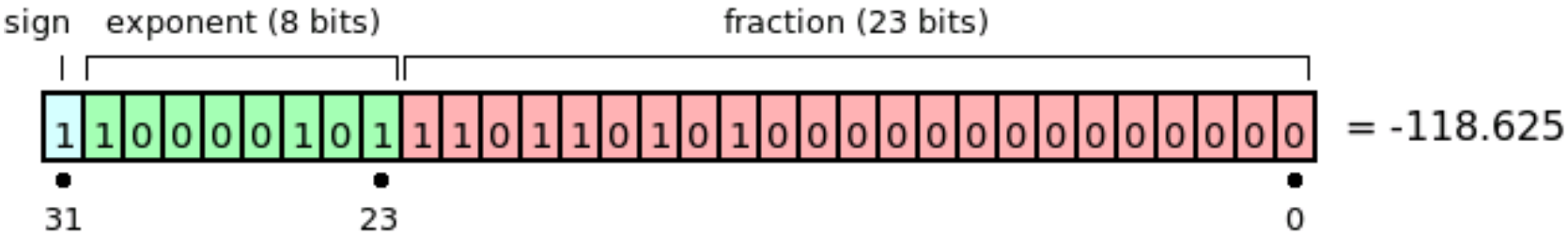
Előjel, kitevő, mantissza



előjel (sign) 1 $\rightarrow$ negatív

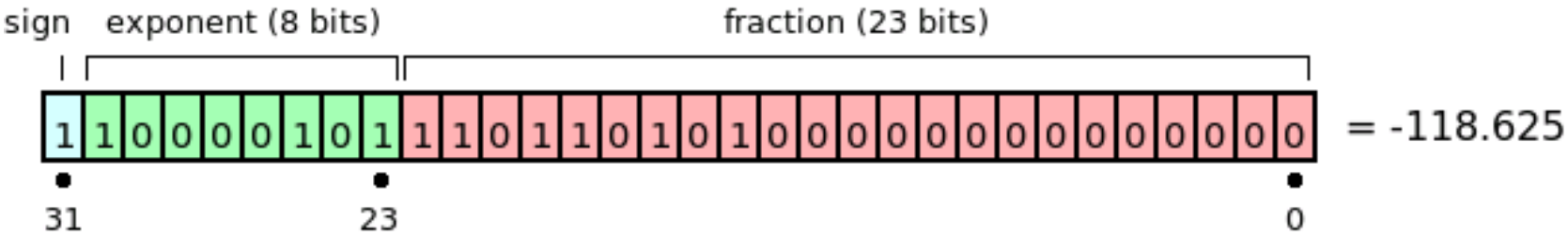
kitevő (exponent) $10000101_2 - 01111111_2 = 00000110_2$, azaz 6

Előjel, kitevő, mantissza



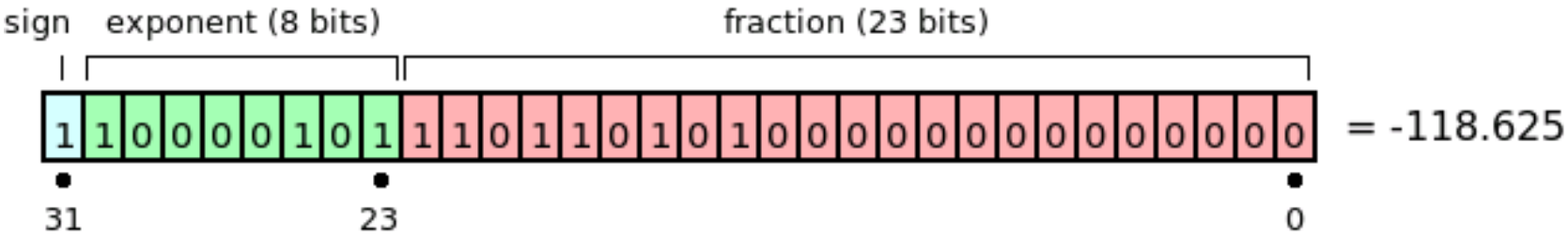
előjel (sign) 1 $\rightarrow$ negatív
kitevő (exponent) $10000101_2 - 01111111_2 = 00000110_2$, azaz 6
mantissa (1.significand) 1.110110101_2,

Előjel, kitevő, mantissza



előjel (sign) 1 $\rightarrow$ negatív
kitevő (exponent) $10000101_2 - 01111111_2 = 00000110_2$, azaz 6
mantissa (1.significand) 1.110110101_2,
így a szám -1110110.101_2,

Előjel, kitevő, mantissza



előjel (sign) 1 $\rightarrow$ negatív
kitevő (exponent) $10000101_2 - 01111111_2 = 00000110_2$, azaz 6
mantissa (1.significand) 1.110110101_2,
így a szám -1110110.101_2, azaz -118.625

Már majdnem csak történelem

1. ISO-8859-1 Latin1 (West European)

Már majdnem csak történelem

1. ISO-8859-1 Latin1 (West European)
2. ISO-8859-2 Latin2 (East European)

Már majdnem csak történelem

1. ISO-8859-1 Latin1 (West European)
2. ISO-8859-2 Latin2 (East European)
3. ISO-8859-3 Latin3 (South European)

Már majdnem csak történelem

1. ISO-8859-1 Latin1 (West European)
2. ISO-8859-2 Latin2 (East European)
3. ISO-8859-3 Latin3 (South European)
4. ISO-8859-4 Latin4 (North European)

