Výpočetní modely

Výpočet algoritmu

Algoritmy jsou vykonávány stroji — může to být například:

- skutečný počítač — vykonává instrukce strojového kódu
- virtuální stroj — vykonává instrukce bytekódu
- nějaký idealizovaný matematický model počítače
- ...

Stroj může být:

- jednoúčelový — vykonává jen jeden algoritmus
- obecnější — algoritmus dostává ve formě **programu**

Stroj pracuje po **krocích**.

Algoritmus během výpočtu zpracovává konkrétní **vstup** a produkuje příslušný **výstup**.

Výpočetní modely

Výpočetní model — nějaký idealizovaný matematický model počítače

- abstrahujeme od různých nepodstatných implementačních detailů
- chceme analyzovat ty vlastnosti algoritmů, které pokud možno co nejméně závisí na detailech stroje, který bude daný algoritmus vykonávat

Příklady některých výpočetních modelů:

- konečné automaty
- zásobníkové automaty
- Turingovy stroje
- stroje RAM
- ...

Výpočetní modely

Během výpočtu si stroj typicky musí pamatovat:

- která instrukce se právě provádí
- obsah své pracovní paměti

Podle typu stroje je určeno:

- s jakým typem dat stroj pracuje
- jak jsou tato data v paměti organizována
- jaké operace s těmito daty může stroj vykonávat

Podle typu algoritmu a typu analýzy, kterou chceme provádět, se můžeme rozhodnout, zda má smysl mezi obsah paměti zahranout i místa

- odkud se čtou vstupní data
- kam se zapisují výstupní data

Výpočetní modely

Jedním z využití výpočetních modelů je to, že mohou sloužit pro přesné definování pojmů, důležitých pro stanovení **výpočetní složitosti** daného algoritmu:

- **doby výpočtu** daného algoritmu $\mathcal{A}$ pro daný vstup w
  (pozn.: většinou je to počet kroků vykonaných strojem během výpočtu)
- **množství použité paměti** během tohoto výpočtu

Obecně je pro různé výpočetní modely také důležité

- zda je daný typ stroje schopen **simulovat** výpočty nějakého jiného typu stroje
- jak se při této simulaci liší doba výpočtu či množství použité paměti oproti původnímu stroji

Simulace výpočtu

Vysvětlení toho, co to znamená, že stroj $\mathcal{M}$ je **simulován** strojem $\mathcal{M}'$:

- Výpočet stroje $\mathcal{M}$ pro vstup w je (konečná nebo nekonečná) posloupnost konfigurací stroje $\mathcal{M}$

$$\alpha_0 \longrightarrow \alpha_1 \longrightarrow \alpha_2 \longrightarrow \cdots$$

- Tomuto výpočtu odpovídá výpočet stroje $\mathcal{M}'$ tvořený konfiguracemi

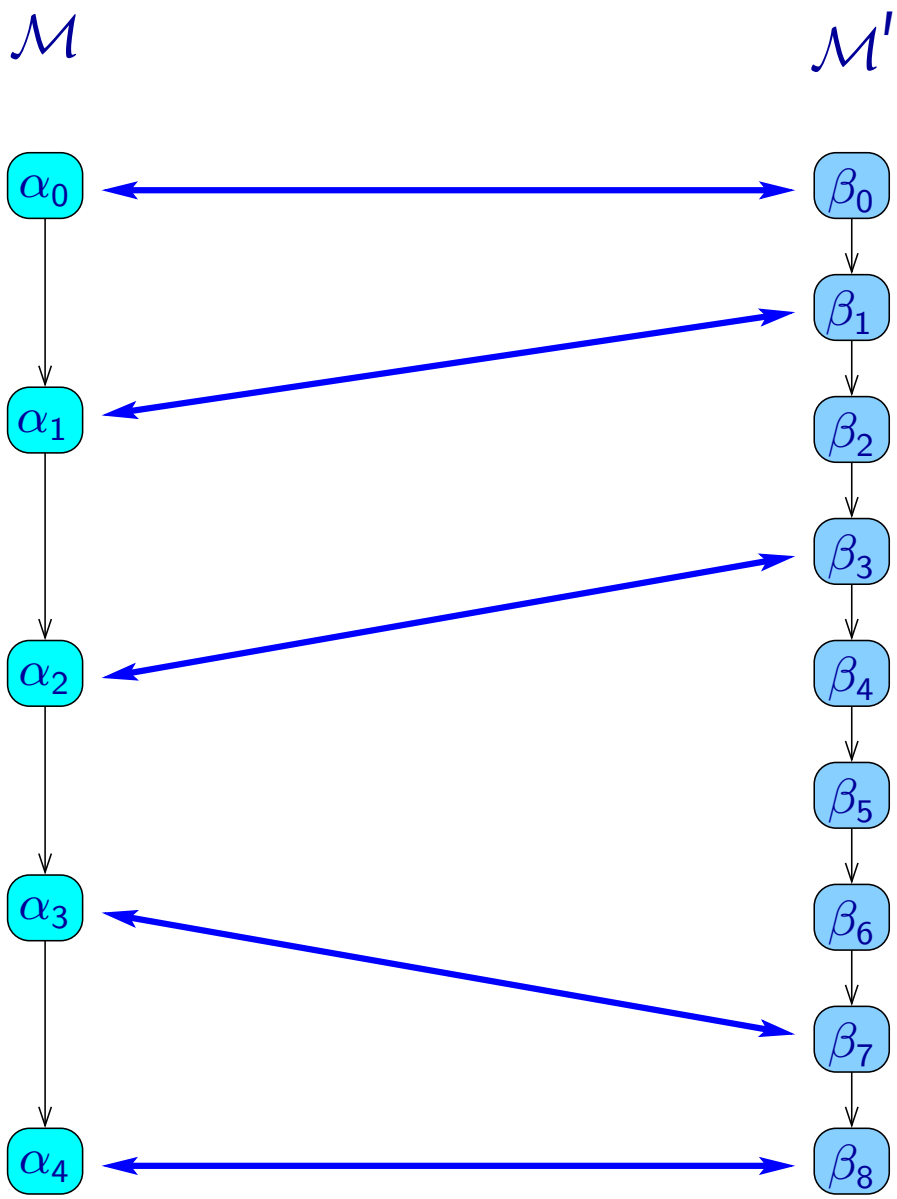
$$\beta_0 \longrightarrow \beta_1 \longrightarrow \beta_2 \longrightarrow \cdots$$

  kde každé konfiguraci α_i odpovídá nějaká konfigurace $\beta_{f(i)}$, kde $f : \mathbb{N} \to \mathbb{N}$ je funkce, pro kterou platí $f(i) \leq f(j)$ pro každé i a j, kde $i < j$.

- Existuje relace mezi vzájemně si odpovídajícími konfiguracemi stroje $\mathcal{M}$ a jim odpovídajícími konfiguracemi stroje $\mathcal{M}'$.

- Existují funkce mapující vstup w na odpovídající počáteční konfigurace α_0 a β_0 a analogicky funkce mapující koncové konfigurace na výsledek výpočtu.

Simulace výpočtu



Výpočetní modely

Některé výpočetní modely jsou slabší (konečné automaty, zásobníkové automaty, . . .) a není pomocí nich možné implementovat libovolný algoritmus.

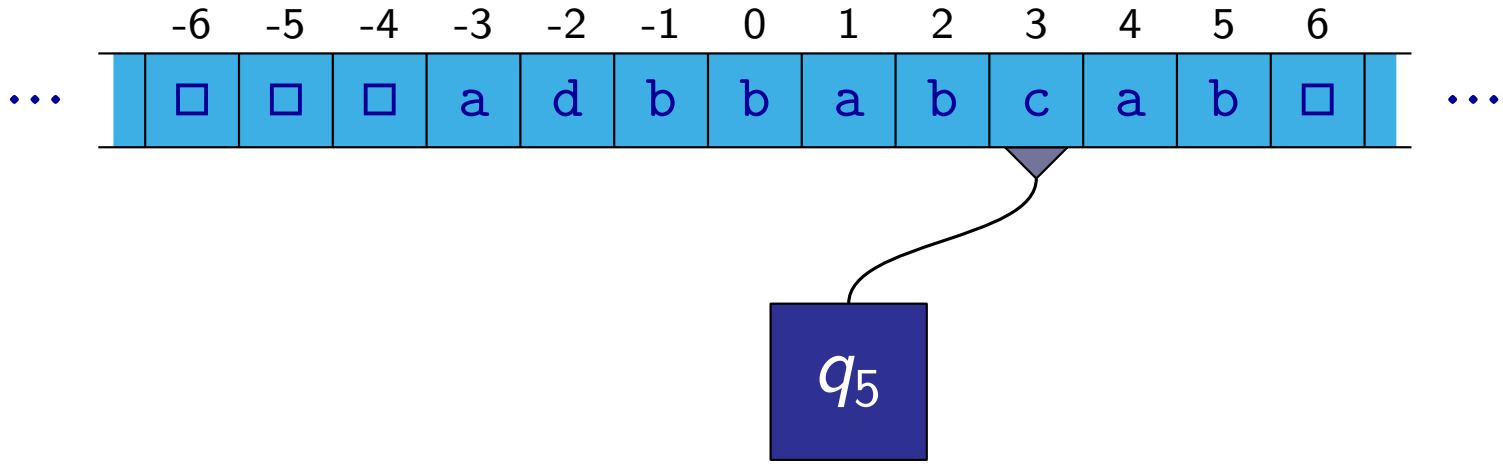
My se teď zaměříme na výpočetní modely, které jsou dostatečně silné na to, aby byly schopny vykonávat libovolný algoritmus (např. takový, jaký je možné zapsat jako program v nějakém programovacím jazyce).

Takovým výpočetním modelům se říká **Turingovsky úplné**:

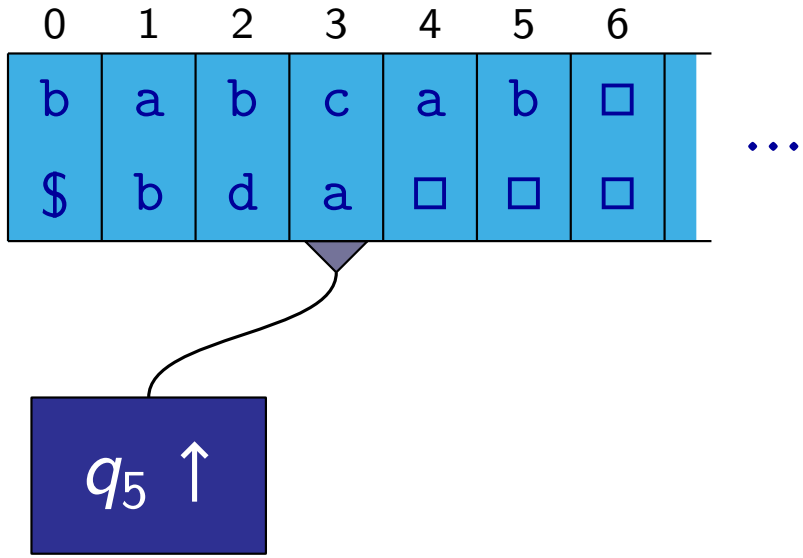
- samy jsou schopny simulovat činnost libovolného Turingova stroje
- jejich činnost může být simulována Turingovým strojem

Oboustranně nekonečná páska pomocí jednostranné

Oboustranně nekonečná páska:

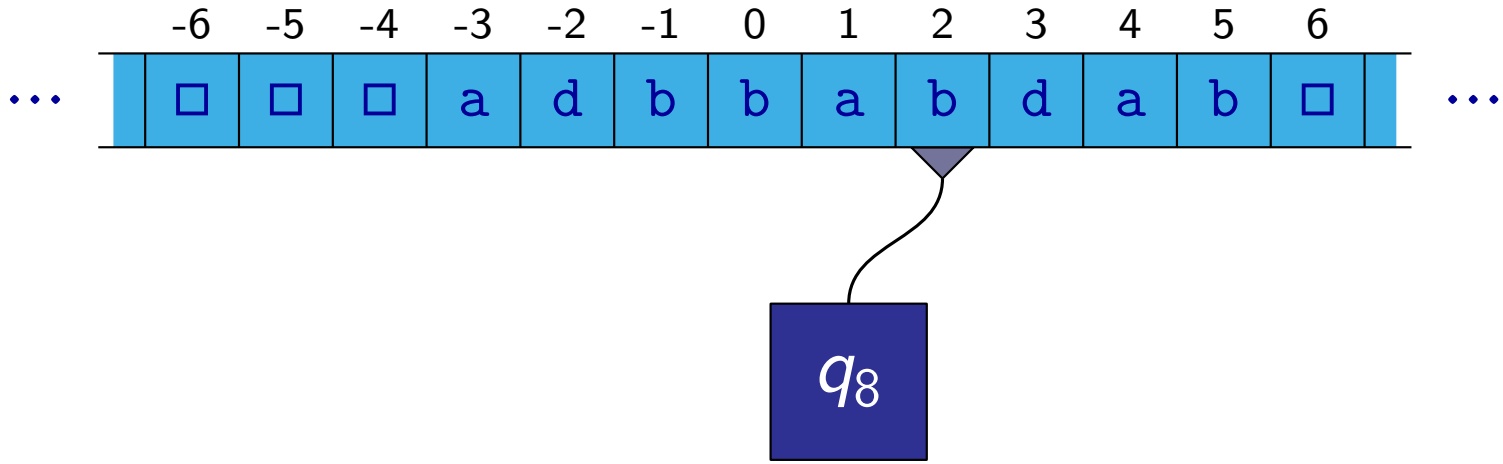


Jednostranně nekonečná páska:

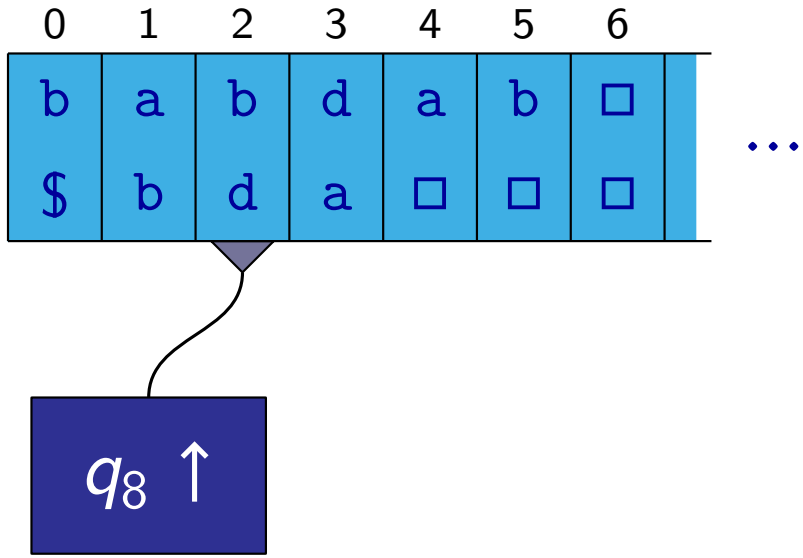


Oboustranně nekonečná páska pomocí jednostranné

Oboustranně nekonečná páska:

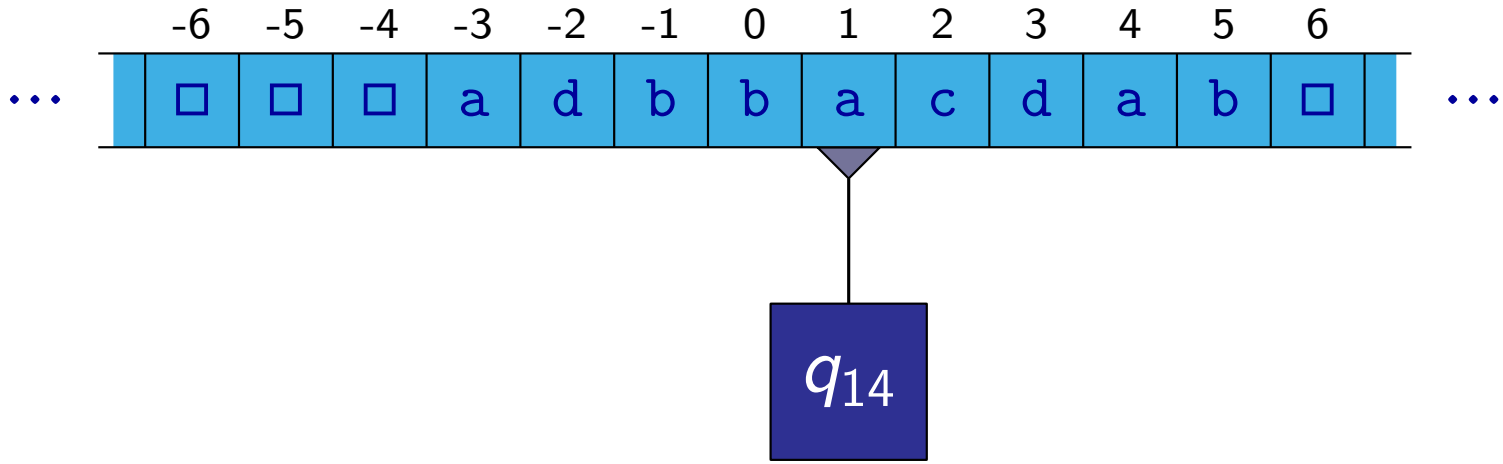


Jednostranně nekonečná páska:

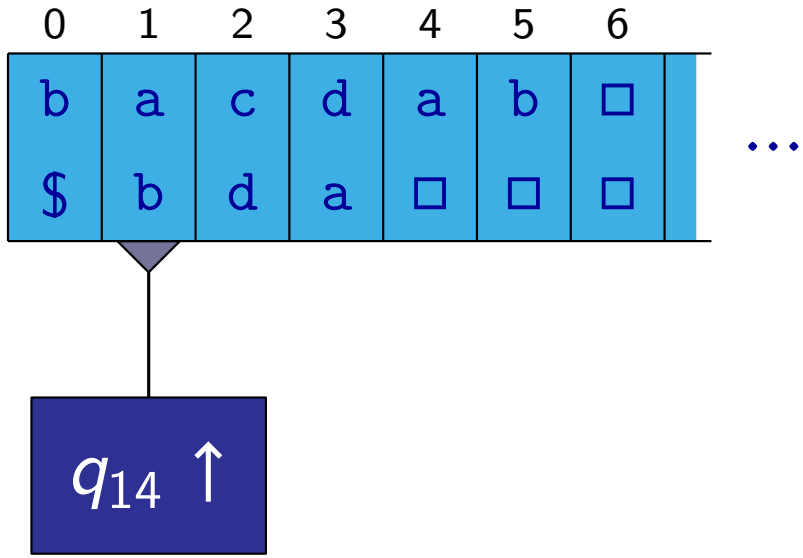


Oboustranně nekonečná páska pomocí jednostranné

Oboustranně nekonečná páska:

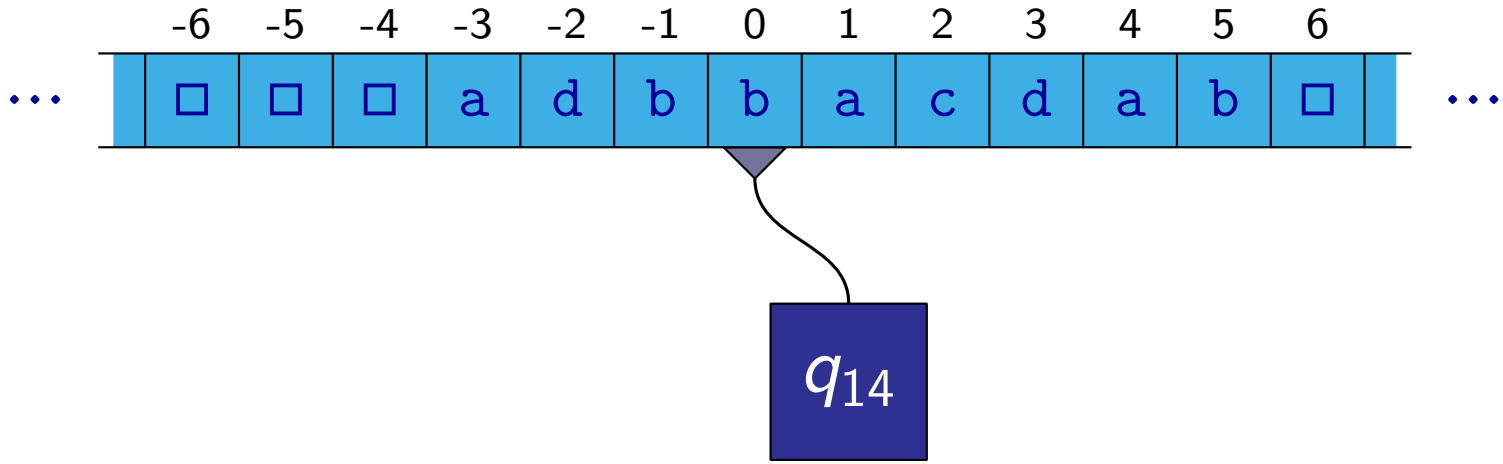


Jednostranně nekonečná páska:

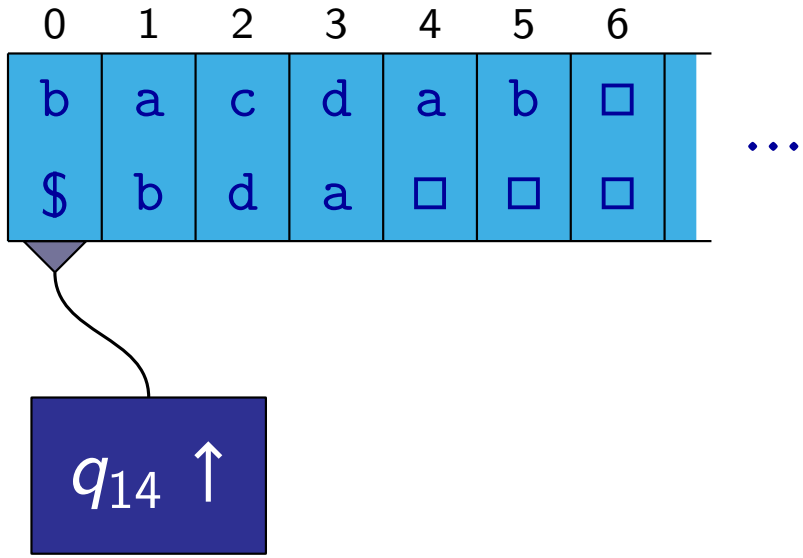


Oboustranně nekonečná páska pomocí jednostranné

Oboustranně nekonečná páska:

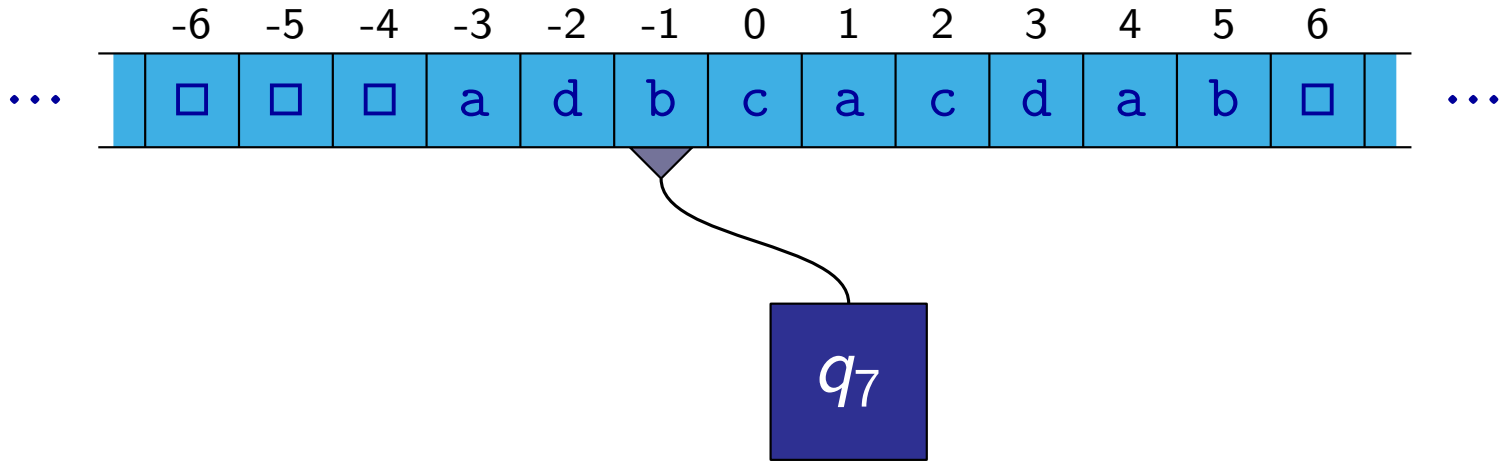


Jednostranně nekonečná páska:

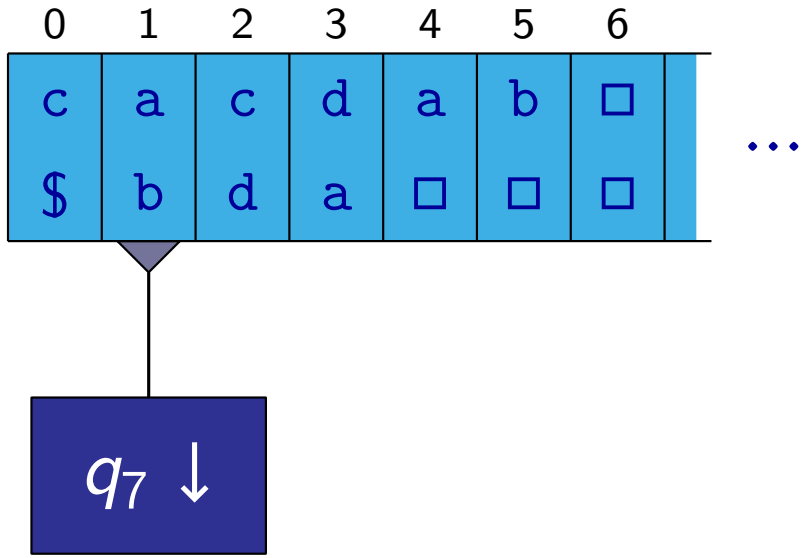


Oboustranně nekonečná páska pomocí jednostranné

Oboustranně nekonečná páska:

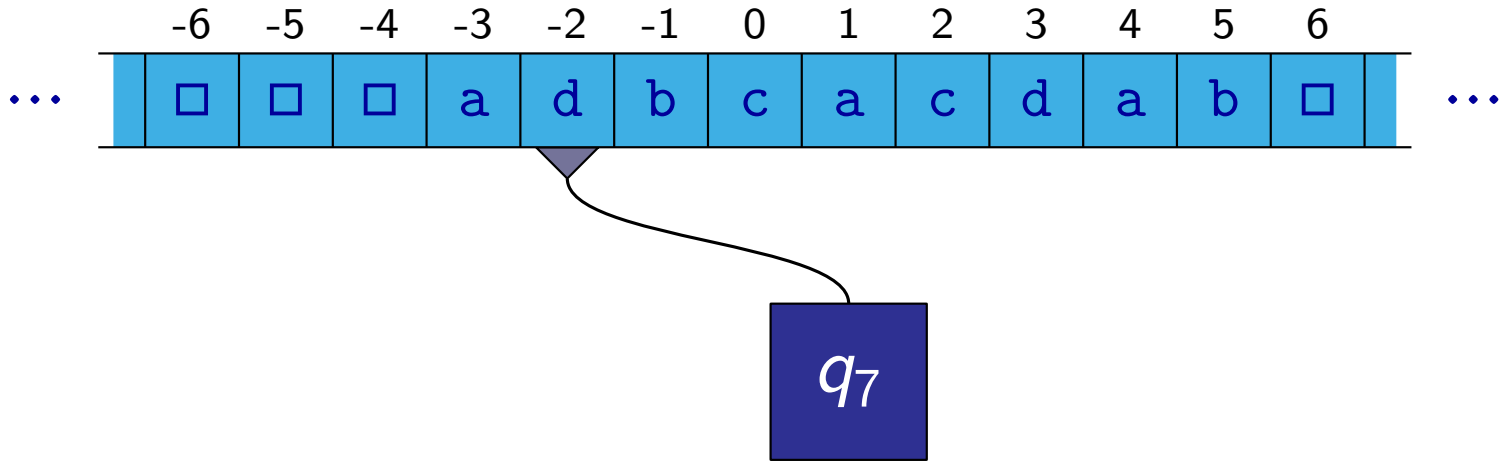


Jednostranně nekonečná páska:

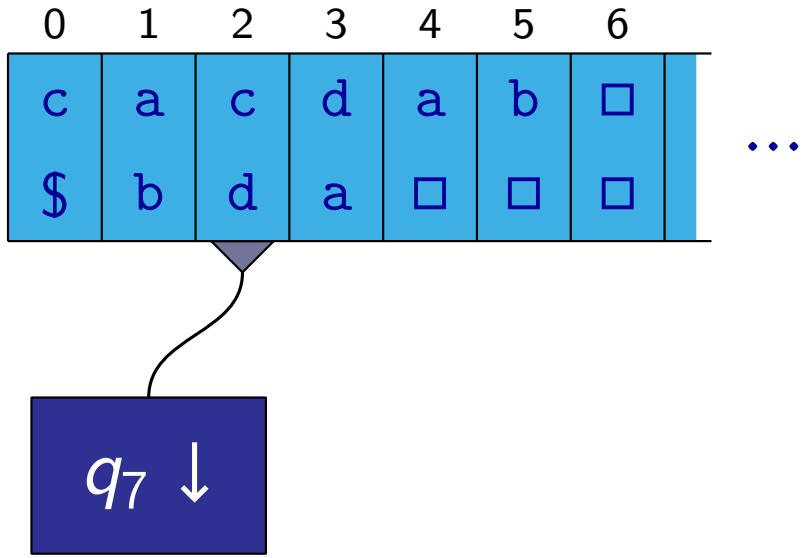


Oboustranně nekonečná páska pomocí jednostranné

Oboustranně nekonečná páska:

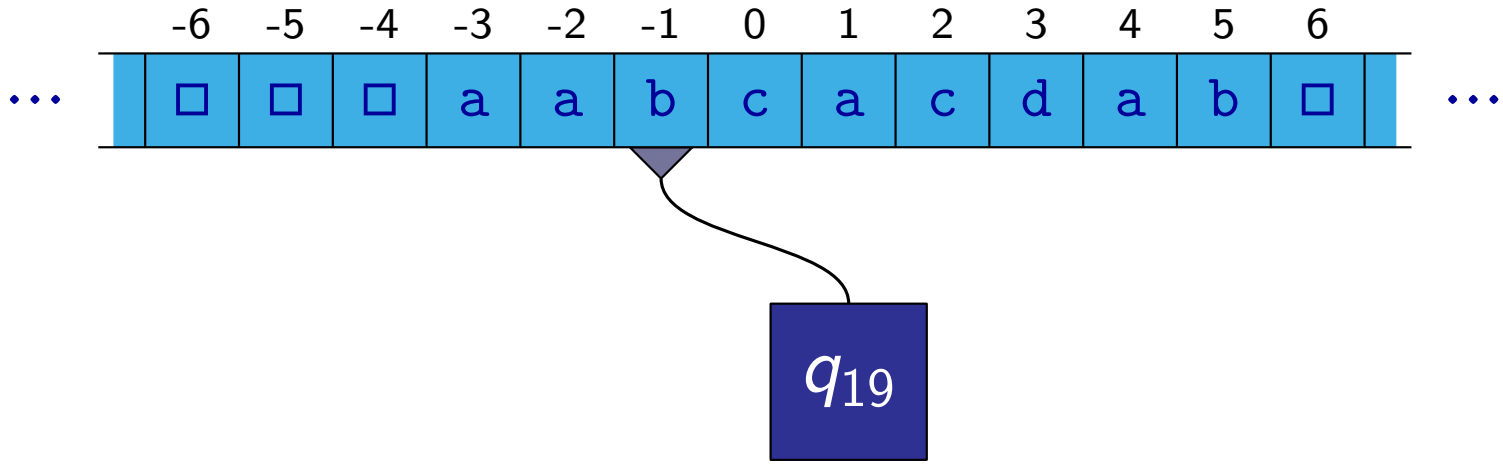


Jednostranně nekonečná páska:

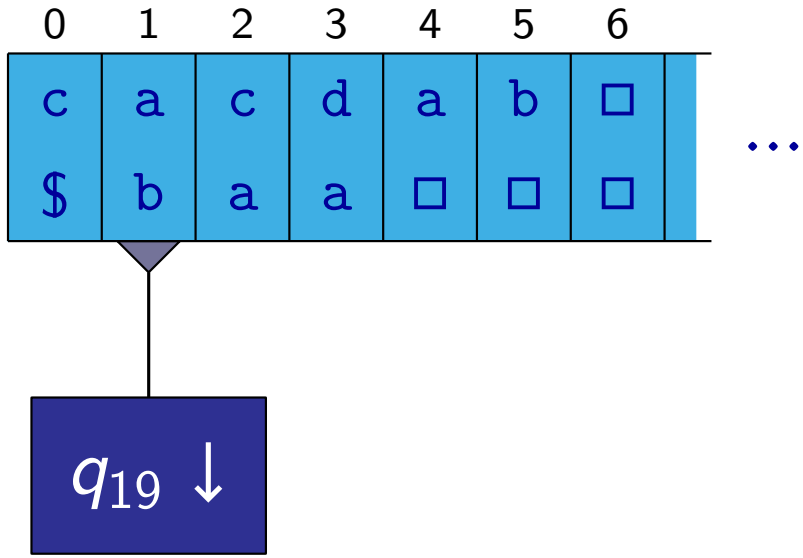


Oboustranně nekonečná páska pomocí jednostranné

Oboustranně nekonečná páska:

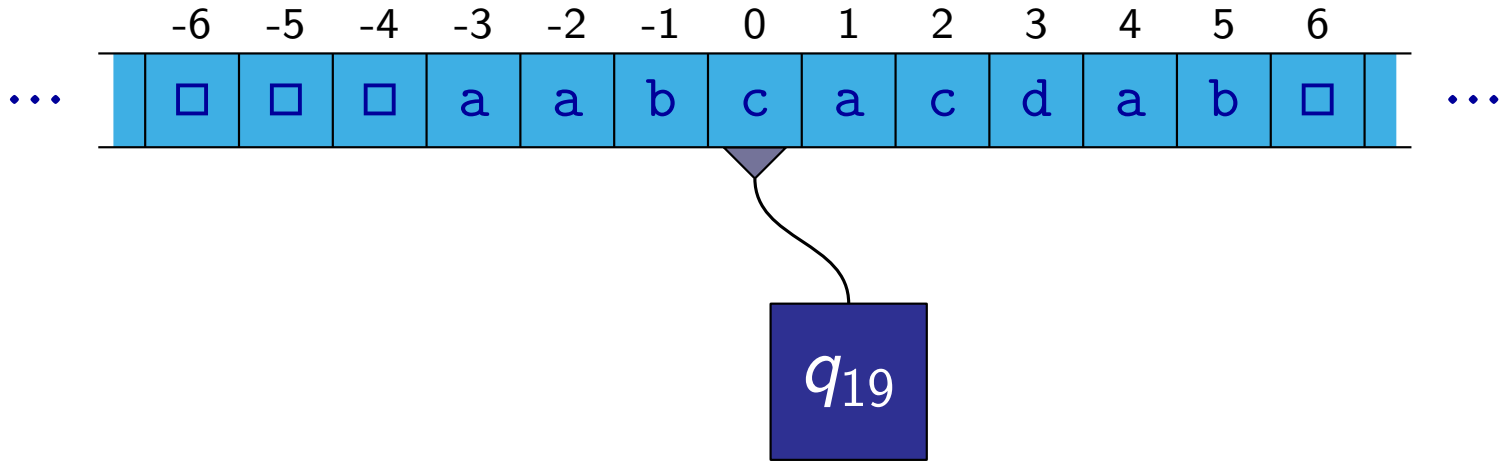


Jednostranně nekonečná páska:

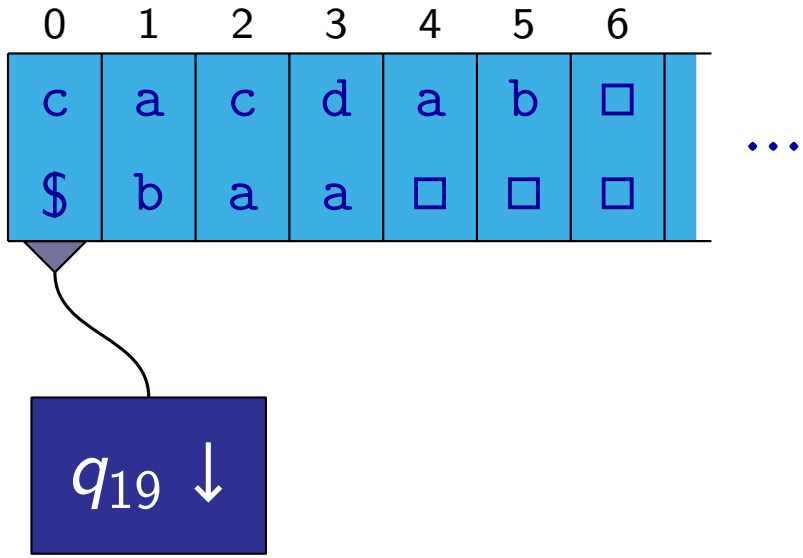


Oboustranně nekonečná páska pomocí jednostranné

Oboustranně nekonečná páska:

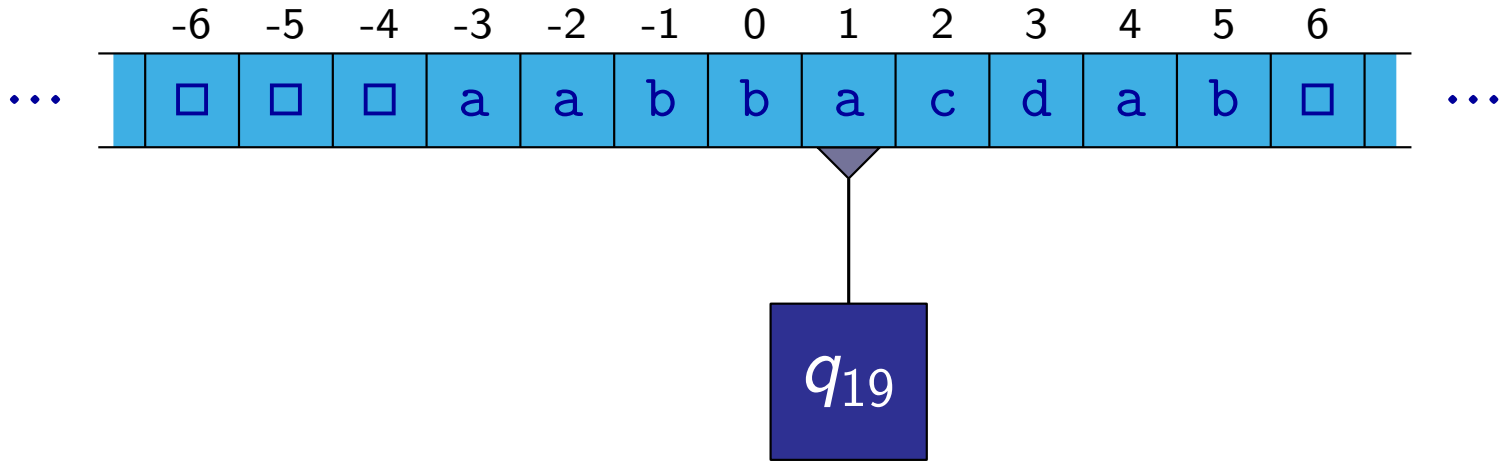


Jednostranně nekonečná páska:

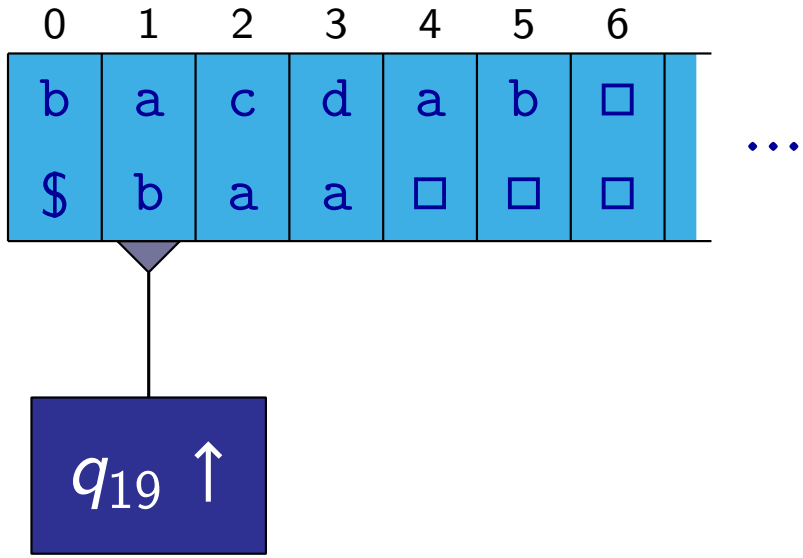


Oboustranně nekonečná páska pomocí jednostranné

Oboustranně nekonečná páska:

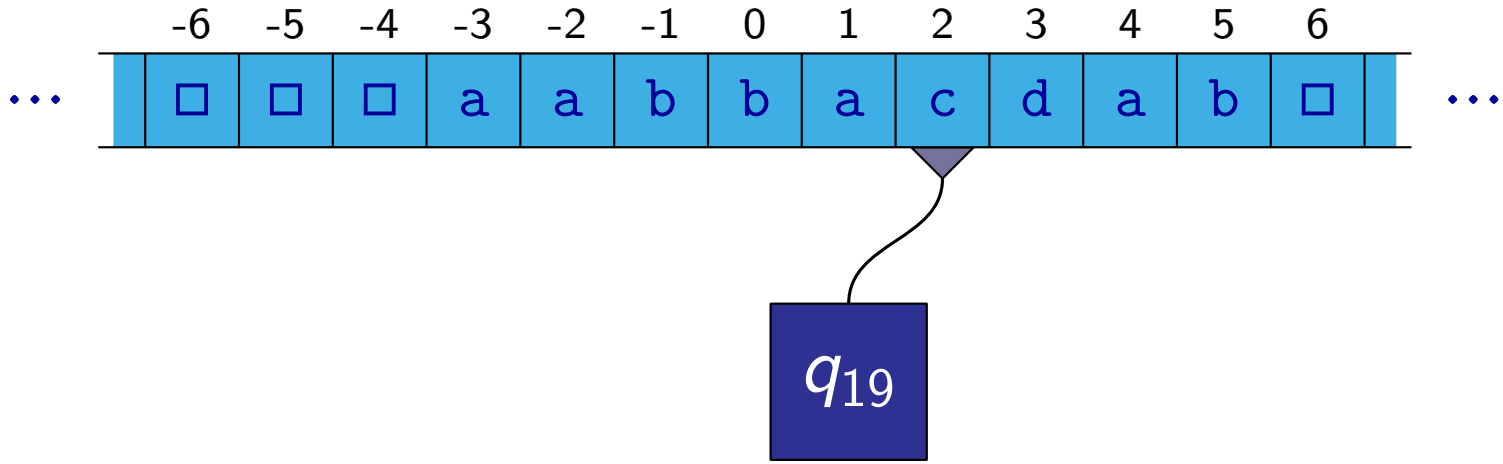


Jednostranně nekonečná páska:

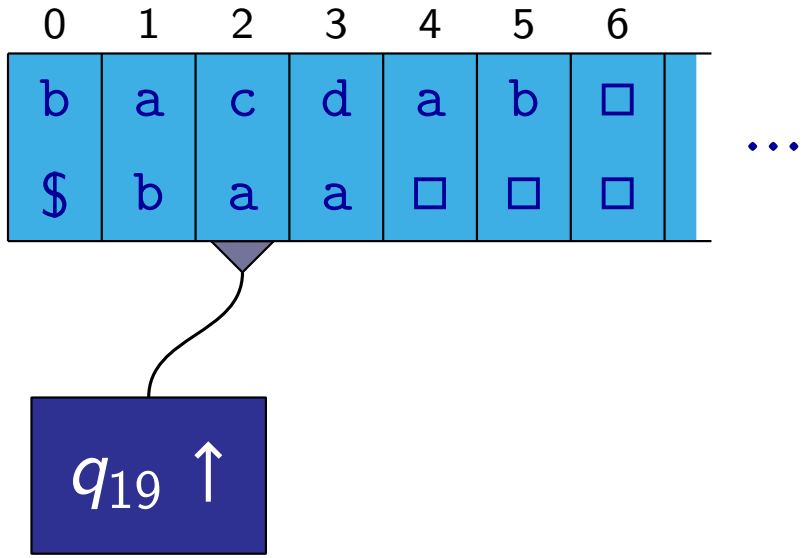


Oboustranně nekonečná páska pomocí jednostranné

Oboustranně nekonečná páska:



Jednostranně nekonečná páska:



Abeceda $\{0, 1\}$

Činnost stroje s libovolnou páskovou abecedou Γ může být simulována strojem s páskovou abecedou $\{0, 1\}$.