Már majdnem csak történelem

1. ISO-8859-1 Latin1 (West European)
2. ISO-8859-2 Latin2 (East European)
3. ISO-8859-3 Latin3 (South European)
4. ISO-8859-4 Latin4 (North European)
5. ISO-8859-5 Cyrillic

Már majdnem csak történelem

1. ISO-8859-1 Latin1 (West European)
2. ISO-8859-2 Latin2 (East European)
3. ISO-8859-3 Latin3 (South European)
4. ISO-8859-4 Latin4 (North European)
5. ISO-8859-5 Cyrillic
6. ISO-8859-6 Arabic

Már majdnem csak történelem

1. ISO-8859-1 Latin1 (West European)
2. ISO-8859-2 Latin2 (East European)
3. ISO-8859-3 Latin3 (South European)
4. ISO-8859-4 Latin4 (North European)
5. ISO-8859-5 Cyrillic
6. ISO-8859-6 Arabic
7. ISO-8859-7 Greek

Már majdnem csak történelem

1. ISO-8859-1 Latin1 (West European)
2. ISO-8859-2 Latin2 (East European)
3. ISO-8859-3 Latin3 (South European)
4. ISO-8859-4 Latin4 (North European)
5. ISO-8859-5 Cyrillic
6. ISO-8859-6 Arabic
7. ISO-8859-7 Greek
8. ISO-8859-8 Hebrew

Már majdnem csak történelem

1. ISO-8859-1 Latin1 (West European)
2. ISO-8859-2 Latin2 (East European)
3. ISO-8859-3 Latin3 (South European)
4. ISO-8859-4 Latin4 (North European)
5. ISO-8859-5 Cyrillic
6. ISO-8859-6 Arabic
7. ISO-8859-7 Greek
8. ISO-8859-8 Hebrew
9. ISO-8859-9 Latin5 (Turkish)

Már majdnem csak történelem

1. ISO-8859-1 Latin1 (West European)
2. ISO-8859-2 Latin2 (East European)
3. ISO-8859-3 Latin3 (South European)
4. ISO-8859-4 Latin4 (North European)
5. ISO-8859-5 Cyrillic
6. ISO-8859-6 Arabic
7. ISO-8859-7 Greek
8. ISO-8859-8 Hebrew
9. ISO-8859-9 Latin5 (Turkish)
10. ISO-8859-10 Latin6 (Nordic)

Már majdnem csak történelem

ISO-8859-2, Microsoft CP1250 (Windows Latin2), CP852 (DOSLatin2)

ISO-8859-1	C1	Á	U+00C1	LATIN CAPITAL LETTER A WITH ACUTE
ISO-8859-1	E1	á	U+00E1	LATIN SMALL LETTER A WITH ACUTE

Már majdnem csak történelem

ISO-8859-2, Microsoft CP1250 (Windows Latin2), CP852 (DOSLatin2)

ISO-8859-1	C1	Á	U+00C1	LATIN CAPITAL LETTER A WITH ACUTE
ISO-8859-1	E1	á	U+00E1	LATIN SMALL LETTER A WITH ACUTE
ISO-8859-1	D5	Õ	U+00D5	LATIN CAPITAL LETTER O WITH TILDE
ISO-8859-1	DB	Û	U+00DB	LATIN CAPITAL LETTER U WITH CIRCUMFLEX
ISO-8859-1	F5	õ	U+00F5	LATIN SMALL LETTER O WITH TILDE
ISO-8859-1	FB	û	U+00FB	LATIN SMALL LETTER U WITH CIRCUMFLEX

Már majdnem csak történelem

ISO-8859-2, Microsoft CP1250 (Windows Latin2), CP852 (DOSLatin2)

ISO-8859-1	C1	Á	U+00C1	LATIN CAPITAL LETTER A WITH ACUTE
ISO-8859-1	E1	á	U+00E1	LATIN SMALL LETTER A WITH ACUTE
ISO-8859-1	D5	Õ	U+00D5	LATIN CAPITAL LETTER O WITH TILDE
ISO-8859-1	DB	Û	U+00DB	LATIN CAPITAL LETTER U WITH CIRCUMFLEX
ISO-8859-1	F5	õ	U+00F5	LATIN SMALL LETTER O WITH TILDE
ISO-8859-1	FB	û	U+00FB	LATIN SMALL LETTER U WITH CIRCUMFLEX
ISO-8859-2	D5	Ő	U+0150	LATIN CAPITAL LETTER O WITH DOUBLE ACU
ISO-8859-2	DB	Ű	U+0170	LATIN CAPITAL LETTER U WITH DOUBLE ACU
ISO-8859-2	F5	ő	U+0151	LATIN SMALL LETTER O WITH DOUBLE ACUTE
ISO-8859-2	FB	ű	U+0171	LATIN SMALL LETTER U WITH DOUBLE ACUTE

Már majdnem csak történelem

ISO-8859-2, Microsoft CP1250 (Windows Latin2), CP852 (DOSLatin2)

ISO-8859-1	C1	Á	U+00C1	LATIN CAPITAL LETTER A WITH ACUTE
ISO-8859-1	E1	á	U+00E1	LATIN SMALL LETTER A WITH ACUTE
ISO-8859-1	D5	Õ	U+00D5	LATIN CAPITAL LETTER O WITH TILDE
ISO-8859-1	DB	Û	U+00DB	LATIN CAPITAL LETTER U WITH CIRCUMFLEX
ISO-8859-1	F5	õ	U+00F5	LATIN SMALL LETTER O WITH TILDE
ISO-8859-1	FB	û	U+00FB	LATIN SMALL LETTER U WITH CIRCUMFLEX
ISO-8859-2	D5	Ő	U+0150	LATIN CAPITAL LETTER O WITH DOUBLE ACU
ISO-8859-2	DB	Ű	U+0170	LATIN CAPITAL LETTER U WITH DOUBLE ACU
ISO-8859-2	F5	ő	U+0151	LATIN SMALL LETTER O WITH DOUBLE ACUTE
ISO-8859-2	FB	ű	U+0171	LATIN SMALL LETTER U WITH DOUBLE ACUTE
CP1250	82	‚	U+201A	SINGLE LOW-9 QUOTATION MARK
CP1250	84	„	U+201E	DOUBLE LOW-9 QUOTATION MARK
CP1250	85	…	U+2026	HORIZONTAL ELLIPSIS
CP1250	91	‘	U+2018	LEFT SINGLE QUOTATION MARK
CP1250	92	’	U+2019	RIGHT SINGLE QUOTATION MARK
CP1250	93	“	U+201C	LEFT DOUBLE QUOTATION MARK
CP1250	94	”	U+201D	RIGHT DOUBLE QUOTATION MARK
CP1250	96	–	U+2013	EN DASH
CP1250	97	—	U+2014	EM DASH

Latin tartományok

- U+0000 - U+007F ASCII

Latin tartományok

- U+0000 - U+007F ASCII
- U+0080 - U+00FF Latin-1

Latin tartományok

- U+0000 - U+007F ASCII
- U+0080 - U+00FF Latin-1
- U+0100 - U+017F Latin Extended-A (latin1, magyar ő, ű)

Latin tartományok

- U+0000 - U+007F ASCII
- U+0080 - U+00FF Latin-1
- U+0100 - U+017F Latin Extended-A (latin1, magyar ő, ű)
- U+0180 - U+024F Latin Extended-B

Latin tartományok

- U+0000 - U+007F ASCII
- U+0080 - U+00FF Latin-1
- U+0100 - U+017F Latin Extended-A (latin1, magyar ő, ű)
- U+0180 - U+024F Latin Extended-B
- U+1E00 - U+1EFF Latin Extended Additional

UTF – Unicode Transformation Format

- UTF-8 minden karakter kódja 8, 16, 24 vagy 32-bites.

UTF – Unicode Transformation Format

- UTF-8 minden karakter kódja 8, 16, 24 vagy 32-bites.
- UTF-16 minden karakter kódja 16 vagy 32-bites.

UTF – Unicode Transformation Format

- UTF-8 minden karakter kódja 8, 16, 24 vagy 32-bites.
- UTF-16 minden karakter kódja 16 vagy 32-bites.
- UTF-32 minden karakter 32-bites.

UTF-8

Unicode		UTF-8	a karakter hivatalos neve
U+0020		20	SPACE
U+0030	0	30	DIGIT ZERO
U+0040	@	40	COMMERCIAL AT
U+0041	A	41	LATIN CAPITAL LETTER A
U+0061	a	61	LATIN SMALL LETTER A

UTF-8

Unicode		UTF-8	a karakter hivatalos neve
U+0020		20	SPACE
U+0030	0	30	DIGIT ZERO
U+0040	@	40	COMMERCIAL AT
U+0041	A	41	LATIN CAPITAL LETTER A
U+0061	a	61	LATIN SMALL LETTER A
U+00C1	Á	c3 81	LATIN CAPITAL LETTER A WITH ACUTE
U+00C9	É	c3 89	LATIN CAPITAL LETTER E WITH ACUTE
U+00CD	Í	c3 8d	LATIN CAPITAL LETTER I WITH ACUTE
U+00D3	Ó	c3 93	LATIN CAPITAL LETTER O WITH ACUTE
U+00D6	Ö	c3 96	LATIN CAPITAL LETTER O WITH DIAERESIS
U+00DA	Ú	c3 9a	LATIN CAPITAL LETTER U WITH ACUTE
U+00DC	Ü	c3 9c	LATIN CAPITAL LETTER U WITH DIAERESIS
U+00E1	á	c3 a1	LATIN SMALL LETTER A WITH ACUTE
U+00E9	é	c3 a9	LATIN SMALL LETTER E WITH ACUTE
U+00ED	í	c3 ad	LATIN SMALL LETTER I WITH ACUTE
U+00F3	ó	c3 b3	LATIN SMALL LETTER O WITH ACUTE
U+00F6	ö	c3 b6	LATIN SMALL LETTER O WITH DIAERESIS
U+00FA	ú	c3 ba	LATIN SMALL LETTER U WITH ACUTE
U+00FC	ü	c3 bc	LATIN SMALL LETTER U WITH DIAERESIS