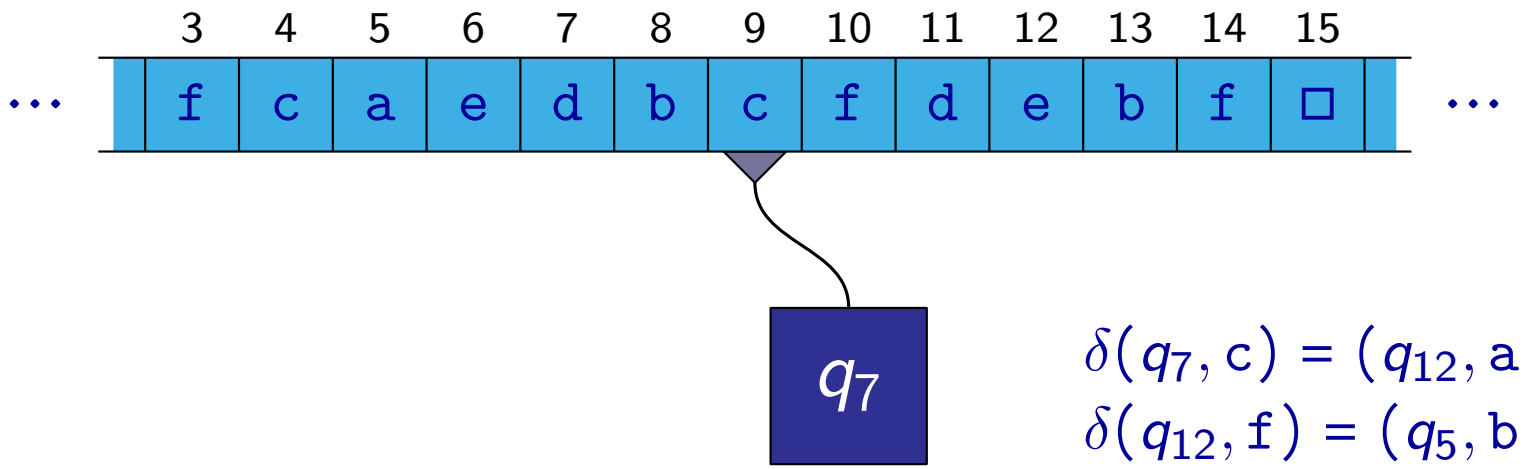
Stačí zvolit nějaké vhodné kódování symbolů abecedy Γ pomocí k-bitových sekvencí.

Příklad: Pásková abeceda $\Gamma = \{\square, \mathtt{a}, \mathtt{b}, \mathtt{c}, \mathtt{d}, \mathtt{e}, \mathtt{f}, \mathtt{g}\}$

$$
\begin{array}{ccc}
\square & \leftrightarrow & 000 \\
\mathtt{a} & \leftrightarrow & 001 \\
\mathtt{b} & \leftrightarrow & 010 \\
\mathtt{c} & \leftrightarrow & 011 \\
\mathtt{d} & \leftrightarrow & 100 \\
\mathtt{e} & \leftrightarrow & 101 \\
\mathtt{f} & \leftrightarrow & 110 \\
\mathtt{g} & \leftrightarrow & 111
\end{array}
$$

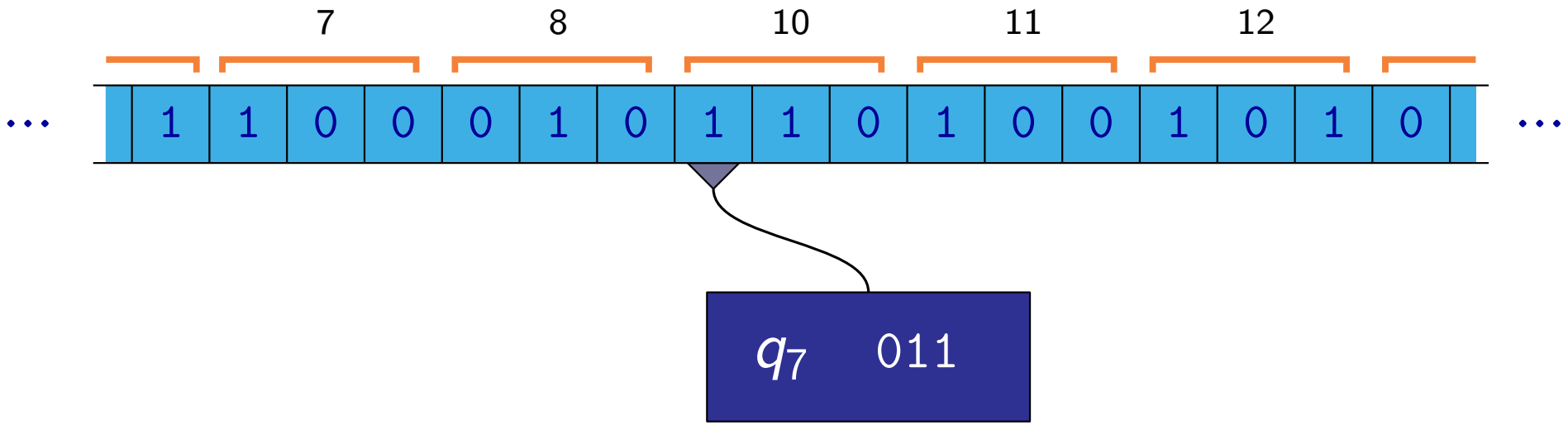
Abeceda $\{0, 1\}$

Stroj s páskovou abecedou Γ:



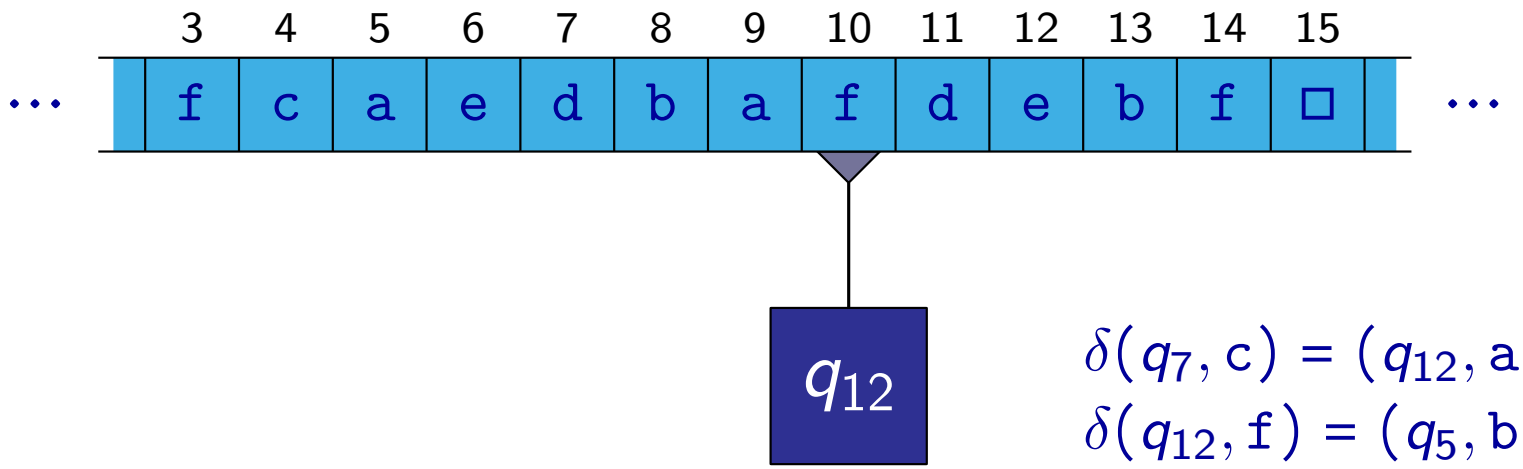
$$\delta(q_7, \mathtt{c}) = (q_{12}, \mathtt{a}, +1)$$
$$\delta(q_{12}, \mathtt{f}) = (q_5, \mathtt{b}, -1)$$

Stroj s abecedou $\{0, 1\}$:



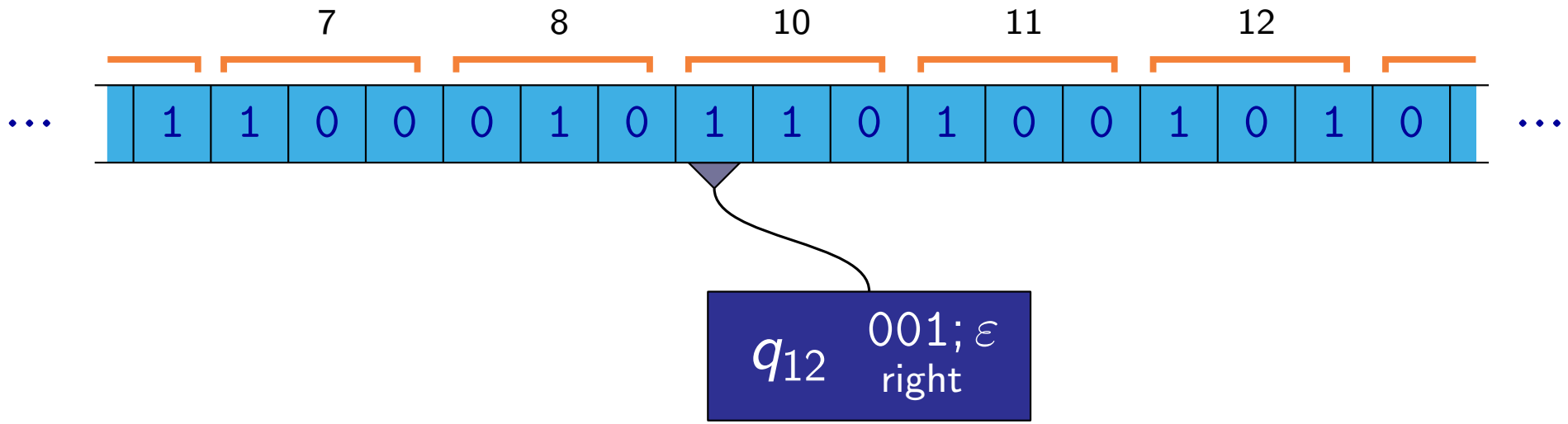
Abeceda $\{0, 1\}$

Stroj s páskovou abecedou Γ:



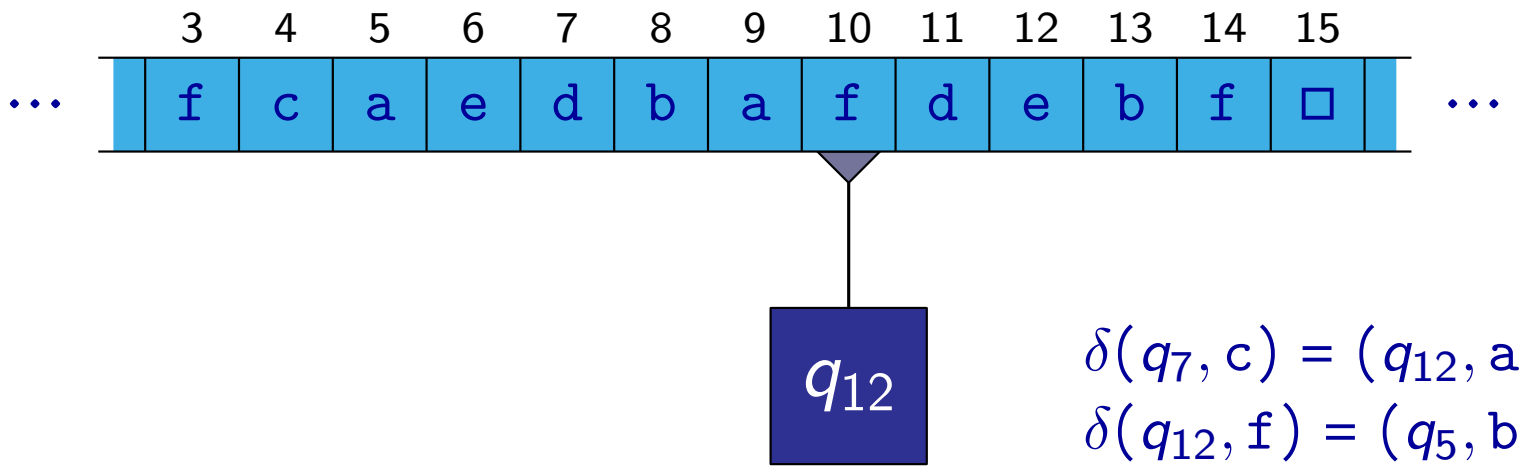
$\delta(q_7, \mathtt{c}) = (q_{12}, \mathtt{a}, +1)$
$\delta(q_{12}, \mathtt{f}) = (q_5, \mathtt{b}, -1)$

Stroj s abecedou $\{0, 1\}$:



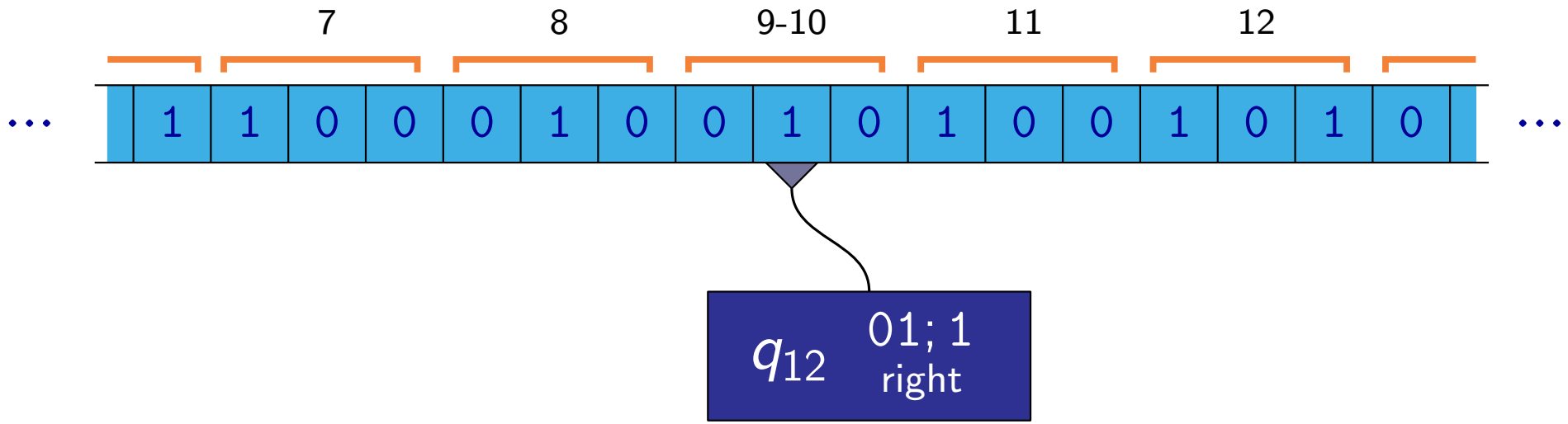
Abeceda $\{0, 1\}$

Stroj s páskovou abecedou Γ:



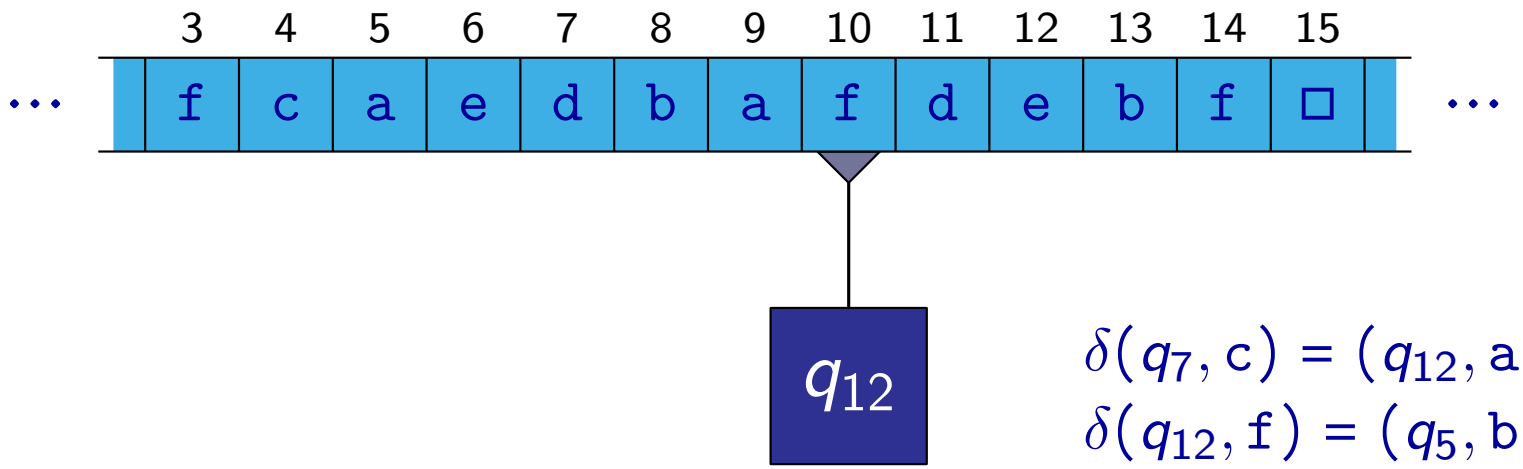
$$\delta(q_7, \mathtt{c}) = (q_{12}, \mathtt{a}, +1)$$
$$\delta(q_{12}, \mathtt{f}) = (q_5, \mathtt{b}, -1)$$

Stroj s abecedou $\{0, 1\}$:



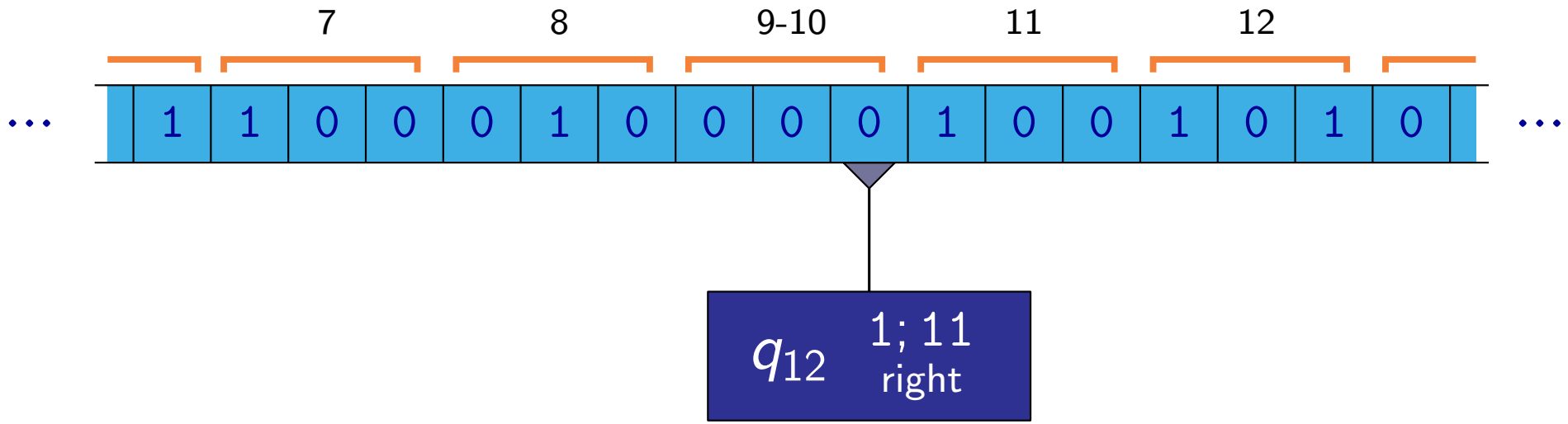
Abeceda $\{0, 1\}$

Stroj s páskovou abecedou Γ:



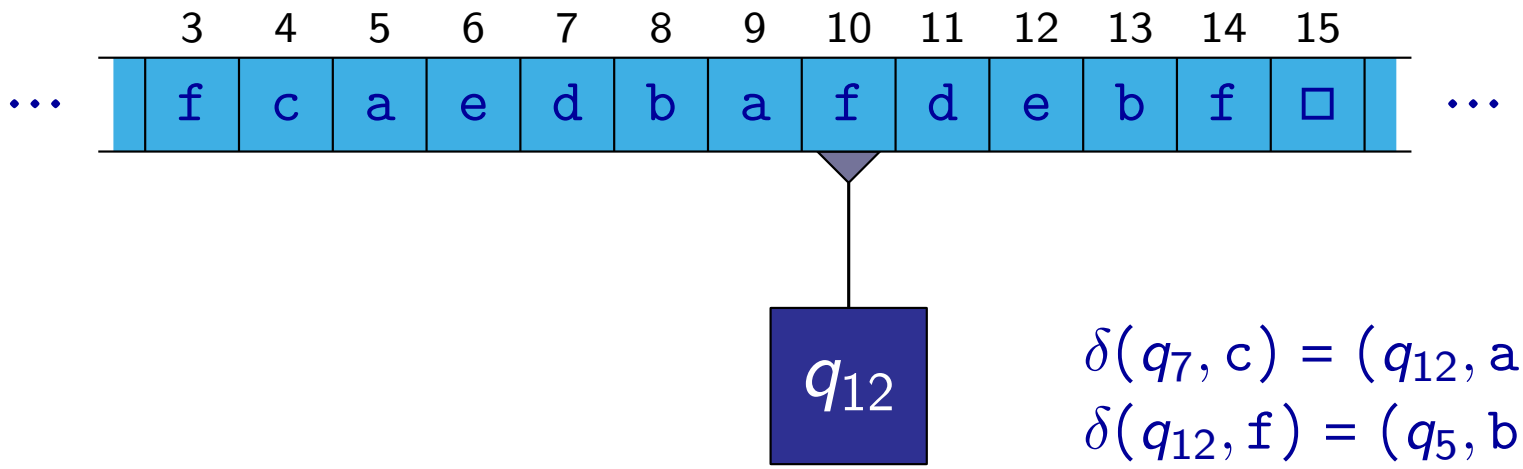
$$\delta(q_7, \mathtt{c}) = (q_{12}, \mathtt{a}, +1)$$
$$\delta(q_{12}, \mathtt{f}) = (q_5, \mathtt{b}, -1)$$

Stroj s abecedou $\{0, 1\}$:



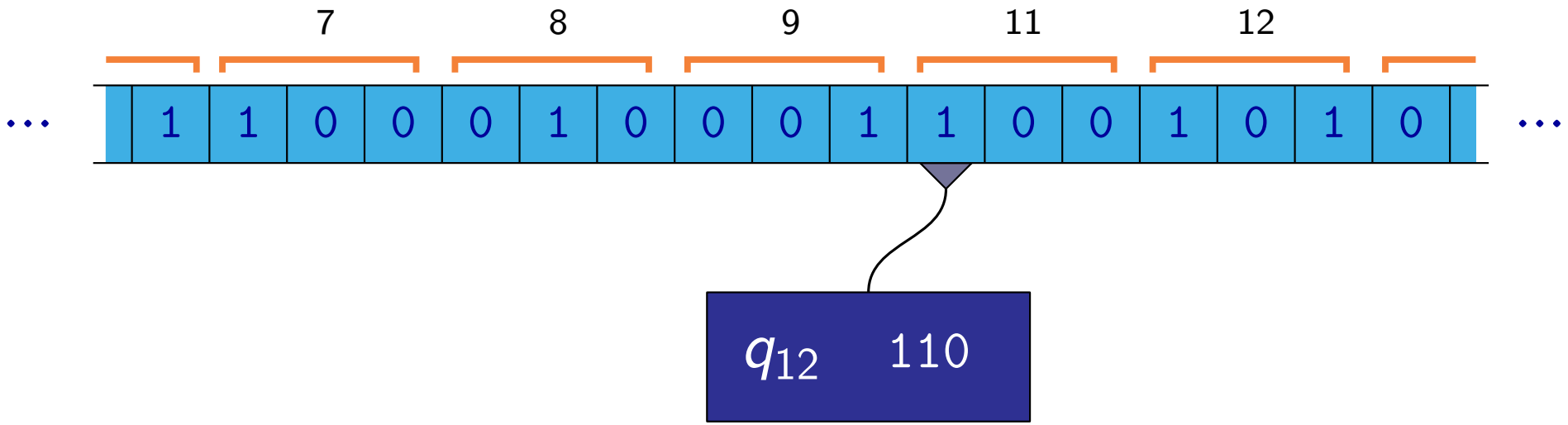
Abeceda $\{0, 1\}$

Stroj s páskovou abecedou Γ:



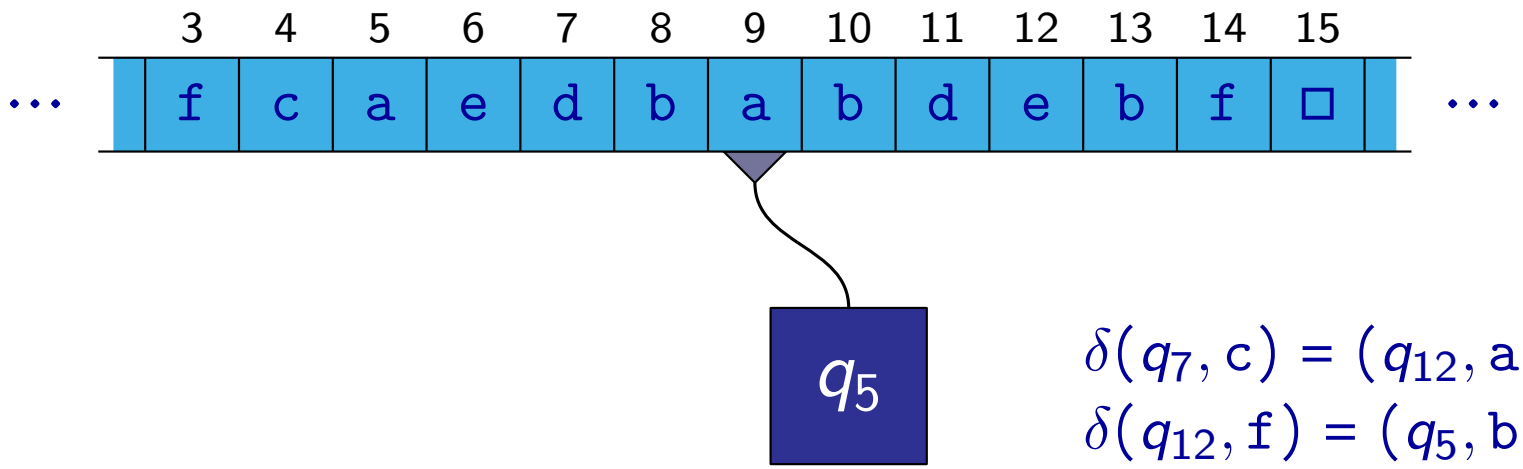
$\delta(q_7, \mathtt{c}) = (q_{12}, \mathtt{a}, +1)$
$\delta(q_{12}, \mathtt{f}) = (q_5, \mathtt{b}, -1)$

Stroj s abecedou $\{0, 1\}$:



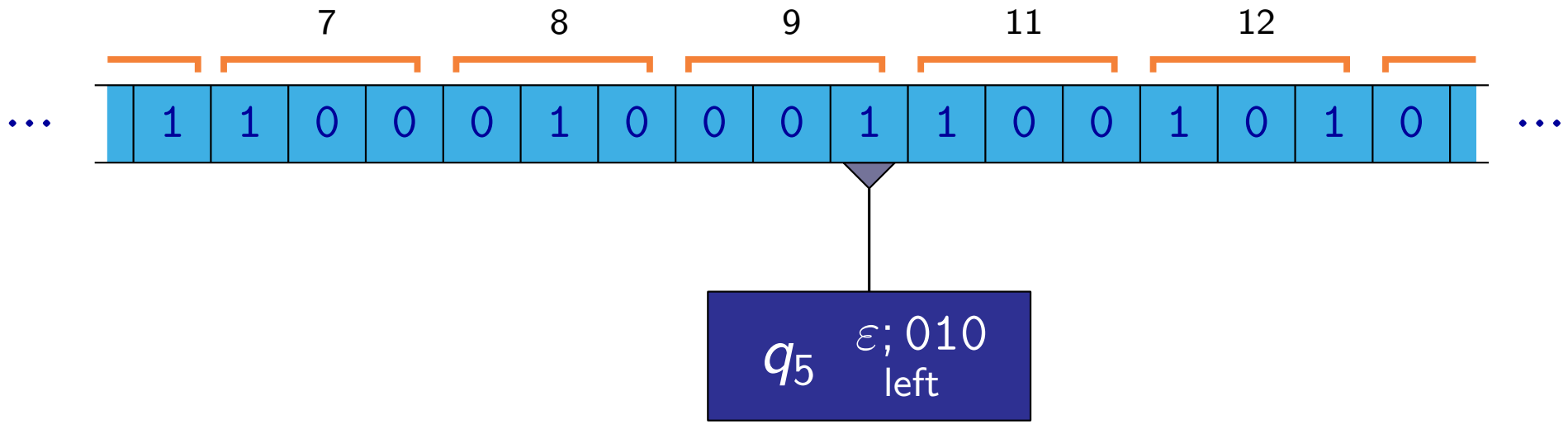
Abeceda $\{0, 1\}$

Stroj s páskovou abecedou Γ:



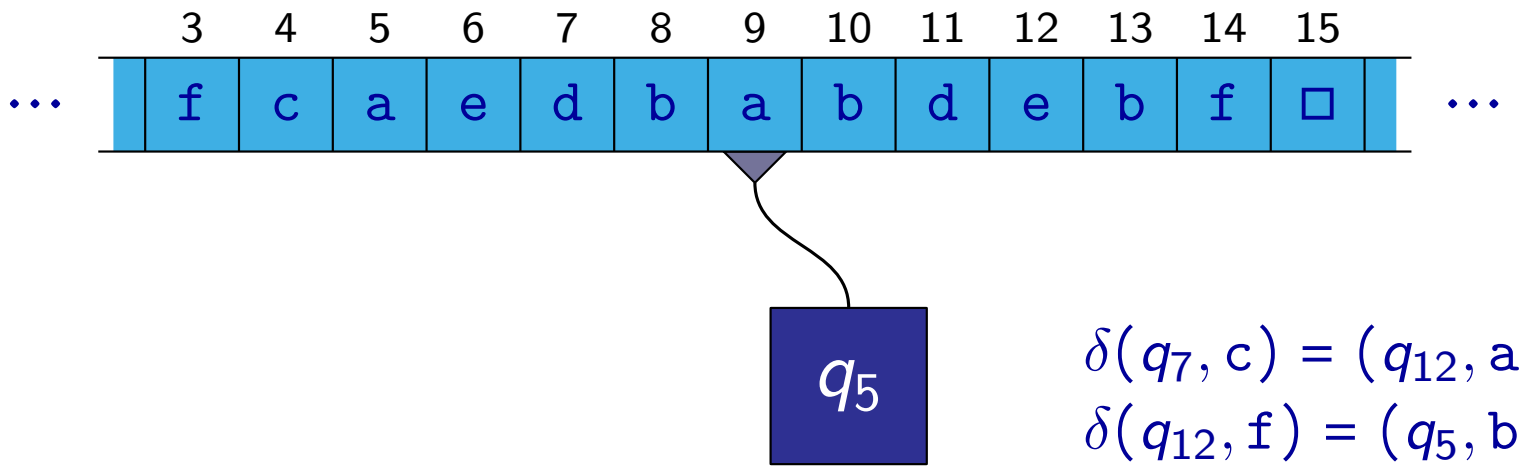
$$\delta(q_7, \mathtt{c}) = (q_{12}, \mathtt{a}, +1)$$
$$\delta(q_{12}, \mathtt{f}) = (q_5, \mathtt{b}, -1)$$

Stroj s abecedou $\{0, 1\}$:



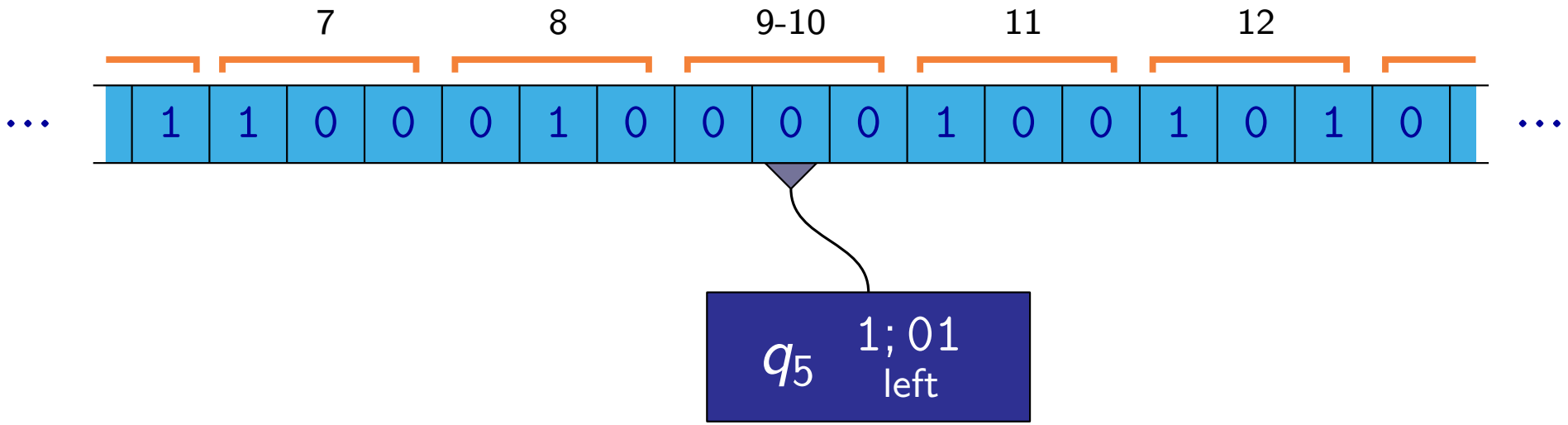
Abeceda $\{0, 1\}$

Stroj s páskovou abecedou Γ:



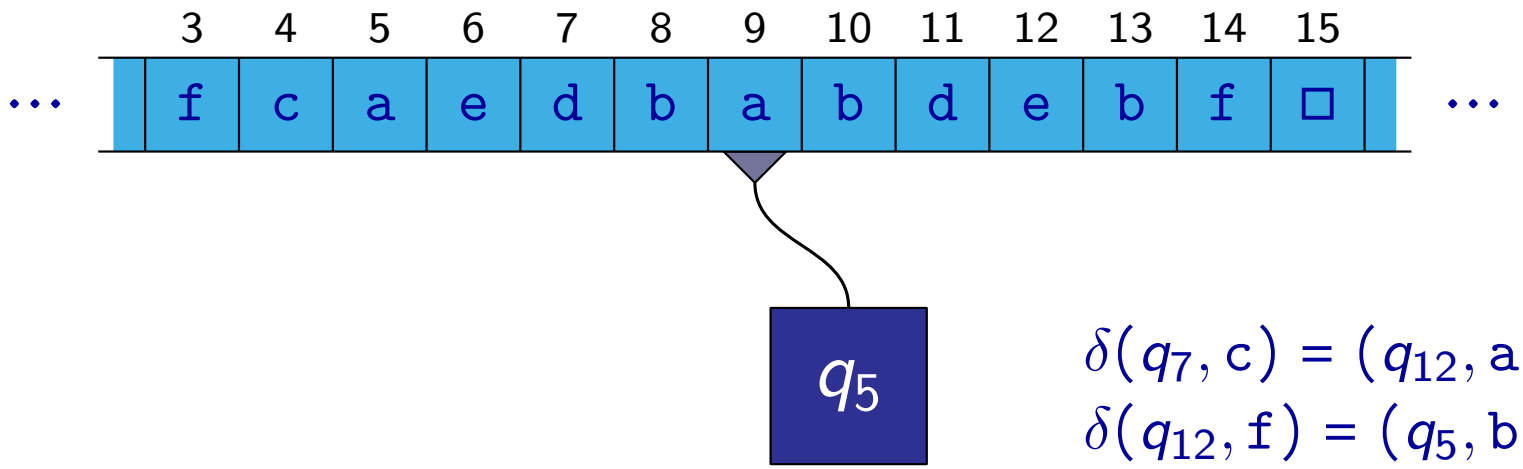
$$\delta(q_7, \mathtt{c}) = (q_{12}, \mathtt{a}, +1)$$
$$\delta(q_{12}, \mathtt{f}) = (q_5, \mathtt{b}, -1)$$

Stroj s abecedou $\{0, 1\}$:



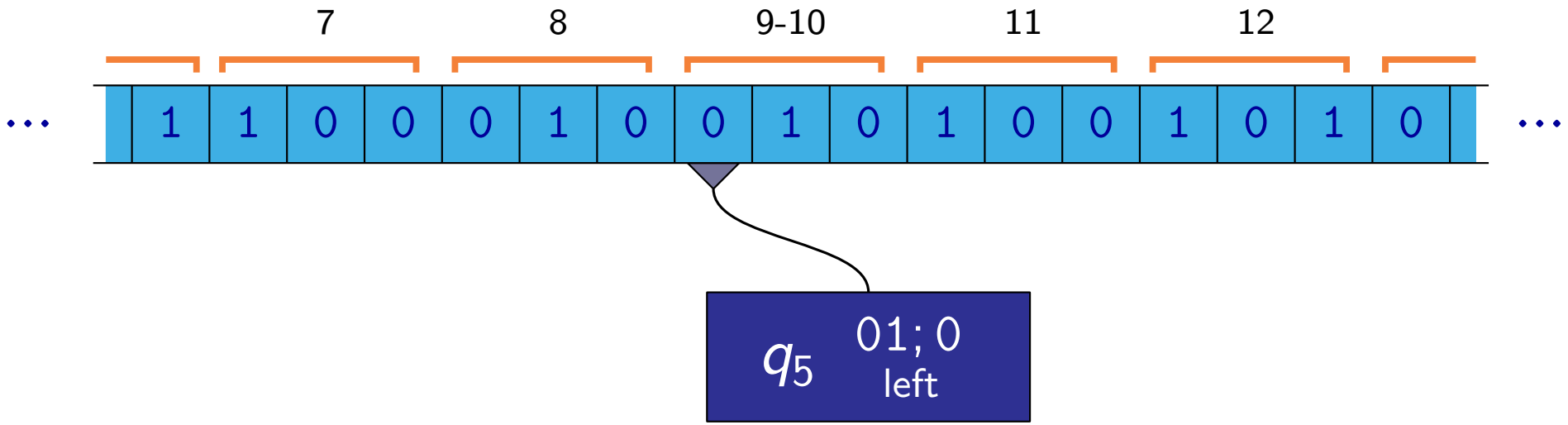
Abeceda $\{0, 1\}$

Stroj s páskovou abecedou Γ:



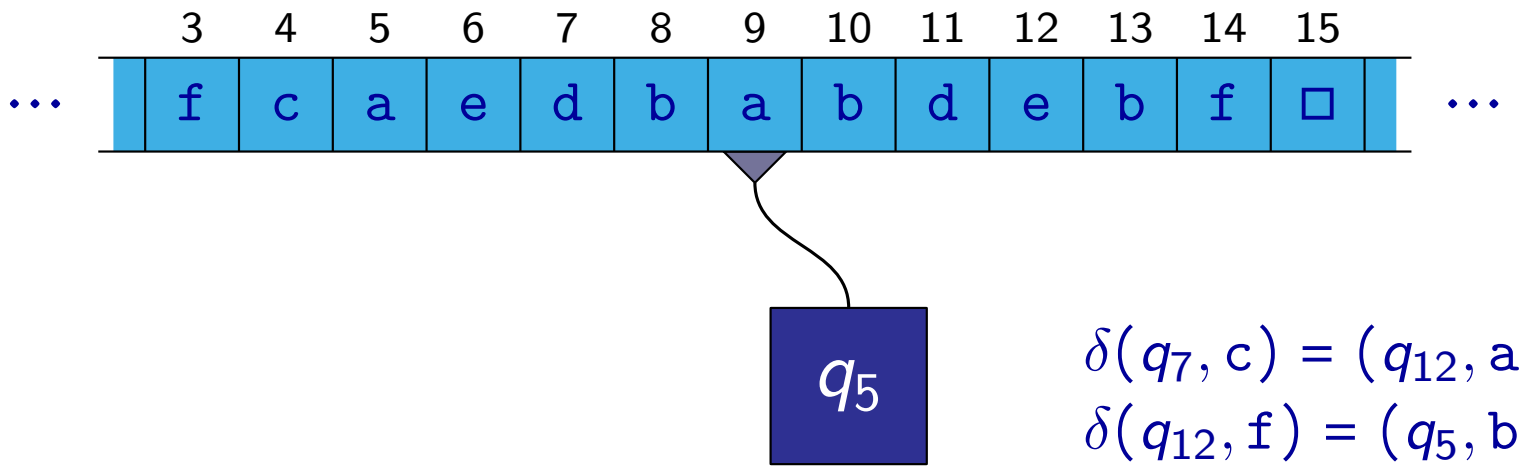
$$\delta(q_7, \mathtt{c}) = (q_{12}, \mathtt{a}, +1)$$
$$\delta(q_{12}, \mathtt{f}) = (q_5, \mathtt{b}, -1)$$

Stroj s abecedou $\{0, 1\}$:



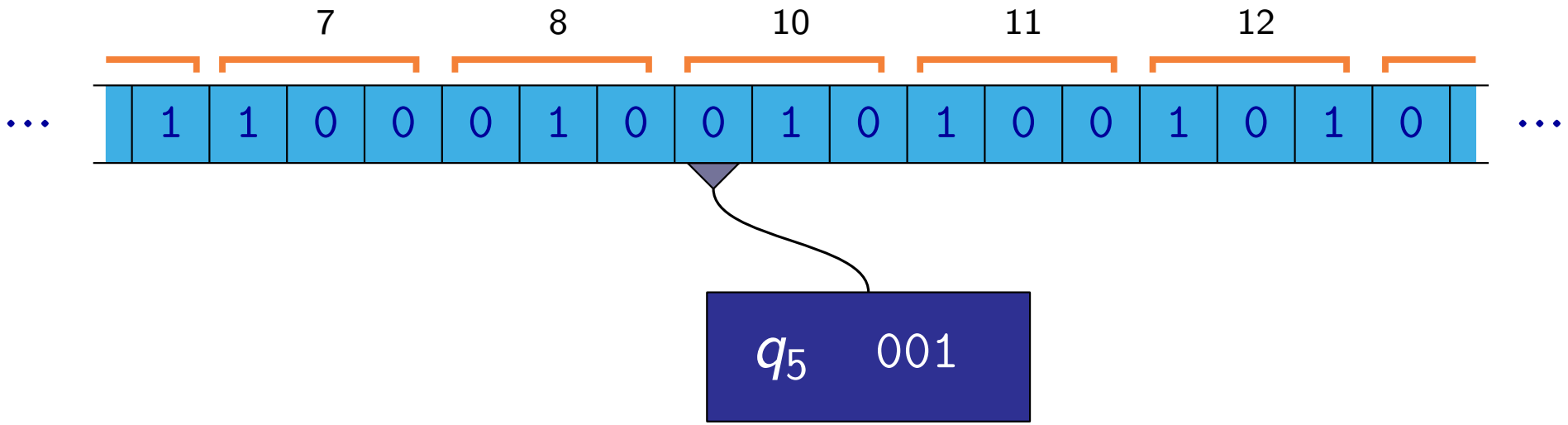
Abeceda $\{0, 1\}$

Stroj s páskovou abecedou Γ:



$$\delta(q_7, \mathtt{c}) = (q_{12}, \mathtt{a}, +1)$$
$$\delta(q_{12}, \mathtt{f}) = (q_5, \mathtt{b}, -1)$$

Stroj s abecedou $\{0, 1\}$:



Abeceda $\{0, 1\}$

Při výše uvedené simulaci je jeden krok původního stroje simulován $k + 1$ kroky, kde k je počet bitů kódující jeden symbol abecedy Γ.