UTF-8

Unicode		UTF-8	a karakter hivatalos neve
U+0020		20	SPACE
U+0030	0	30	DIGIT ZERO
U+0040	@	40	COMMERCIAL AT
U+0041	A	41	LATIN CAPITAL LETTER A
U+0061	a	61	LATIN SMALL LETTER A
U+00C1	Á	c3 81	LATIN CAPITAL LETTER A WITH ACUTE
U+00C9	É	c3 89	LATIN CAPITAL LETTER E WITH ACUTE
U+00CD	Í	c3 8d	LATIN CAPITAL LETTER I WITH ACUTE
U+00D3	Ó	c3 93	LATIN CAPITAL LETTER O WITH ACUTE
U+00D6	Ö	c3 96	LATIN CAPITAL LETTER O WITH DIAERESIS
U+00DA	Ú	c3 9a	LATIN CAPITAL LETTER U WITH ACUTE
U+00DC	Ü	c3 9c	LATIN CAPITAL LETTER U WITH DIAERESIS
U+00E1	á	c3 a1	LATIN SMALL LETTER A WITH ACUTE
U+00E9	é	c3 a9	LATIN SMALL LETTER E WITH ACUTE
U+00ED	í	c3 ad	LATIN SMALL LETTER I WITH ACUTE
U+00F3	ó	c3 b3	LATIN SMALL LETTER O WITH ACUTE
U+00F6	ö	c3 b6	LATIN SMALL LETTER O WITH DIAERESIS
U+00FA	ú	c3 ba	LATIN SMALL LETTER U WITH ACUTE
U+00FC	ü	c3 bc	LATIN SMALL LETTER U WITH DIAERESIS
U+0150	Ő	c5 90	LATIN CAPITAL LETTER O WITH DOUBLE ACUTE
U+0151	ő	c5 91	LATIN SMALL LETTER O WITH DOUBLE ACUTE

UTF-8

Unicode		UTF-8	a karakter hivatalos neve
U+0020		20	SPACE
U+0030	0	30	DIGIT ZERO
U+0040	@	40	COMMERCIAL AT
U+0041	A	41	LATIN CAPITAL LETTER A
U+0061	a	61	LATIN SMALL LETTER A
U+00C1	Á	c3 81	LATIN CAPITAL LETTER A WITH ACUTE
U+00C9	É	c3 89	LATIN CAPITAL LETTER E WITH ACUTE
U+00CD	Í	c3 8d	LATIN CAPITAL LETTER I WITH ACUTE
U+00D3	Ó	c3 93	LATIN CAPITAL LETTER O WITH ACUTE
U+00D6	Ö	c3 96	LATIN CAPITAL LETTER O WITH DIAERESIS
U+00DA	Ú	c3 9a	LATIN CAPITAL LETTER U WITH ACUTE
U+00DC	Ü	c3 9c	LATIN CAPITAL LETTER U WITH DIAERESIS
U+00E1	á	c3 a1	LATIN SMALL LETTER A WITH ACUTE
U+00E9	é	c3 a9	LATIN SMALL LETTER E WITH ACUTE
U+00ED	í	c3 ad	LATIN SMALL LETTER I WITH ACUTE
U+00F3	ó	c3 b3	LATIN SMALL LETTER O WITH ACUTE
U+00F6	ö	c3 b6	LATIN SMALL LETTER O WITH DIAERESIS
U+00FA	ú	c3 ba	LATIN SMALL LETTER U WITH ACUTE
U+00FC	ü	c3 bc	LATIN SMALL LETTER U WITH DIAERESIS
U+0150	Ő	c5 90	LATIN CAPITAL LETTER O WITH DOUBLE ACUTE
U+0151	ő	c5 91	LATIN SMALL LETTER O WITH DOUBLE ACUTE
U+0170	Ű	c5 b0	LATIN CAPITAL LETTER U WITH DOUBLE ACUTE
U+0171	ű	c5 b1	LATIN SMALL LETTER U WITH DOUBLE ACUTE

UTF-8

Kódtartomány (darab)	bináris alak	UTF-8

UTF-8

Kódtartomány (darab)	bináris alak	UTF-8
000000-00007F (128)	0zzzzzzz	0zzzzzzz
000080-0007FF (1920)	00000yyy yyzzzzzz	110yyyyy 10zzzzzz
000800-00FFFF (63488)	xxxxyyyy yyzzzzzz	1110xxxx 10yyyyyy 10zzzzzz
010000-10FFFF (1048576)	000wwwxx xxxxyyyy yyzzzzzz	11110www 10xxxxxx 10yyyyyy 10zzz

UTF-8

Kódtartomány (darab)	bináris alak	UTF-8
000000-00007F (128)	0zzzzzzz	0zzzzzzz
000080-0007FF (1920)	00000yyy yyzzzzzz	110yyyyy 10zzzzzz
000800-00FFFF (63488)	xxxxyyyy yyzzzzzz	1110xxxx 10yyyyyy 10zzzzzz
010000-10FFFF (1048576)	000wwwxx xxxxyyyy yyzzzzzz	11110www 10xxxxxx 10yyyyyy 10zzz

Á 00C1

UTF-8

Kódtartomány (darab)	bináris alak	UTF-8
000000-00007F (128)	0zzzzzzz	0zzzzzzz
000080-0007FF (1920)	00000yyy yyzzzzzz	110yyyyy 10zzzzzz
000800-00FFFF (63488)	xxxxyyyy yyzzzzzz	1110xxxx 10yyyyyy 10zzzzzz
010000-10FFFF (1048576)	000wwwxx xxxxyyyy yyzzzzzz	11110www 10xxxxxx 10yyyyyy 10zzz

Á 00C1→1100 0001

UTF-8

Kódtartomány (darab)	bináris alak	UTF-8
000000-00007F (128)	0zzzzzzz	0zzzzzzz
000080-0007FF (1920)	00000yyy yyzzzzzz	110yyyyy 10zzzzzz
000800-00FFFF (63488)	xxxxyyyy yyzzzzzz	1110xxxx 10yyyyyy 10zzzzzz
010000-10FFFF (1048576)	000wwwxx xxxxyyyy yyzzzzzz	11110www 10xxxxxx 10yyyyyy 10zzz

Á 00C1→1100 0001→00011 000001

UTF-8

Kódtartomány (darab)	bináris alak	UTF-8
000000-00007F (128)	0zzzzzzz	0zzzzzzz
000080-0007FF (1920)	00000yyy yyzzzzzz	110yyyyy 10zzzzzz
000800-00FFFF (63488)	xxxxyyyy yyzzzzzz	1110xxxx 10yyyyyy 10zzzzzz
010000-10FFFF (1048576)	000wwwxx xxxxyyyy yyzzzzzz	11110www 10xxxxxx 10yyyyyy 10zzz

Á 00C1→1100 0001→00011 000001→11000011 10000001

UTF-8

Kódtartomány (darab)	bináris alak	UTF-8
000000-00007F (128)	0zzzzzzz	0zzzzzzz
000080-0007FF (1920)	00000yyy yyzzzzzz	110yyyyy 10zzzzzz
000800-00FFFF (63488)	xxxxyyyy yyzzzzzz	1110xxxx 10yyyyyy 10zzzzzz
010000-10FFFF (1048576)	000wwwxx xxxxyyyy yyzzzzzz	11110www 10xxxxxx 10yyyyyy 10zzz

Á 00C1→1100 0001→00011 000001→11000011 10000001→C3 81

UTF-8

Kódtartomány (darab)	bináris alak	UTF-8
000000-00007F (128)	0zzzzzzz	0zzzzzzz
000080-0007FF (1920)	00000yyy yyzzzzzz	110yyyyy 10zzzzzz
000800-00FFFF (63488)	xxxxyyyy yyzzzzzz	1110xxxx 10yyyyyy 10zzzzzz
010000-10FFFF (1048576)	000wwwxx xxxxyyyy yyzzzzzz	11110www 10xxxxxx 10yyyyyy 10zzz

Á 00C1→1100 0001→00011 000001→11000011 10000001→C3 81

Õ 00D5→1101 0101→00011 010101→11000011 10010101→C3 95

UTF-8

Kódtartomány (darab)	bináris alak	UTF-8
000000-00007F (128)	0zzzzzzz	0zzzzzzz
000080-0007FF (1920)	00000yyy yyzzzzzz	110yyyyy 10zzzzzz
000800-00FFFF (63488)	xxxxyyyy yyzzzzzz	1110xxxx 10yyyyyy 10zzzzzz
010000-10FFFF (1048576)	000wwwxx xxxxyyyy yyzzzzzz	11110www 10xxxxxx 10yyyyyy 10zzz

Á 00C1→1100 0001→00011 000001→11000011 10000001→C3 81

Ő 00D5→1101 0101→00011 010101→11000011 10010101→C3 95

Ő 0150→0001 0101 0000→00101 010000→11000101

10010000→C5 90

UTF-8

Kódtartomány (darab)	bináris alak	UTF-8
000000-00007F (128)	0zzzzzzz	0zzzzzzz
000080-0007FF (1920)	00000yyy yyzzzzzz	110yyyyy 10zzzzzz
000800-00FFFF (63488)	xxxxyyyy yyzzzzzz	1110xxxx 10yyyyyy 10zzzzzz
010000-10FFFF (1048576)	000wwwxx xxxxyyyy yyzzzzzz	11110www 10xxxxxx 10yyyyyy 10zzz

Á 00C1→1100 0001→00011 000001→11000011 10000001→C3 81

Ő 00D5→1101 0101→00011 010101→11000011 10010101→C3 95

Ő 0150→0001 0101 0000→00101 010000→11000101

10010000→C5 90

Byte Order Mark FEFF

UTF-8

Kódtartomány (darab)	bináris alak	UTF-8
000000-00007F (128)	0zzzzzzz	0zzzzzzz
000080-0007FF (1920)	00000yyy yyzzzzzz	110yyyyy 10zzzzzz
000800-00FFFF (63488)	xxxxyyyy yyzzzzzz	1110xxxx 10yyyyyy 10zzzzzz
010000-10FFFF (1048576)	000wwwxx xxxxyyyy yyzzzzzz	11110www 10xxxxxx 10yyyyyy 10zzz

Á 00C1→1100 0001→00011 000001→11000011 10000001→C3 81

Ő 00D5→1101 0101→00011 010101→11000011 10010101→C3 95

Ő 0150→0001 0101 0000→00101 010000→11000101

10010000→C5 90

Byte Order Mark FEFF→11111110 11111111→

11101111 10111011 10111111

UTF-8

Kódtartomány (darab)	bináris alak	UTF-8
000000-00007F (128)	0zzzzzzz	0zzzzzzz
000080-0007FF (1920)	00000yyy yyzzzzzz	110yyyyy 10zzzzzz
000800-00FFFF (63488)	xxxxyyyy yyzzzzzz	1110xxxx 10yyyyyy 10zzzzzz
010000-10FFFF (1048576)	000wwwxx xxxxyyyy yyzzzzzz	11110www 10xxxxxx 10yyyyyy 10zzz

Á 00C1→1100 0001→00011 000001→11000011 10000001→C3 81

Ő 00D5→1101 0101→00011 010101→11000011 10010101→C3 95

Ő 0150→0001 0101 0000→00101 010000→11000101
10010000→C5 90

Byte Order Mark FEFF→11111110 11111111→
11101111 10111011 10111111→EF BB BF (ï»¿ Windows fájlok elején az UTF-8 formátum jelzése – ha latin-1-ben látjuk)

UTF-8

Kódtartomány (darab)	bináris alak	UTF-8
000000-00007F (128)	0zzzzzzz	0zzzzzzz
000080-0007FF (1920)	00000yyy yyzzzzzz	110yyyyy 10zzzzzz
000800-00FFFF (63488)	xxxxyyyy yyzzzzzz	1110xxxx 10yyyyyy 10zzzzzz
010000-10FFFF (1048576)	000wwwxx xxxxyyyy yyzzzzzz	11110www 10xxxxxx 10yyyyyy 10zzz