Pokud tedy původní stroj provede během výpočtu t kroků, simulující stroj provede $O(t)$ kroků.

Zmenšení počtu stavů řídící jednotky

Poznámka: Tak, jako je možné zmenšit páskovou abecedu na pouhé dva symboly za cenu nárůstu velikosti počtu stavů řídící jednotky, je rovněž možné snížit počet stavů řídící jednotky:

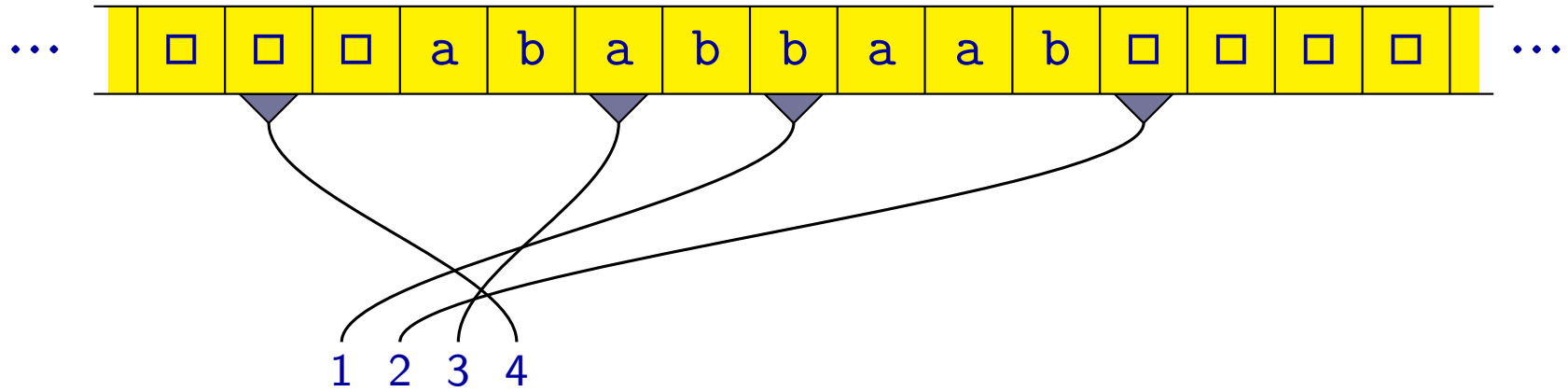
- Činnost libovolného Turingova stroje je možné simulovat Turingovým strojem, který má pouze dva nekoncové stavy řídící jednotky (a případně nějaké koncové stavy), ovšem za cenu nárůstu velikosti páskové abecedy.

Podobně jako v předchozím případě je jeden krok původního stroje simulován s kroky, kde s je konstanta závisející pouze na počtu stavů řídící jednotky původního stroje (tj. na velikosti množiny Q).

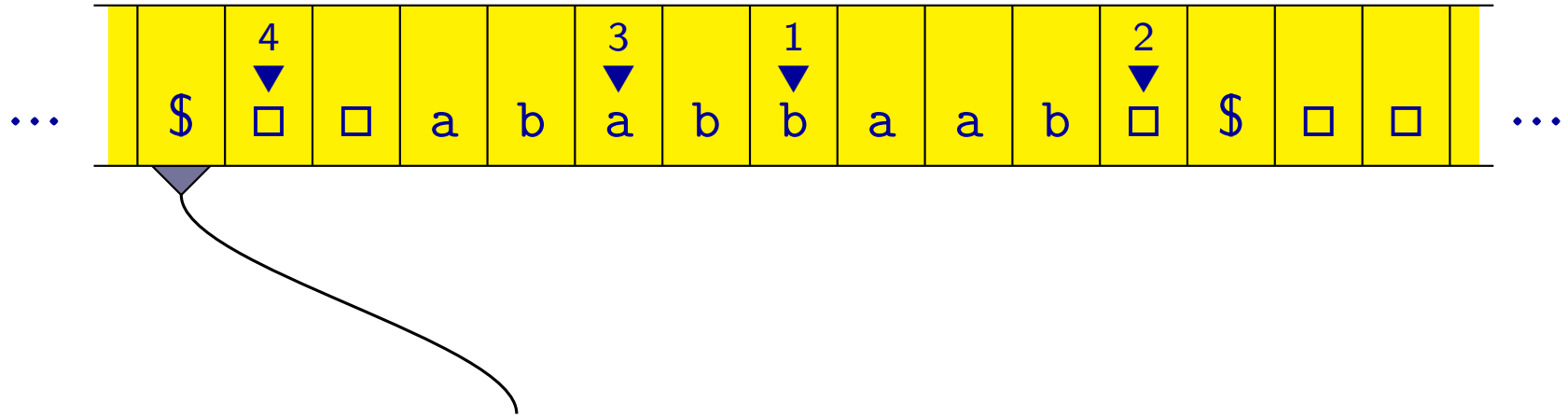
Opět zde tedy platí, že pokud původní stroj provede během výpočtu t kroků, simulující stroj provede $O(t)$ kroků.

Simulace více hlav na pásce pomocí jedné

Více hlav na pásce:

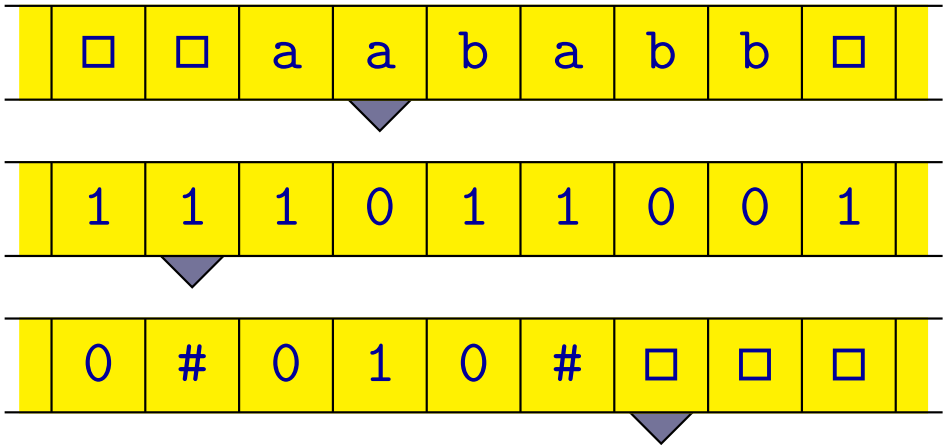


Páska s jednou hlavou:

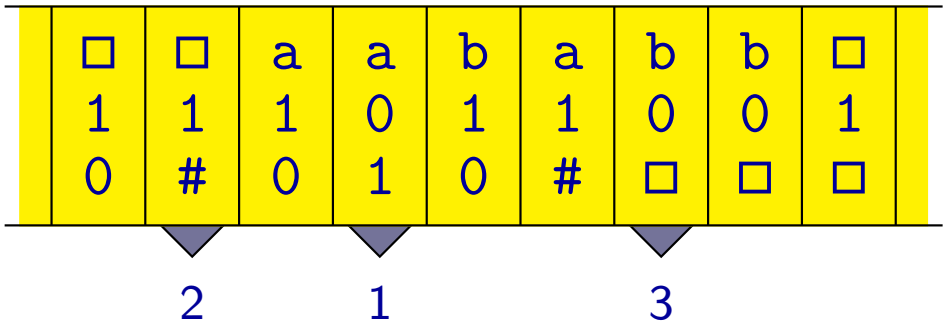


Simulace více pásek pomocí jedné

Více pásek:

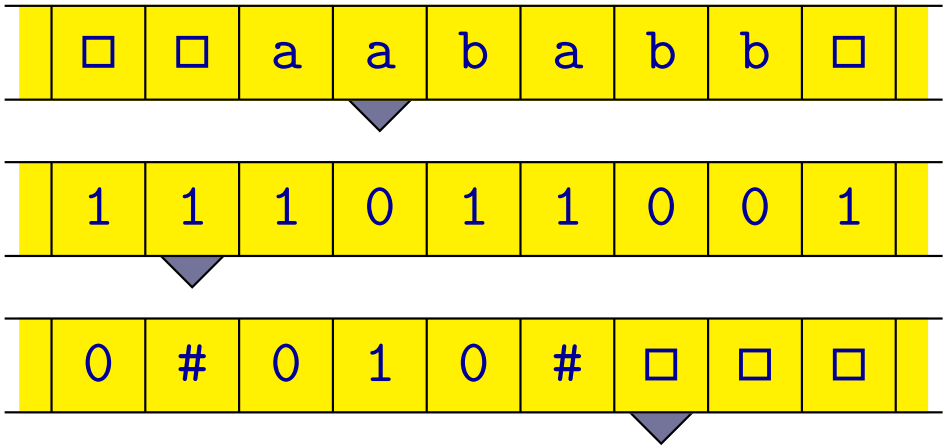


Jedna páska s více hlavami:

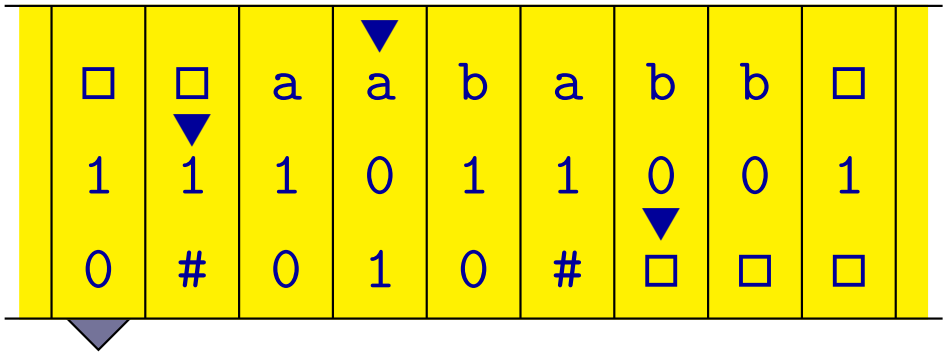


Simulace více pásek pomocí jedné

Více pásek:

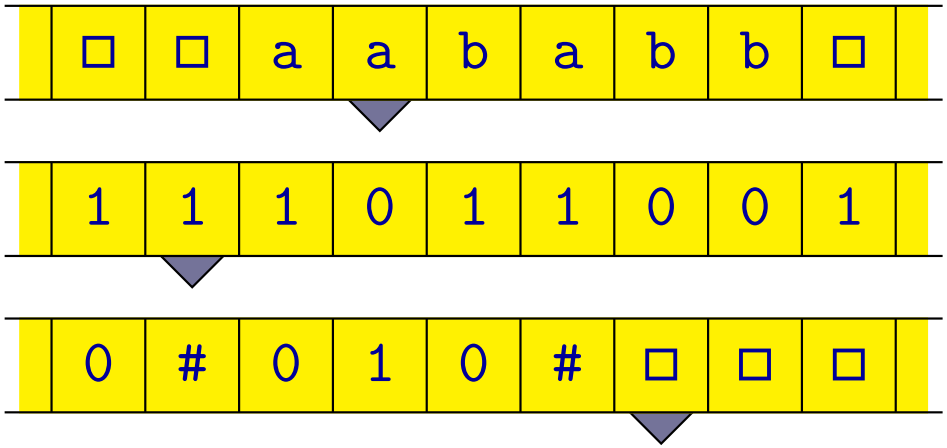


Jedna páska s jednou hlavou: varianta, kde se posunují značky hlav

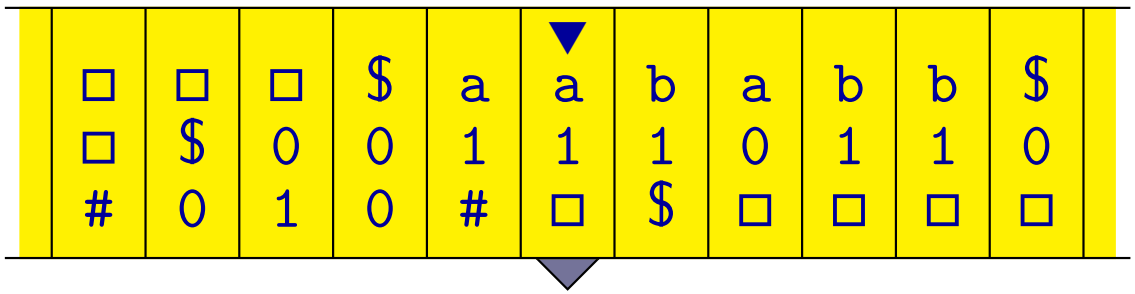


Simulace více pásek pomocí jedné

Více pásek:



Jedna páska s jednou hlavou: varianta, kde se posunují obsahy pásek



Pásky, zásobníky a čítače

Můžeme uvažovat různé stroje, které mají konečnou řídící jednotku doplněnou o nějaký druh neomezeně velké paměti.

Tato paměť může být tvořena jednou nebo více strukturami, jako jsou třeba:

- **Páska** — čtení a zápis symbolu na aktuální pozici, posun hlavy doleva a doprava

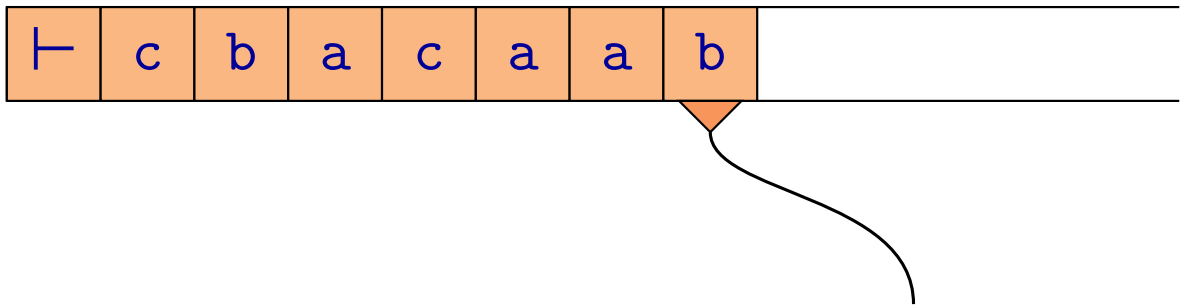
  Poznámka: Páska může být jednostranně nebo oboustranně nekonečná.

- **Zásobník** — push, pop, test prázdnosti zásobníku
- **Čítač** — hodnotou je přirozené číslo, operace přičtení nebo odečtení hodnoty jedna, test rovnosti nule

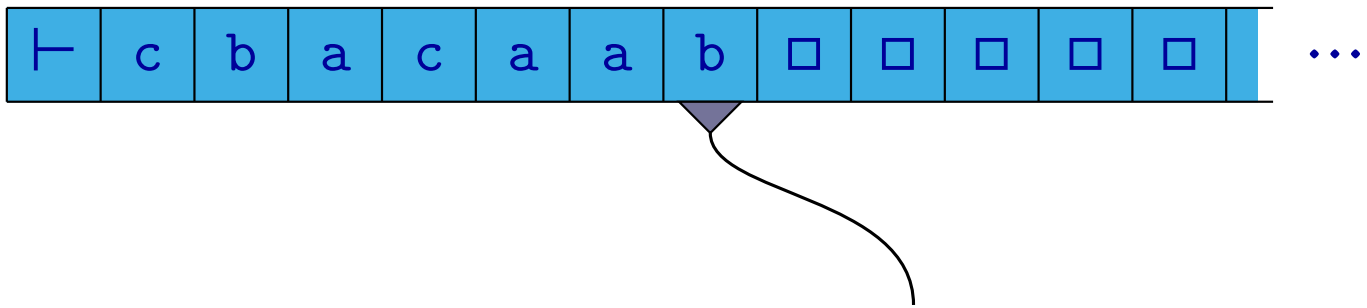
Zásobník

Na zásobník je možné se dívat jako na speciální případ jednostranně nekonečné pásky.

Zásobník:



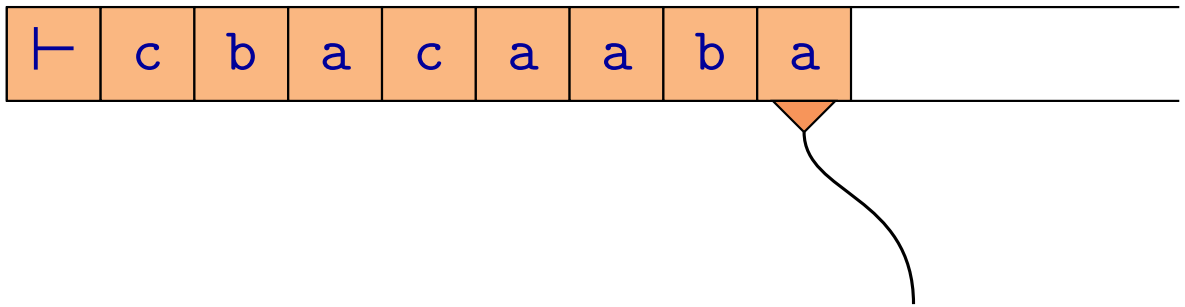
Páska:



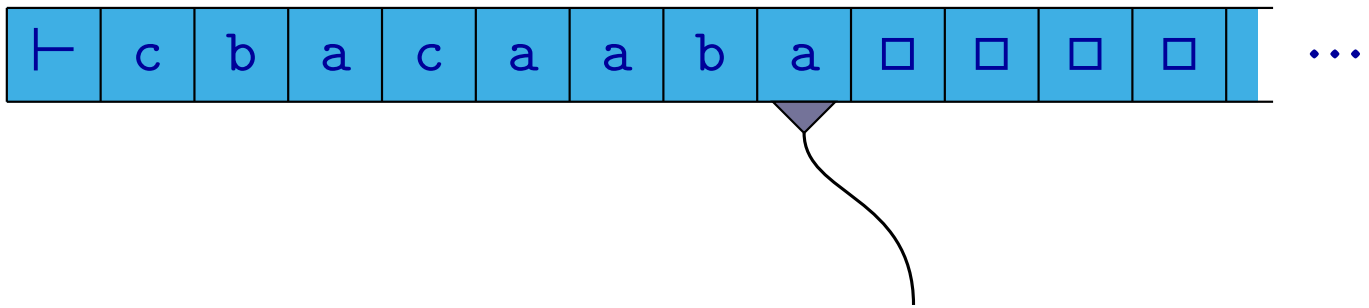
Zásobník

Na zásobník je možné se dívat jako na speciální případ jednostranně nekonečné pásky.

Zásobník:



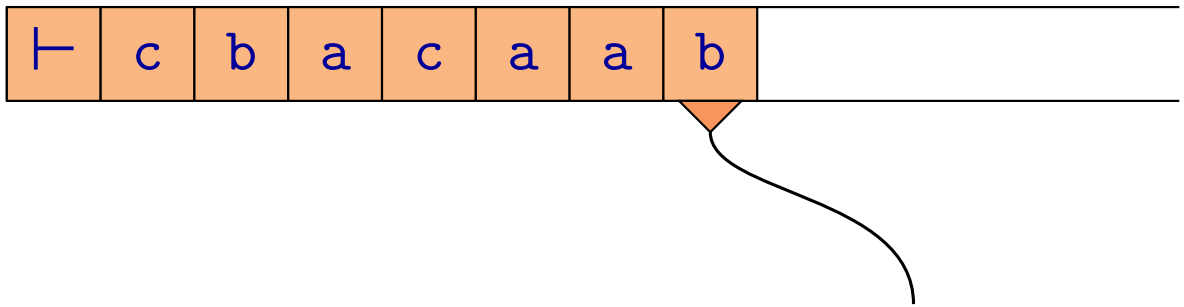
Páska:



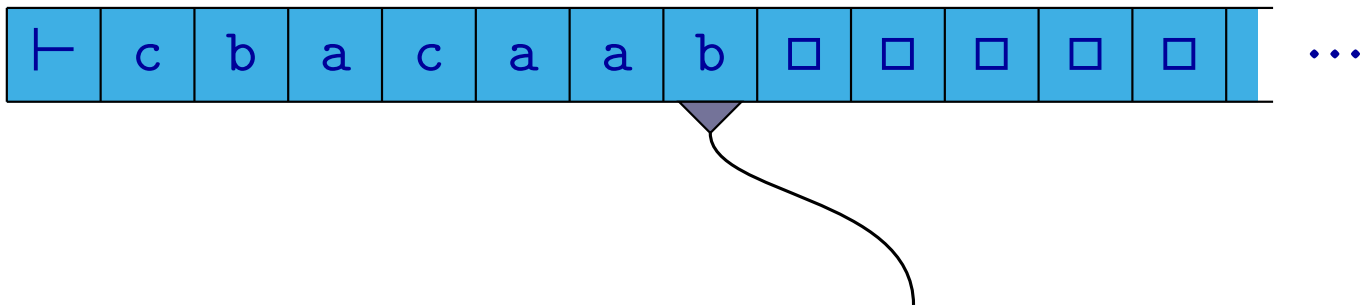
Zásobník

Na zásobník je možné se dívat jako na speciální případ jednostranně nekonečné pásky.