Á 00C1→1100 0001→00011 000001→11000011 10000001→C3 81
Ő 00D5→1101 0101→00011 010101→11000011 10010101→C3 95
Ő 0150→0001 0101 0000→00101 010000→11000101
10010000→C5 90
Byte Order Mark FEFF→11111110 11111111→
11101111 10111011 10111111→EF BB BF (ï»¿ Windows fájlok elején az UTF-8 formátum jelzése – ha latin-1-ben látjuk)

Egy `file` nevű fájl tartalmának hexadecimális módon való megtekintése parancssorból (hexdump): `hd file`

RAM-gép (random access machine)

- A RAM-gép egy természetes számokkal indexelt p programtárból és egy természetes számokkal indexelt r adattárból áll, mely induláskor csak 0-kat tartalmaz.

RAM-gép (random access machine)

- A RAM-gép egy természetes számokkal indexelt p programtárból és egy természetes számokkal indexelt r adattárból áll, mely induláskor csak 0-kat tartalmaz.
- A program végrehajtása a p_0 cellájába írt utasítással indul és egy üres utasítással zárul.

RAM-gép (random access machine)

- A RAM-gép egy természetes számokkal indexelt p programtárból és egy természetes számokkal indexelt r adattárból áll, mely induláskor csak 0-kat tartalmaz.
- A program végrehajtása a p_0 cellájába írt utasítással indul és egy üres utasítással zárul.
- Az adattár i-edik cellájának ($i \in \mathbb{N}_0$) tartalmát $r[i]$ vagy r_i jelöli, ami csak egész szám lehet.

RAM-gép (random access machine)

- A RAM-gép egy természetes számokkal indexelt p programtárból és egy természetes számokkal indexelt r adattárból áll, mely induláskor csak 0-kat tartalmaz.
- A program végrehajtása a p_0 cellájába írt utasítással indul és egy üres utasítással zárul.
- Az adattár i-edik cellájának ($i \in \mathbb{N}_0$) tartalmát $r[i]$ vagy r_i jelöli, ami csak egész szám lehet.
- Megengedett utasítások, ahol $z \in \mathbb{Z}$, $i, n \in \mathbb{N}_0$:

RAM-gép (random access machine)

- A RAM-gép egy természetes számokkal indexelt p programtárból és egy természetes számokkal indexelt r adattárból áll, mely induláskor csak 0-kat tartalmaz.
- A program végrehajtása a p_0 cellájába írt utasítással indul és egy üres utasítással zárul.
- Az adattár i-edik cellájának ($i \in \mathbb{N}_0$) tartalmát $r[i]$ vagy r_i jelöli, ami csak egész szám lehet.
- Megengedett utasítások, ahol $z \in \mathbb{Z}$, $i, n \in \mathbb{N}_0$:

  $r_i \leftarrow z$

RAM-gép (random access machine)

- A RAM-gép egy természetes számokkal indexelt p programtárból és egy természetes számokkal indexelt r adattárból áll, mely induláskor csak 0-kat tartalmaz.
- A program végrehajtása a p_0 cellájába írt utasítással indul és egy üres utasítással zárul.
- Az adattár i-edik cellájának ($i \in \mathbb{N}_0$) tartalmát $r[i]$ vagy r_i jelöli, ami csak egész szám lehet.
- Megengedett utasítások, ahol $z \in \mathbb{Z}$, $i, n \in \mathbb{N}_0$:

  $r_i \leftarrow z$

  $r_i \leftarrow r_n$, $r_i \leftarrow r_{r_n}$ (azaz $r_i \leftarrow r[r[n]]$),

RAM-gép (random access machine)

- A RAM-gép egy természetes számokkal indexelt p programtárból és egy természetes számokkal indexelt r adattárból áll, mely induláskor csak 0-kat tartalmaz.
- A program végrehajtása a p_0 cellájába írt utasítással indul és egy üres utasítással zárul.
- Az adattár i-edik cellájának ($i \in \mathbb{N}_0$) tartalmát $r[i]$ vagy r_i jelöli, ami csak egész szám lehet.
- Megengedett utasítások, ahol $z \in \mathbb{Z}$, $i, n \in \mathbb{N}_0$:

  $r_i \leftarrow z$

  $r_i \leftarrow r_n$, $r_i \leftarrow r_{r_n}$ (azaz $r_i \leftarrow r[r[n]]$),

  $r_i \leftarrow r_i \pm r_n$, ($r_i \leftarrow r_i * r_n$, $r_i \leftarrow r_i / r_n$),

RAM-gép (random access machine)

- A RAM-gép egy természetes számokkal indexelt p programtárból és egy természetes számokkal indexelt r adattárból áll, mely induláskor csak 0-kat tartalmaz.
- A program végrehajtása a p_0 cellájába írt utasítással indul és egy üres utasítással zárul.
- Az adattár i-edik cellájának ($i \in \mathbb{N}_0$) tartalmát $r[i]$ vagy r_i jelöli, ami csak egész szám lehet.
- Megengedett utasítások, ahol $z \in \mathbb{Z}$, $i, n \in \mathbb{N}_0$:

  $r_i \leftarrow z$

  $r_i \leftarrow r_n$, $r_i \leftarrow r_{r_n}$ (azaz $r_i \leftarrow r[r[n]]$),

  $r_i \leftarrow r_i \pm r_n$, ($r_i \leftarrow r_i * r_n$, $r_i \leftarrow r_i / r_n$),

  p_n: ugrás az n-edik programsorra,

RAM-gép (random access machine)