Zásobník:



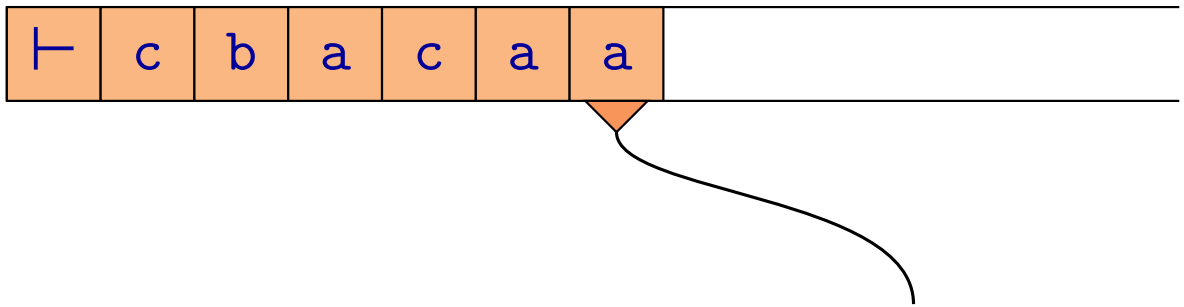
Páska:



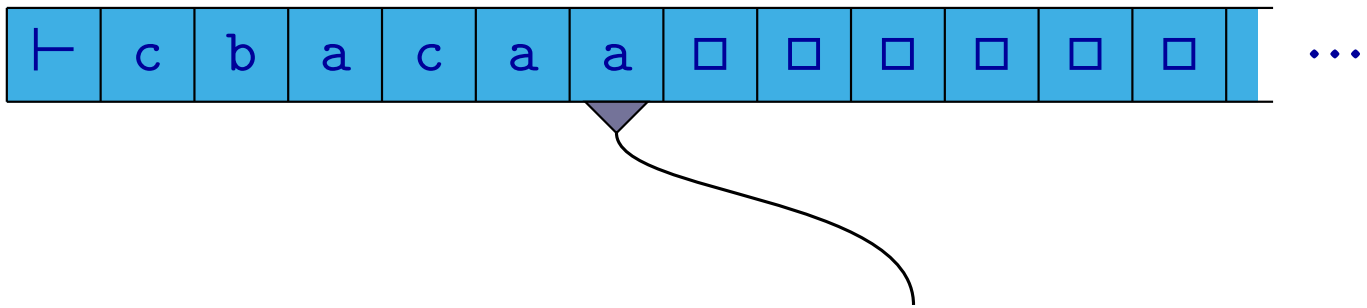
Zásobník

Na zásobník je možné se dívat jako na speciální případ jednostranně nekonečné pásky.

Zásobník:



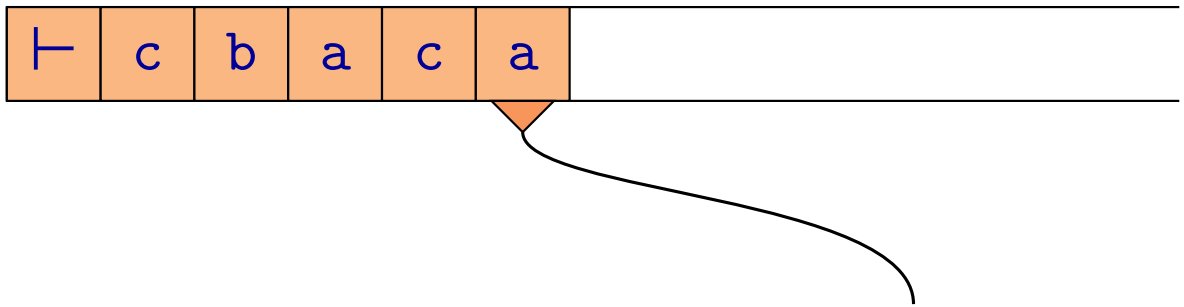
Páska:



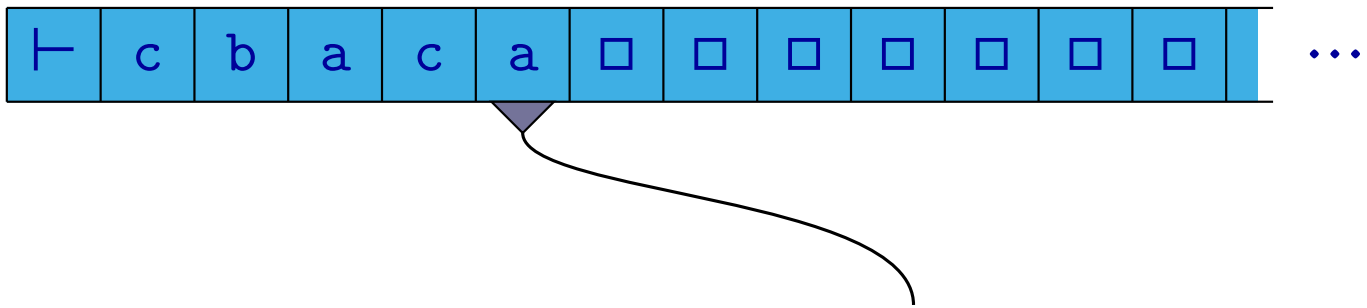
Zásobník

Na zásobník je možné se dívat jako na speciální případ jednostranně nekonečné pásky.

Zásobník:

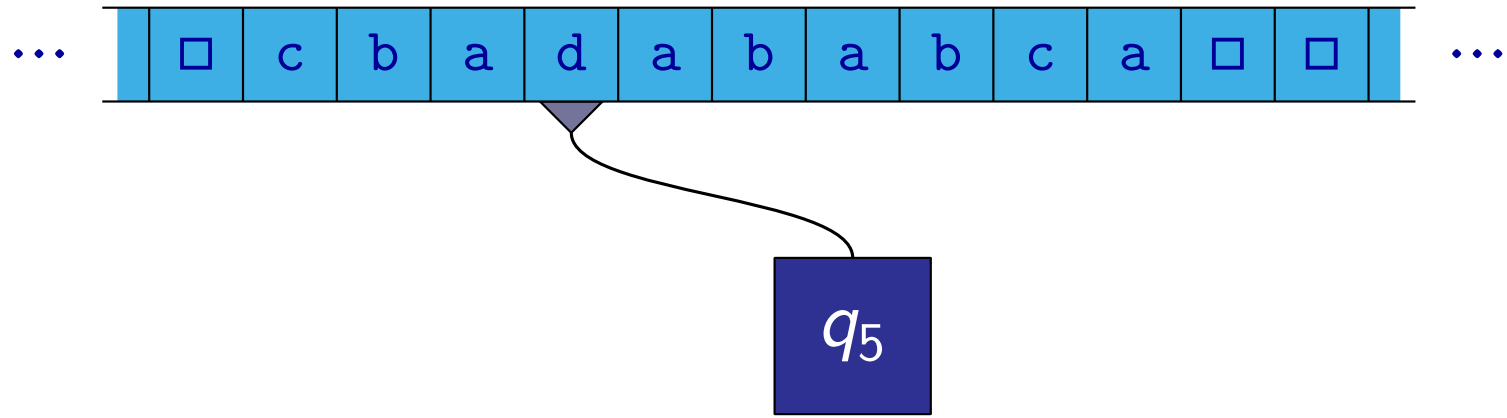


Páska:

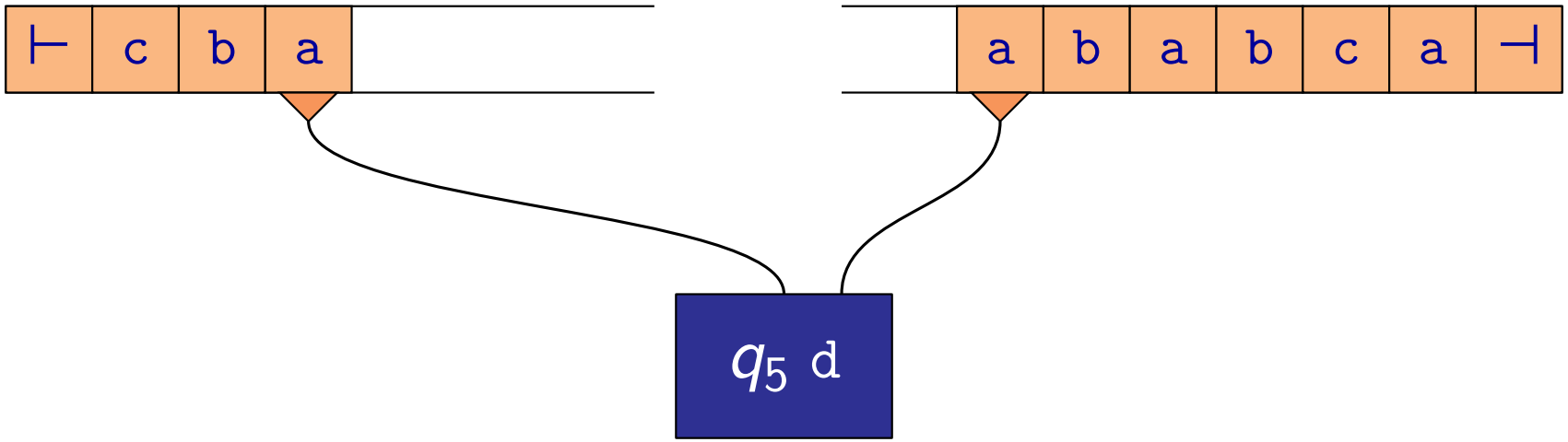


Zásobník

Oboustranně nekonečnou pásku je možné simulovat pomocí dvou zásobníků:

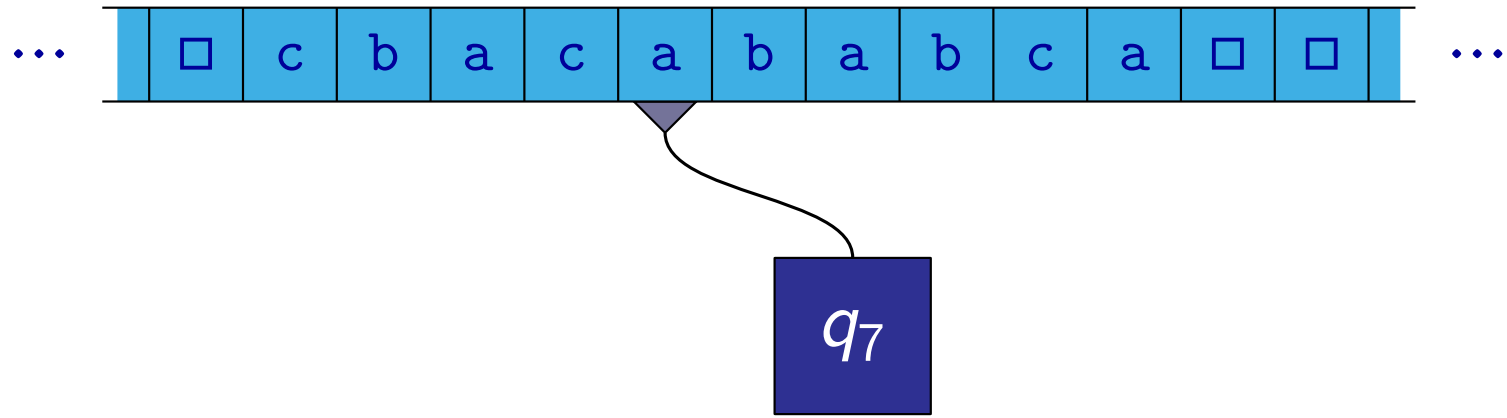


Stroj se dvěma zásobníky:

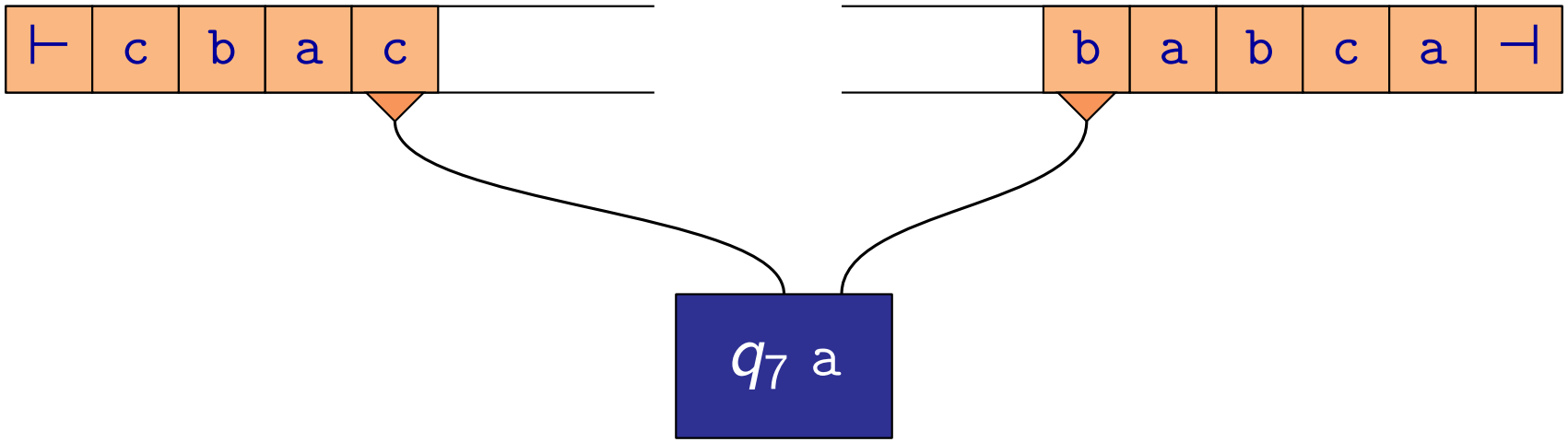


Zásobník

Oboustranně nekonečnou pásku je možné simulovat pomocí dvou zásobníků:



Stroj se dvěma zásobníky:



Čítač

Čítač — hodnotou čítače může být libovolně velké přirozené číslo, tj. prvek množiny $\mathbb{N} = \{0, 1, 2, 3, \ldots\}$.

Základní operace:

- zvýšení hodnoty o jedna:

$$x := x + 1$$

- snížení hodnoty o jedna:

$$x := x - 1$$

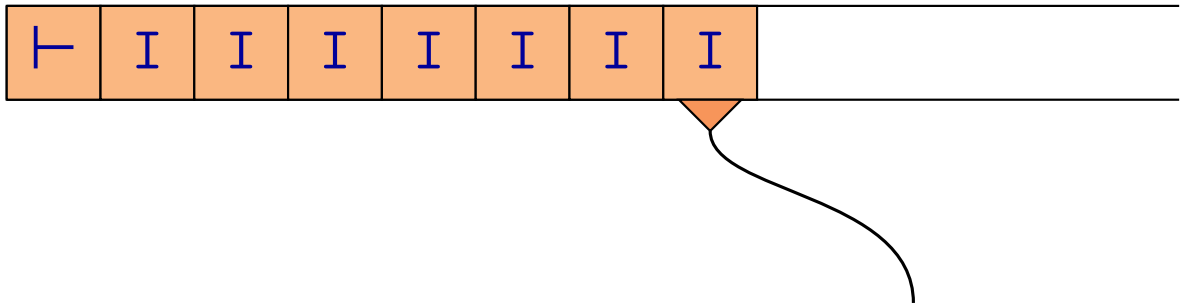
- test, jestli je hodnota čítače nula:

$$\textbf{if } (x = 0) \textbf{ goto } \ell$$

Čítač

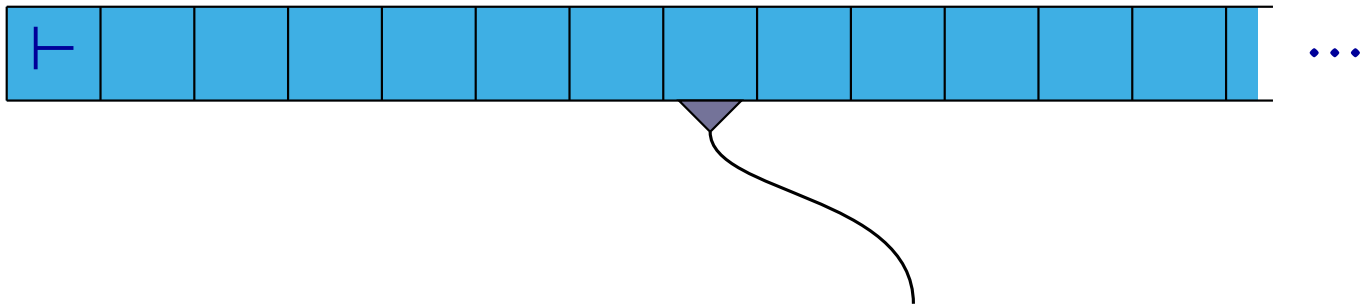
Na čítač je možné se dívat jako na speciální případ zásobníku či pásky.

Zásobník:



Čítač:

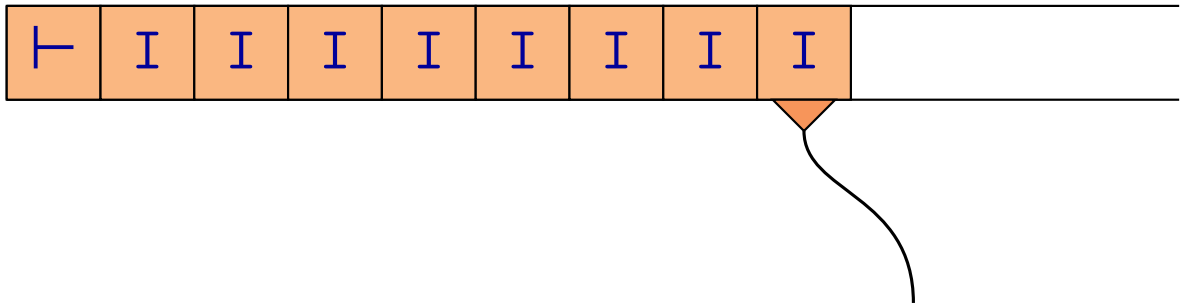
Páska:



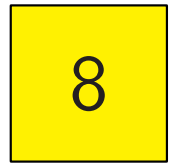
Čítač

Na čítač je možné se dívat jako na speciální případ zásobníku či pásky.

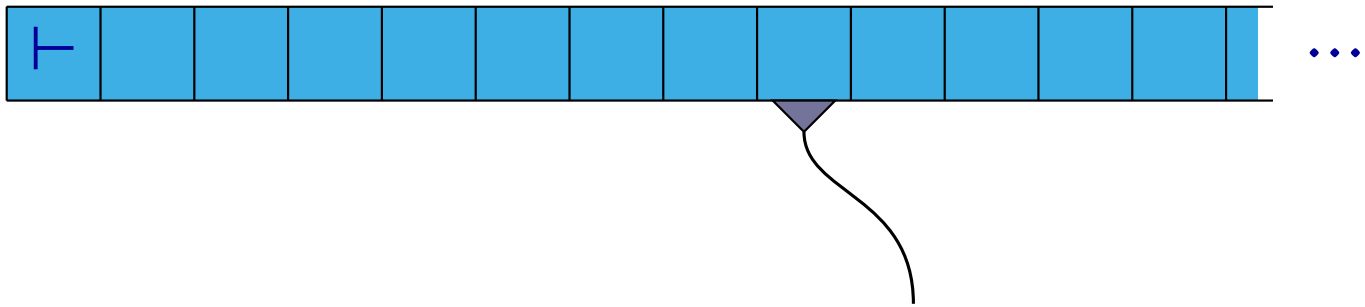
Zásobník:



Čítač:



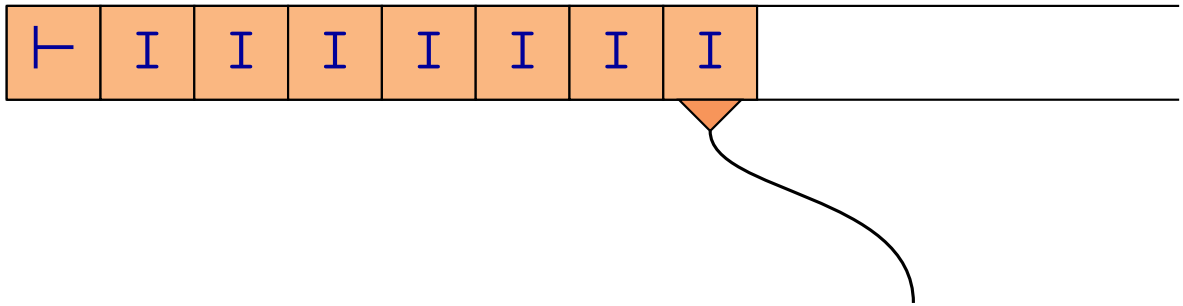
Páska:



Čítač

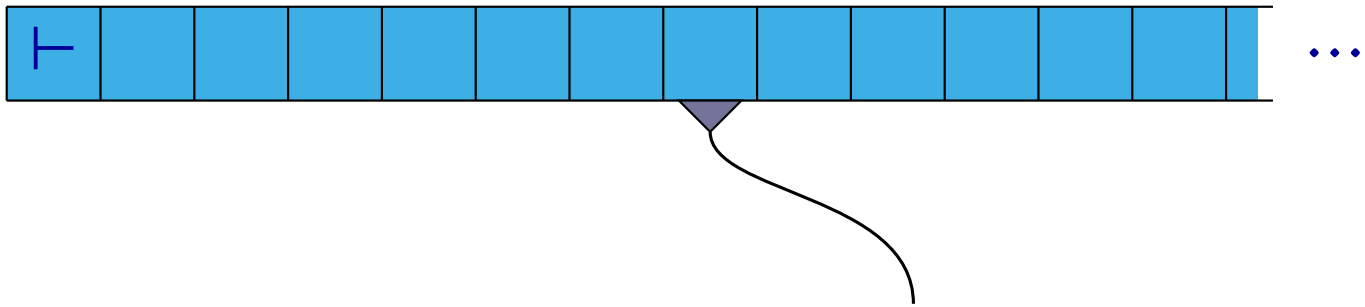
Na čítač je možné se dívat jako na speciální případ zásobníku či pásky.

Zásobník:



Čítač:

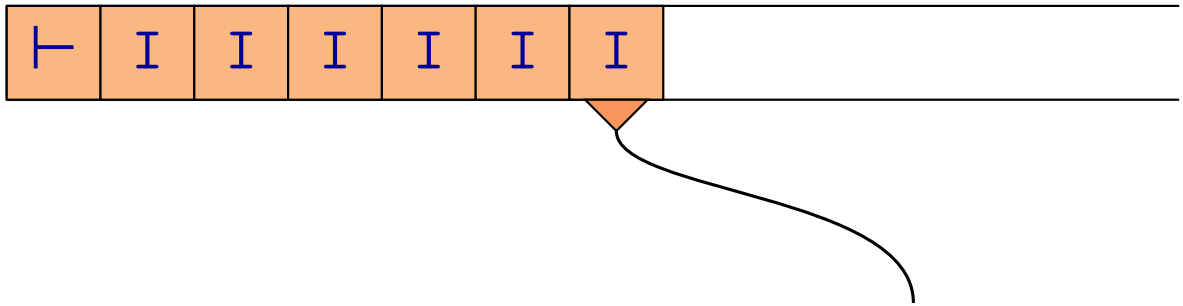
Páska:



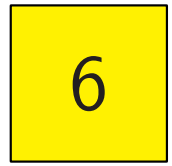
Čítač

Na čítač je možné se dívat jako na speciální případ zásobníku či pásky.

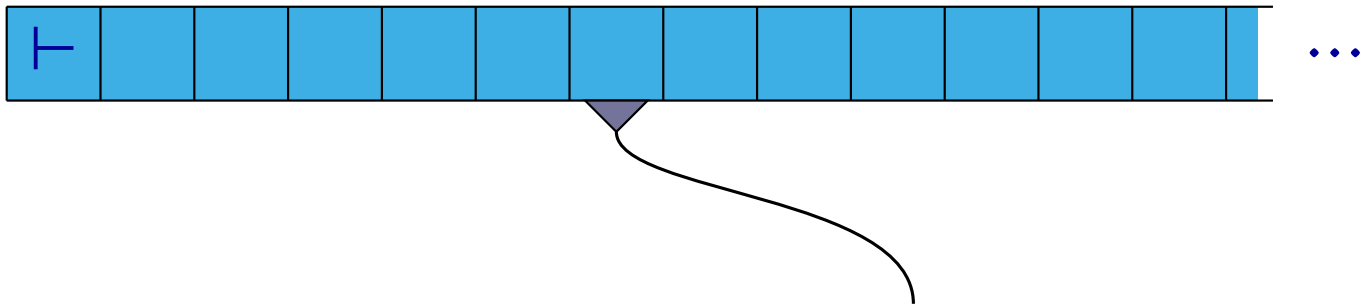
Zásobník:



Čítač:



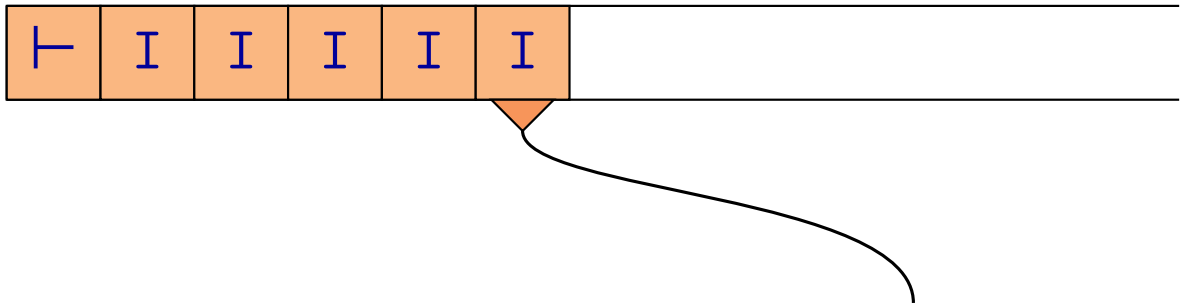
Páska:



Čítač

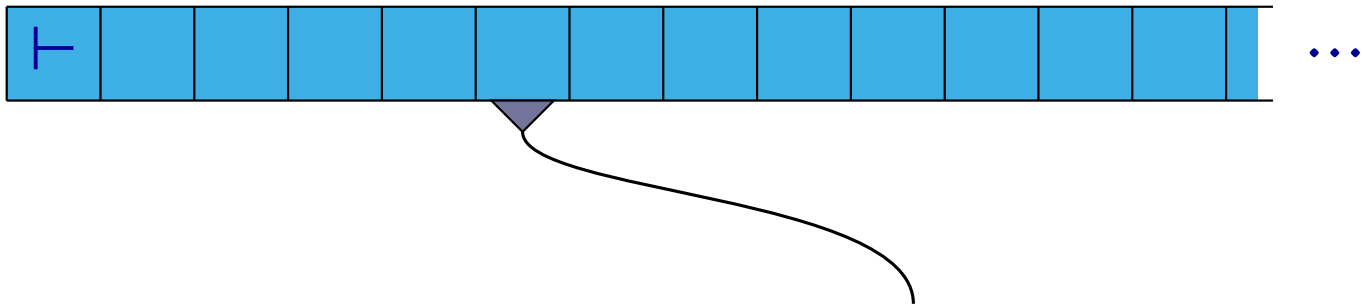
Na čítač je možné se dívat jako na speciální případ zásobníku či pásky.

Zásobník:



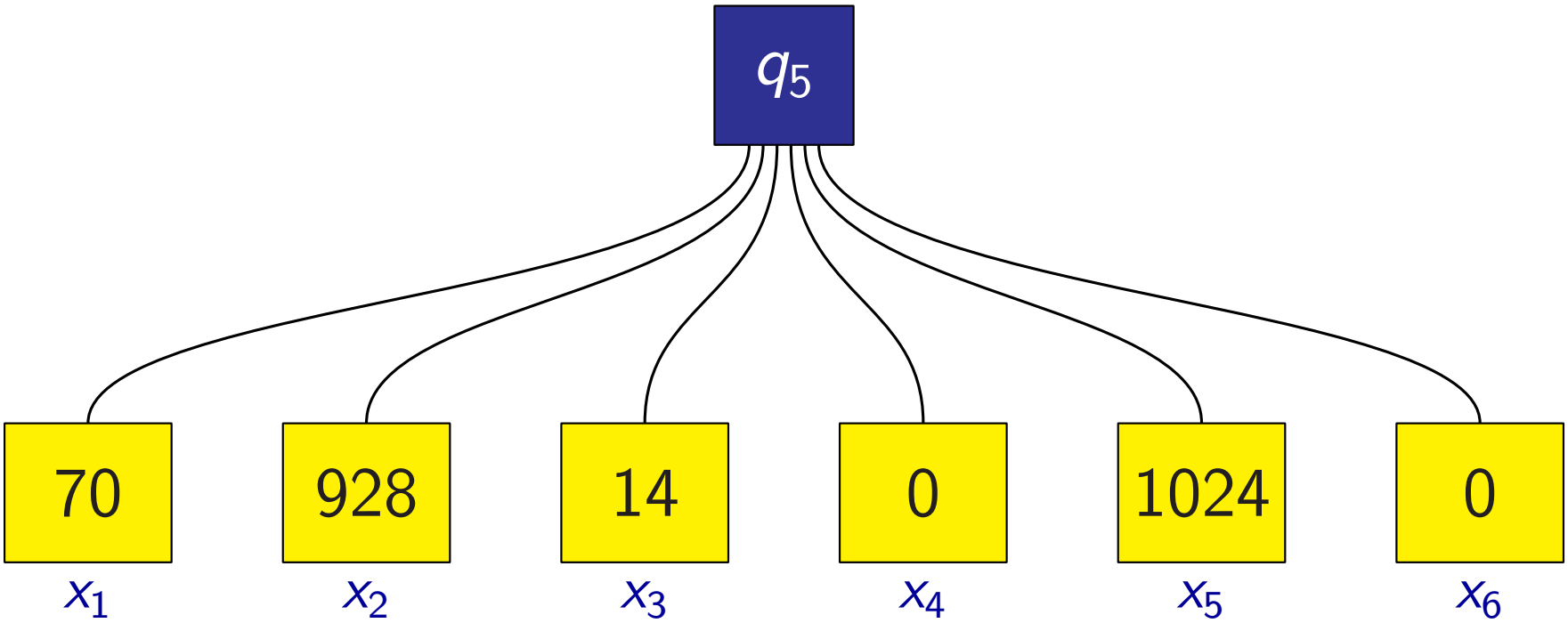
Čítač:

Páska:



Minského stroj

Minského stroj — stroj, který má konečnou řídící jednotku a konečný počet čítačů $x_1, x_2, \ldots, x_k$:



Poznámka: Pro označení čítačů budeme kromě symbolů $x_1, x_2, \ldots$ používat také symboly jako $x, y, z, \ldots$

Minského stroj

Na Minského stroj se můžeme dívat jako na program tvořený posloupností instrukcí následujících pěti typů:

- zvýšení hodnoty daného čítače o jedna:

$$x_i := x_i + 1$$

- snížení hodnoty daného čítače o jedna:

$$x_i := x_i - 1$$

- test, jestli je hodnota daného čítače nula:

$$\textbf{if } (x_i = 0) \textbf{ goto } \ell$$

- nepodmíněný skok:

$$\textbf{goto } \ell$$

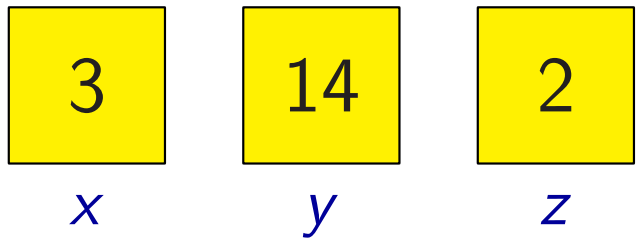
- zastavení programu:

$$\textbf{halt}$$

Minského stroj

Vynulování čítače x:

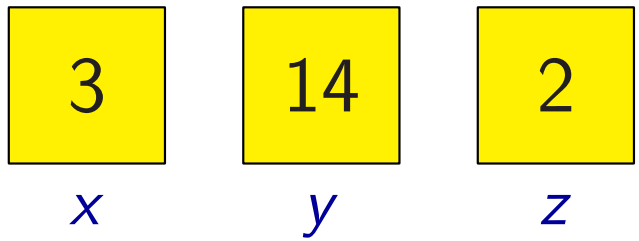
L_1 : **if** $(x = 0)$ **goto** L_2
$x := x - 1$
goto L_1
L_2 : ...



Minského stroj

Vynulování čítače x:

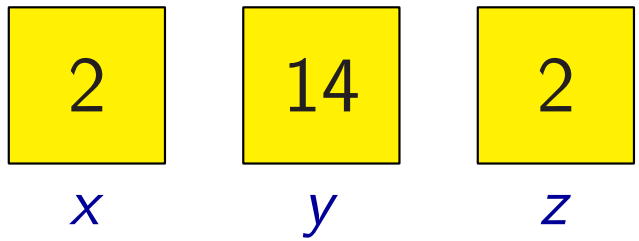
L_1 : **if** $(x = 0)$ **goto** L_2
$x := x - 1$
goto L_1
L_2 : ...



Minského stroj

Vynulování čítače x:

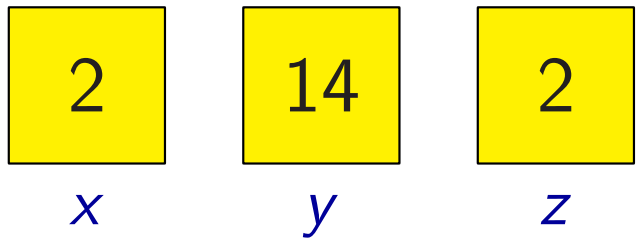
L_1 : **if** $(x = 0)$ **goto** L_2
$\quad x := x - 1$
$\quad$ **goto** L_1
L_2 : $\quad \ldots$



Minského stroj

Vynulování čítače x:

L_1 : **if** $(x = 0)$ **goto** L_2
$x := x - 1$
goto L_1
L_2 : ...



Minského stroj

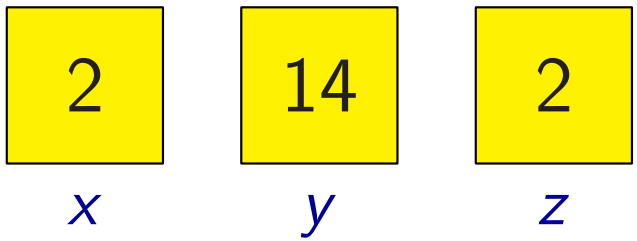
Vynulování čítače x:

L_1 : **if** $(x = 0)$ **goto** L_2

→ $x := x - 1$

goto L_1

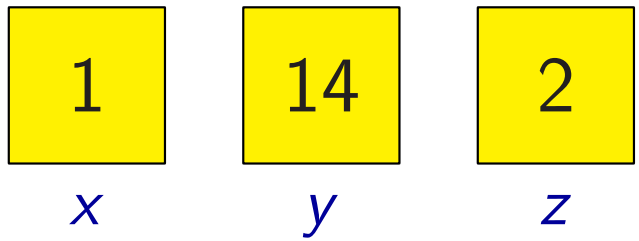
L_2 : ...



Minského stroj

Vynulování čítače x:

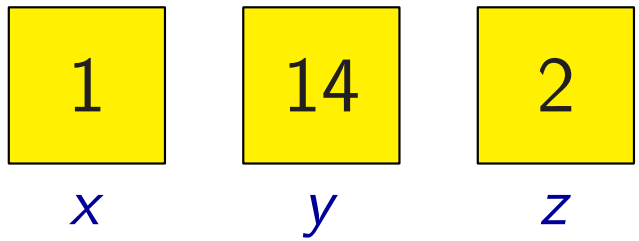
L_1 : **if** $(x = 0)$ **goto** L_2
$\quad x := x - 1$
$\quad$ **goto** L_1
L_2 : $\quad \ldots$



Minského stroj

Vynulování čítače x:

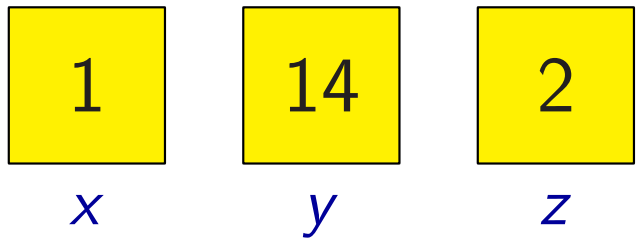
L_1 : **if** $(x = 0)$ **goto** L_2
$x := x - 1$
goto L_1
L_2 : ...



Minského stroj

Vynulování čítače x:

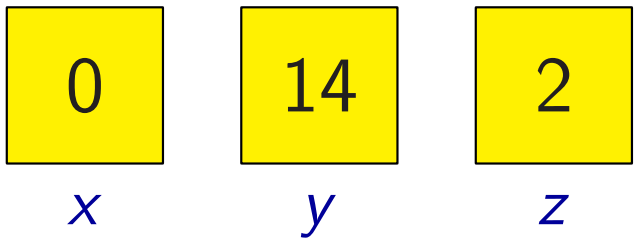
L_1 : **if** $(x = 0)$ **goto** L_2
$x := x - 1$
goto L_1
L_2 : ...



Minského stroj

Vynulování čítače x:

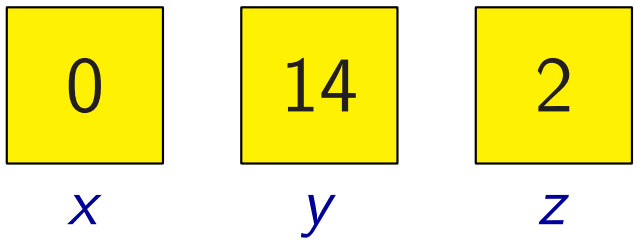
L_1 : **if** $(x = 0)$ **goto** L_2
$x := x - 1$
goto L_1
L_2 : ...



Minského stroj

Vynulování čítače x:

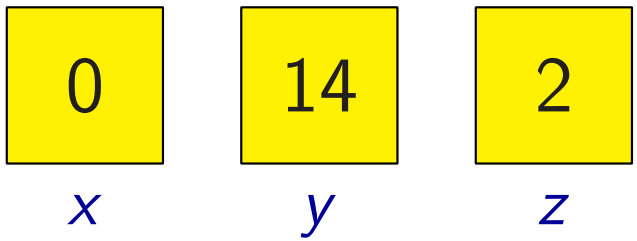
L_1 : **if** $(x = 0)$ **goto** L_2
$x := x - 1$
goto L_1
L_2 : ...



Minského stroj

Vynulování čítače x:

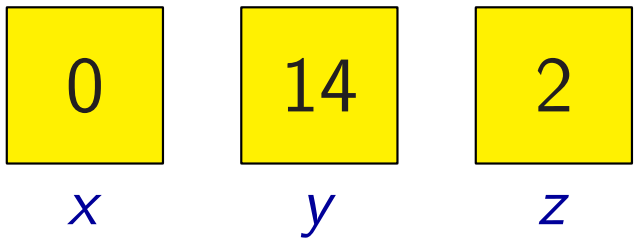
L_1 : **if** $(x = 0)$ **goto** L_2
$x := x - 1$
goto L_1
L_2 : $\ldots$



Minského stroj

Přičtení obsahu čítače z k čítači y (a vynulování čítače z):

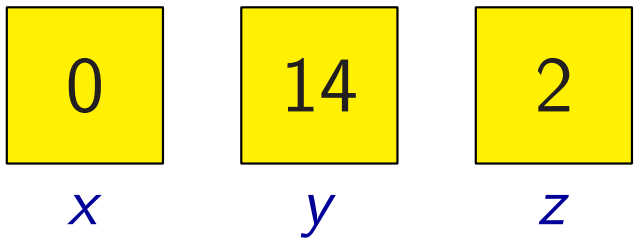
L_2 : **if** $(z = 0)$ **goto** L_3
$z := z - 1$
$y := y + 1$
goto L_1
L_3 : ...



Minského stroj

Přičtení obsahu čítače z k čítači y (a vynulování čítače z):

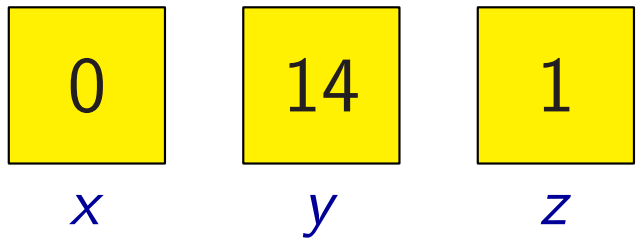
L_2 : **if** $(z = 0)$ **goto** L_3
$z := z - 1$
$y := y + 1$
goto L_1
L_3 : ...



Minského stroj

Přičtení obsahu čítače z k čítači y (a vynulování čítače z):

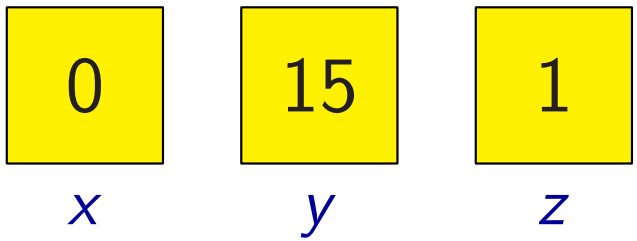
L_2 : **if** $(z = 0)$ **goto** L_3
$z := z - 1$
$y := y + 1$
goto L_1
L_3 : ...



Minského stroj

Přičtení obsahu čítače z k čítači y (a vynulování čítače z):

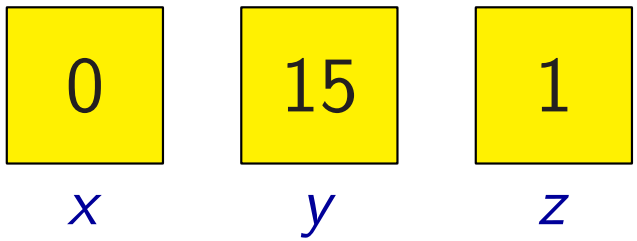
L_2 : **if** $(z = 0)$ **goto** L_3
$z := z - 1$
$y := y + 1$
goto L_1
L_3 : ...



Minského stroj

Přičtení obsahu čítače z k čítači y (a vynulování čítače z):

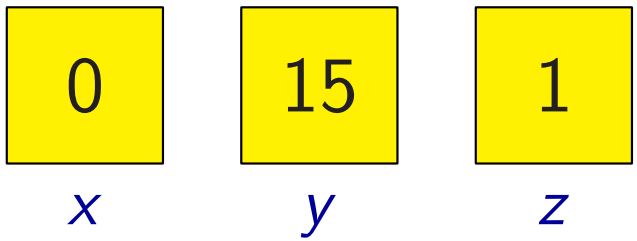
L_2 : **if** $(z = 0)$ **goto** L_3
$z := z - 1$
$y := y + 1$
goto L_1
L_3 : ...



Minského stroj

Přičtení obsahu čítače z k čítači y (a vynulování čítače z):

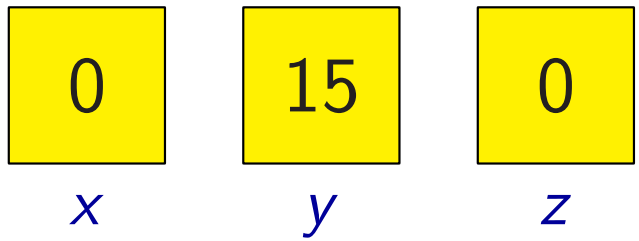
L_2 : **if** $(z = 0)$ **goto** L_3
→ $z := z - 1$
$y := y + 1$
goto L_1
L_3 : ...



Minského stroj

Přičtení obsahu čítače z k čítači y (a vynulování čítače z):

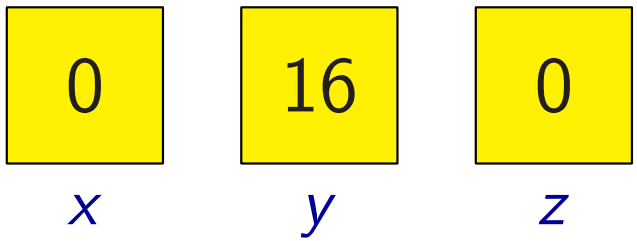
L_2 : **if** $(z = 0)$ **goto** L_3
$z := z - 1$
$y := y + 1$
goto L_1
L_3 : ...