- A RAM-gép egy természetes számokkal indexelt p programtárból és egy természetes számokkal indexelt r adattárból áll, mely induláskor csak 0-kat tartalmaz.
- A program végrehajtása a p_0 cellájába írt utasítással indul és egy üres utasítással zárul.
- Az adattár i-edik cellájának ($i \in \mathbb{N}_0$) tartalmát $r[i]$ vagy r_i jelöli, ami csak egész szám lehet.
- Megengedett utasítások, ahol $z \in \mathbb{Z}$, $i, n \in \mathbb{N}_0$:

  $r_i \leftarrow z$

  $r_i \leftarrow r_n$, $r_i \leftarrow r_{r_n}$ (azaz $r_i \leftarrow r[r[n]]$),

  $r_i \leftarrow r_i \pm r_n$, ($r_i \leftarrow r_i * r_n$, $r_i \leftarrow r_i / r_n$),

  p_n: ugrás az n-edik programsorra,

  if $r_i = 0$ p_n: ugrás az n-edik programsorra, ha $r_i = 0$,

RAM-gép (random access machine)

- A RAM-gép egy természetes számokkal indexelt p programtárból és egy természetes számokkal indexelt r adattárból áll, mely induláskor csak 0-kat tartalmaz.
- A program végrehajtása a p_0 cellájába írt utasítással indul és egy üres utasítással zárul.
- Az adattár i-edik cellájának ($i \in \mathbb{N}_0$) tartalmát $r[i]$ vagy r_i jelöli, ami csak egész szám lehet.
- Megengedett utasítások, ahol $z \in \mathbb{Z}$, $i, n \in \mathbb{N}_0$:

  $r_i \leftarrow z$

  $r_i \leftarrow r_n$, $r_i \leftarrow r_{r_n}$ (azaz $r_i \leftarrow r[r[n]]$),

  $r_i \leftarrow r_i \pm r_n$, ($r_i \leftarrow r_i * r_n$, $r_i \leftarrow r_i / r_n$),

  p_n: ugrás az n-edik programsorra,

  if $r_i = 0$ p_n: ugrás az n-edik programsorra, ha $r_i = 0$,

  if $r_i > 0$ p_n: ugrás az n-edik programsorra, ha $r_i > 0$,

RAM-gép (random access machine)

A RAM-gép előadáson bemutatott „számítógépszerű” változata:

- A programtár és a memória véges,

RAM-gép (random access machine)

A RAM-gép előadáson bemutatott „számítógépszerű” változata:

- A programtár és a memória véges,
- minden memóriacella 1-bytos, minden utasítás 2-bytos, az első byte az utasítást, a második az operandust tartalmazza, pl.

RAM-gép (random access machine)

A RAM-gép előadáson bemutatott „számítógépszerű” változata:

- A programtár és a memória véges,
- minden memóriacella 1-bytos, minden utasítás 2-bytos, az első byte az utasítást, a második az operandust tartalmazza, pl.

  `ADD 12` jelentése: $r_0 \leftarrow r_0 + r_{12}$

RAM-gép (random access machine)

A RAM-gép előadáson bemutatott „számítógépszerű” változata:

- A programtár és a memória véges,
- minden memóriacella 1-bytos, minden utasítás 2-bytos, az első byte az utasítást, a második az operandust tartalmazza, pl.

  `ADD 12` jelentése: $r_0 \leftarrow r_0 + r_{12}$
- számítás és összehasonlítás csak a 0-dik memóriacella (és esetleg egy másik) tartalmával végezhető,

RAM-gép (random access machine)

A RAM-gép előadáson bemutatott „számítógépszerű” változata:

- A programtár és a memória véges,
- minden memóriacella 1-bytos, minden utasítás 2-bytos, az első byte az utasítást, a második az operandust tartalmazza, pl.

  `ADD 12` jelentése: $r_0 \leftarrow r_0 + r_{12}$
- számítás és összehasonlítás csak a 0-dik memóriacella (és esetleg egy másik) tartalmával végezhető,
- az utasításokat mnemonikokkal jelöljük, ezek három változata:

RAM-gép (random access machine)

A RAM-gép előadáson bemutatott „számítógépszerű” változata:

- A programtár és a memória véges,
- minden memóriacella 1-bytos, minden utasítás 2-bytos, az első byte az utasítást, a második az operandust tartalmazza, pl.

  `ADD 12` jelentése: $r_0 \leftarrow r_0 + r_{12}$
- számítás és összehasonlítás csak a 0-dik memóriacella (és esetleg egy másik) tartalmával végezhető,
- az utasításokat mnemonikokkal jelöljük, ezek három változata:
  - közvetlen: az n operandus egy szám, a műveletet azzal végezzük (jelölése az utasítás végére tett $=$)

RAM-gép (random access machine)

A RAM-gép előadáson bemutatott „számítógépszerű” változata:

- A programtár és a memória véges,
- minden memóriacella 1-bytos, minden utasítás 2-bytos, az első byte az utasítást, a második az operandust tartalmazza, pl.

  `ADD 12` jelentése: $r_0 \leftarrow r_0 + r_{12}$
- számítás és összehasonlítás csak a 0-dik memóriacella (és esetleg egy másik) tartalmával végezhető,
- az utasításokat mnemonikokkal jelöljük, ezek három változata:
  - közvetlen: az n operandus egy szám, a műveletet azzal végezzük (jelölése az utasítás végére tett $=$)
  - direkt: az n operandust memóriacímnek tekintjük és a műveletet az $r[n]$ (azaz az r_n) tartalmával végezzük,

RAM-gép (random access machine)

A RAM-gép előadáson bemutatott „számítógépszerű” változata:

- A programtár és a memória véges,
- minden memóriacella 1-bytos, minden utasítás 2-bytos, az első byte az utasítást, a második az operandust tartalmazza, pl.

  `ADD 12` jelentése: $r_0 \leftarrow r_0 + r_{12}$
- számítás és összehasonlítás csak a 0-dik memóriacella (és esetleg egy másik) tartalmával végezhető,
- az utasításokat mnemonikokkal jelöljük, ezek három változata:
  - közvetlen: az n operandus egy szám, a műveletet azzal végezzük (jelölése az utasítás végére tett =)
  - direkt: az n operandust memóriacímnek tekintjük és a műveletet az $r[n]$ (azaz az r_n) tartalmával végezzük,
  - indirekt: az n operandust egy memóriacím címének tekintjük, a műveletet az $r[r[n]]$ (azaz az r_{r_n}) számmal végezzük (jelölése az utasítás végére tett *)

RAM-gép (random access machine)

Vezérlő utasítások

`JUMP`	n	ugrás az n-edik utasításra
`JZERO`	n	ugrás az n-edik utasításra, ha $r_0 = 0$
`JGTZ`	n	ugrás az n-edik utasításra, ha $r_0 > 0$
`HALT`		leállás

Aritmetikai utasítások

	direkt		*indirekt*		*közvetlen op*	
`ADD` n	$r_0 \leftarrow r_0 + r_n$	`ADD*` n	$r_0 \leftarrow r_0 + r_{r_n}$	`ADD=` n	$r_0 \leftarrow r_0 + n$
`SUB` n	$r_0 \leftarrow r_0 - r_n$	`SUB*` n	$r_0 \leftarrow r_0 - r_{r_n}$	`SUB=` n	$r_0 \leftarrow r_0 - n$
`MULT` n	$r_0 \leftarrow r_0 * r_n$	`MULT*` n	$r_0 \leftarrow r_0 * r_{r_n}$	`MULT=` n	$r_0 \leftarrow r_0 * n$
`DIV` n	$r_0 \leftarrow r_0 / r_n$	`DIV*` n	$r_0 \leftarrow r_0 / r_{r_n}$	`DIV=` n	$r_0 \leftarrow r_0 / n$

Adatmozgatás, IO

	direkt		*indirekt*		*közvetlen op*	
`LOAD` n	$r_0 \leftarrow r_n$	`LOAD*` n	$r_0 \leftarrow r_{r_n}$	`LOAD=` n	$r_0 \leftarrow n$
`STORE` n	$r_n \leftarrow r_0$	`STORE*` n	$r_{r_n} \leftarrow r_0$		
`READ` n	az input eszközről $r_1, r_2, \ldots, r_n$-be olvas n számot				
`WRITE` n	az output eszközre írja $r_1, r_2, \ldots, r_n$ tartalmát				

RAM-gép (random access machine)

Írjunk programot (a, b) kiszámítására, ahol $a, b \in \mathbb{N}_0$!

```
 p   utasítás operandus  megjegyzés
 0   LOAD  =  12
 1   STORE     1         r[1] <- a
 2   LOAD  =  16
 3   STORE     2         r[2] <- b
 4   JZERO    17
 5   LOAD      1         r[0] <- r[1]
 6   DIV       2         r[0] <- ⌊a/b⌋
 7   STORE     3         r[3] <- ⌊a/b⌋
 8   MULT      2
 9   STORE     4         r[4] <- b*⌊a/b⌋
10   LOAD      1
11   SUB       4         r[0] <- a - b*⌊a/b⌋ = a mod b
12   STORE     5
13   LOAD      2
14   STORE     1         r[1] <- b
15   LOAD      5         b <- a mod b
16   JUMP      3
17   LOAD      1
18   STORE     6         itt az (a,b)
19   HALT      0
```

RAM-gép (random access machine)

Írjunk programot a Collatz-problémára: legyen $x \in \mathbb{N}^+$, ha x páros legyen $x \leftarrow x/2$, ha x páratlan, legyen $x \leftarrow 3x + 1$. Ellenőrizzük, hogy valamely x-ből indulva eljutunk-e az 1-esig.

p	Assembly	op.	Gépi kód		3x + 1 (COLLATZ PROBLÉMA)
0	LOAD =	33	10000011	00100001	input érték megadása
1	STORE	2	10010000	00000010	a 2-ben tároljuk
2	DIV =	2	01110011	00000010	osztjuk 2-vel
3	STORE	1	10010000	00000001	1 rekeszbe
4	MULT =	2	01100011	00000010	szorozva 2-vel
5	SUB	2	01010000	00000010	
6	JZERO	11	11100000	00001100	ha páros volt, elugrunk
7	LOAD	2	10000000	00000010	
8	MULT =	3	01100011	00000011	szorzás 3-mal
9	ADD =	1	01000011	00000001	plusz 1
10	JUMP	1	11010000	00000010	ugrás 1-re
11	LOAD	1	10000000	00000001	ha páros volt
12	STORE	2	10010000	00000010	
13	SUB =	1	01010011	00000001	egyenlő 1?
14	JZERO	17	11100000	00010010	ha igen, HALT
15	LOAD	1	10000000	00000001	ha nem, tovább
16	JUMP	2	11010000	00000010	ugrás 2-re
17	HALT		11000000	00000000	