Minského stroj

Přičtení obsahu čítače z k čítači y (a vynulování čítače z):

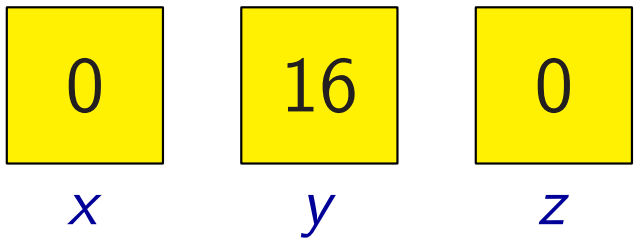
L_2 : **if** $(z = 0)$ **goto** L_3
$z := z - 1$
$y := y + 1$
goto L_1
L_3 : ...



Minského stroj

Přičtení obsahu čítače z k čítači y (a vynulování čítače z):

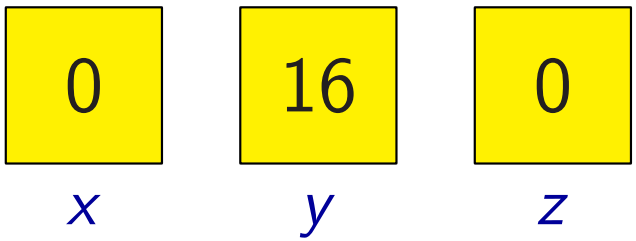
L_2 : **if** $(z = 0)$ **goto** L_3
$z := z - 1$
$y := y + 1$
goto L_1
L_3 : ...



Minského stroj

Přičtení obsahu čítače z k čítači y (a vynulování čítače z):

L_2 : **if** $(z = 0)$ **goto** L_3
$z := z - 1$
$y := y + 1$
goto L_1
L_3 : ...



Minského stroj

Vynásobení hodnoty čítače x číslem 5:

$$
\begin{aligned}
L_1 &: \textbf{if } (x = 0) \textbf{ goto } L_2 \\
&\quad x := x - 1 \\
&\quad y := y + 1 \\
&\quad y := y + 1 \\
&\quad y := y + 1 \\
&\quad y := y + 1 \\
&\quad y := y + 1 \\
&\quad \textbf{goto } L_1 \\
L_2 &: \textbf{if } (y = 0) \textbf{ goto } L_3 \\
&\quad y := y - 1 \\
&\quad x := x + 1 \\
&\quad \textbf{goto } L_2 \\
L_3 &: \quad \ldots
\end{aligned}
$$

Minského stroj

Vydělení hodnoty čítače x číslem 5 a zjištění zbytku po dělení:

$$
\begin{aligned}
L_1 : \ & \textbf{if } (x = 0) \textbf{ goto } M_0 \\
& x := x - 1 \\
& \textbf{if } (x = 0) \textbf{ goto } M_1 \\
& x := x - 1 \\
& \textbf{if } (x = 0) \textbf{ goto } M_2 \\
& x := x - 1 \\
& \textbf{if } (x = 0) \textbf{ goto } M_3 \\
& x := x - 1 \\
& \textbf{if } (x = 0) \textbf{ goto } M_4 \\
& x := x - 1 \\
& y := y + 1 \\
& \textbf{goto } L_1
\end{aligned}
$$

Minského stroj

Zásobník je možné simulovat pomocí dvou čítačů — hodnota jednoho čítače reprezentuje obsah zásobníku jako číslo, jehož zápis v číselné soustavě o základu $k = |\Gamma| + 1$ (kde Γ je zásobníková abeceda) odpovídá obsahu zásobníku.

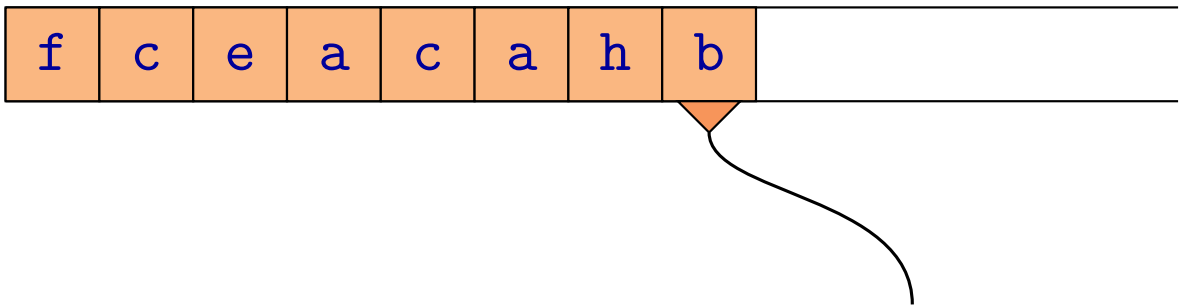
- Symbol na vrcholu zásobníku — zbytek po dělení číslem k
- Pop — vydělit číslem k
- Push — vynásobit číslem k a přičíst kód příslušného symbolu

Druhý čítač slouží jako pomocný při provádění výše uvedených operací.

Minského stroj

Příklad:

a ↔ 1
b ↔ 2
c ↔ 3
d ↔ 4
e ↔ 5
f ↔ 6
g ↔ 7
h ↔ 8
i ↔ 9

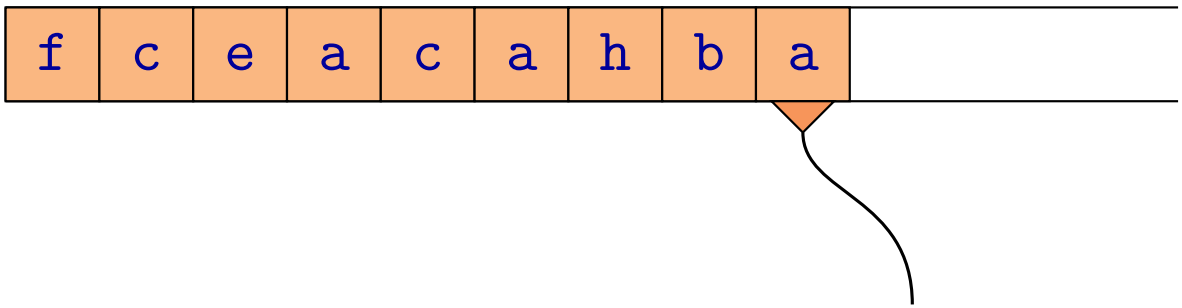


63513182

Minského stroj

Příklad:

a ↔ 1
b ↔ 2
c ↔ 3
d ↔ 4
e ↔ 5
f ↔ 6
g ↔ 7
h ↔ 8
i ↔ 9

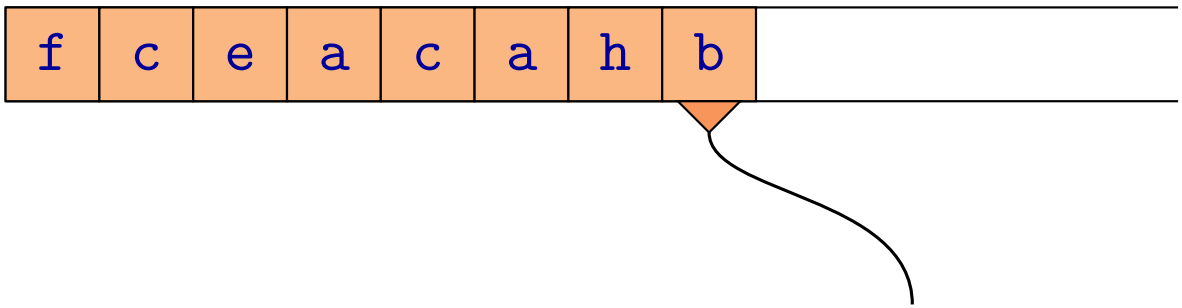


635131821

Minského stroj

Příklad:

a ↔ 1
b ↔ 2
c ↔ 3
d ↔ 4
e ↔ 5
f ↔ 6
g ↔ 7
h ↔ 8
i ↔ 9

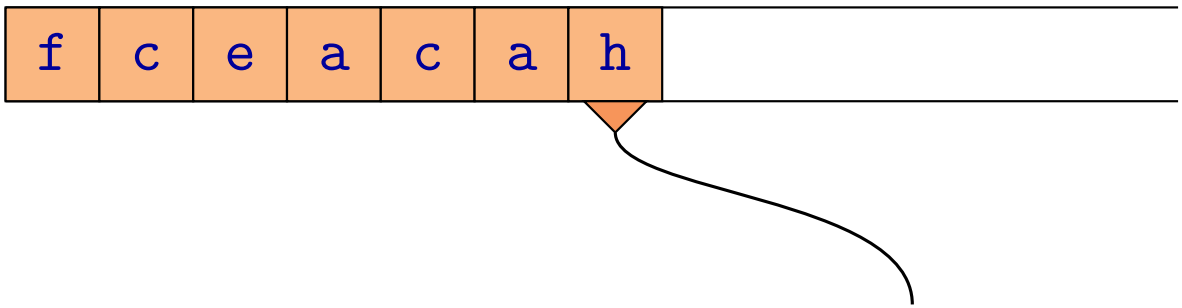


63513182

Minského stroj

Příklad:

a ↔ 1
b ↔ 2
c ↔ 3
d ↔ 4
e ↔ 5
f ↔ 6
g ↔ 7
h ↔ 8
i ↔ 9

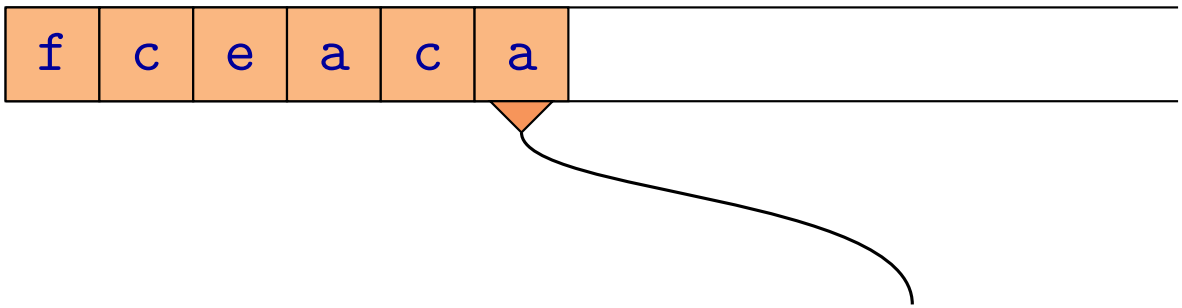


6351318

Minského stroj

Příklad:

a ↔ 1
b ↔ 2
c ↔ 3
d ↔ 4
e ↔ 5
f ↔ 6
g ↔ 7
h ↔ 8
i ↔ 9



635131

Minského stroj

Připomeňme, že oboustranně nekonečnou pásku je možné simulovat pomocí dvou zásobníků.

V Minského stroji může být obsah každého z těchto zásobníků reprezentován jemu odpovídajícím čítačem.

Navíc potřebujeme ještě jeden pomocný čítač pro implementaci operací násobení a dělení na těchto čítačích reprezentujících obsahy zásobníků.

Vidíme, že Turingův stroj s k páskami je možné simulovat Minského strojem s $2k + 1$ čítači.

Minského stroj

Libovolný konečný počet čítačů je možné simulovat pomocí dvou čítačů.

- Jeden čítač (označme jej C) reprezentuje hodnoty všech čítačů — např. hodnoty tří čítačů x, y, z mohou být v čítači C reprezentovány jako číslo $2^x 3^y 5^z$.
- Druhý čítač je používán jako pomocný při provádění operací násobení a dělení na čítači C.
- Přičtení jedničky k čítači x je simulováno jako vynásobení čítače C hodnotou 2, přičtení jedničky k čítači y jako vynásobení hodnotou 3, atd.
- Analogicky je odečtení jedničky od čítače x simulováno pomocí vydělení čítače C hodnotou 2, odečtení jedničky od čítače y vydělením hodnotou 3, atd.
- Test podmínky $x = 0$ odpovídá testu, zda je hodnota C dělitelná dvěmi, atd.

Minského stroj

Vidíme, že činnost libovolného Turingova stroje je možné simulovat Minského strojem s dvěma čítači.

Tato simulace je však mimořádně neefektivní:

- Už simulace pásky Turingova stroje pomocí tří čítačů vyžaduje exponenciálně větší počet kroků, než kolik by jich vykonal tento Turingův stroj.
- Simulace činnosti těchto tří čítačů pomocí dvou čítačů tento počet kroků dále exponenciálně zvyšuje.

Stroje RAM

Stroj RAM

Stroj RAM (Random Access Machine) je idealizovaný model počítače.

Skládá se z těchto částí:

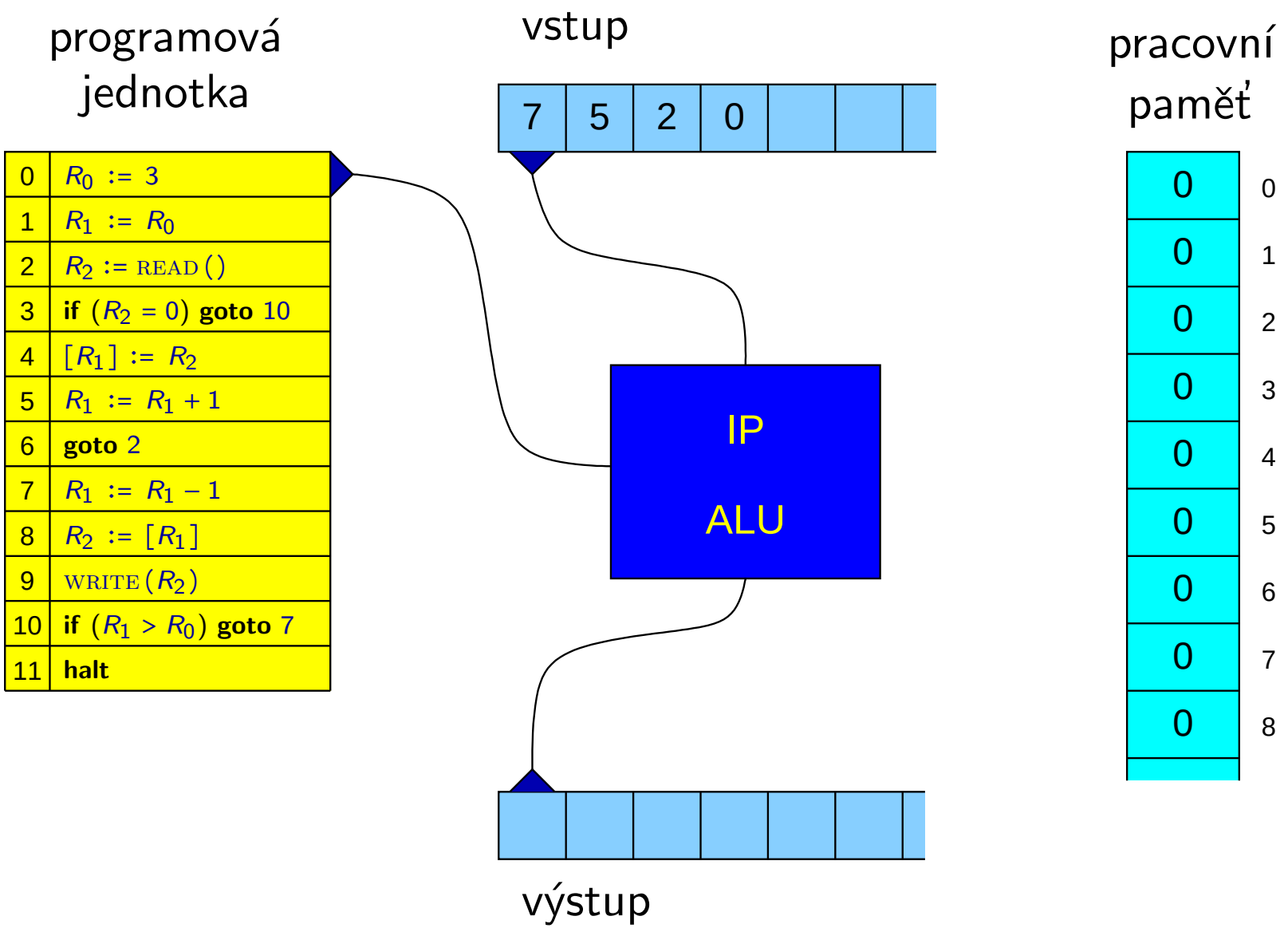
- **Programová jednotka** – obsahuje program stroje RAM a ukazatel na právě prováděnou instrukci
- **Pracovní paměť** tvořená buňkami očíslovanými $0, 1, 2, \ldots$

  Tyto buňky buňky budeme označovat $R_0, R_1, R_2, \ldots$

  Obsah buněk je možno číst i do nich zapisovat.
- **Vstupní páska** – je z ní možné pouze číst
- **Výstupní páska** – je na ni možno pouze zapisovat

Buňky paměti i vstupní a výstupní pásky obsahují jako hodnoty celá čísla (tj. prvky množiny $\mathbb{Z}$).

Stroj RAM


programová jednotka
vstup
pracovní paměť
0 $R_0 := 3$
1 $R_1 := R_0$
2 $R_2 := \text{READ}()$
3 **if** $(R_2 = 0)$ **goto** 10
4 $[R_1] := R_2$
5 $R_1 := R_1 + 1$
6 **goto** 2
7 $R_1 := R_1 - 1$
8 $R_2 := [R_1]$
9 $\text{WRITE}(R_2)$
10 **if** $(R_1 > R_0)$ **goto** 7
11 **halt**
7 5 2 0
IP
ALU
0 0 0 0 0 0 0 0
0 1 2 3 4 5 6 7 8
výstup

Stroj RAM

Přehled instrukcí:

$R_i := c$	– přiřazení konstanty
$R_i := R_j$	– přiřazení
$R_i := [R_j]$	– load (čtení z paměti)
$[R_i] := R_j$	– store (zápis do paměti)
$R_i := R_j \ op \ R_k$	– aritmetické instrukce, $op \in \{+, -, *, /\}$
nebo $R_i := R_j \ op \ c$	
if $(R_i \ rel \ R_j)$ **goto** ℓ	– podmíněný skok, $rel \in \{=, \neq, \leq, \geq, <, >\}$
nebo **if** $(R_i \ rel \ c)$ **goto** ℓ	
goto ℓ	– nepodmíněný skok
$R_i :=$ READ $()$	– čtení ze vstupu
WRITE (R_i)	– zápis na výstup
halt	– zastavení programu

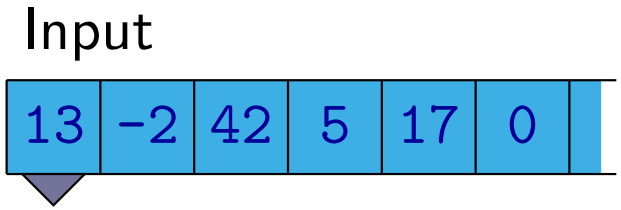
Stroj RAM

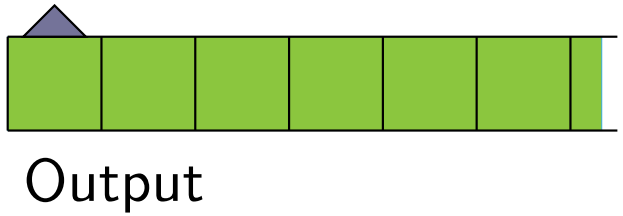
Příklady instrukcí:

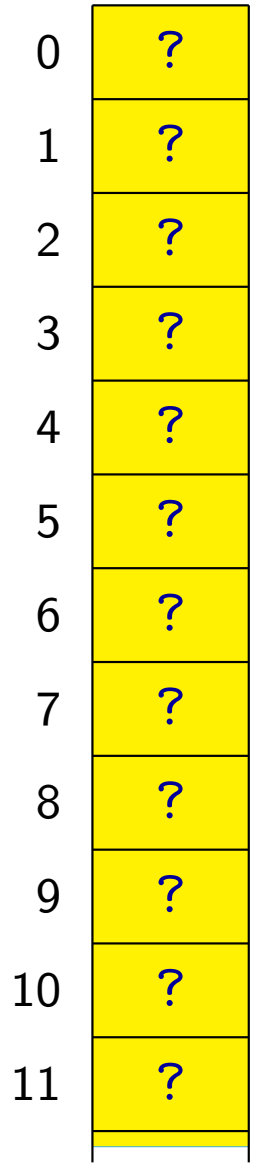
$R_5 := 42$	– přiřazení konstanty
$R_{12} := R_3$	– přiřazení
$R_8 := [R_2]$	– load (čtení z paměti)
$[R_{15}] := R_9$	– store (zápis do paměti)
$R_7 := R_3 + R_6$	– aritmetická instrukce
$R_{18} := R_{18} - 1$	– aritmetická instrukce
if $(R_4 \geq R_1)$ **goto** 2801	– podmíněný skok
if $(R_2 \neq 0)$ **goto** 3581	– podmíněný skok
goto 537	– nepodmíněný skok
$R_{23} :=$ READ $()$	– čtení ze vstupu
WRITE (R_{17})	– zápis na výstup
halt	– zastavení programu

Stroj RAM

$R_0 := 3$
$R_1 := R_0$
$L_1 : R_2 := \text{READ}()$
if $(R_2 = 0)$ **goto** L_3
$[R_1] := R_2$
$R_1 := R_1 + 1$
goto L_1
$L_2 : R_1 := R_1 - 1$
$R_2 := [R_1]$
$\text{WRITE}(R_2)$
$L_3 :$ **if** $(R_1 > R_0)$ **goto** L_2
halt

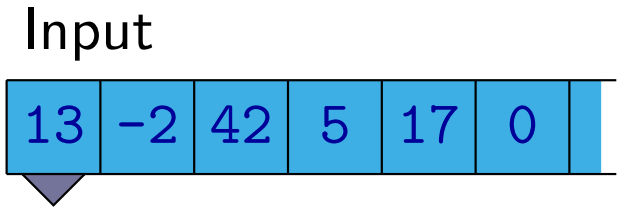


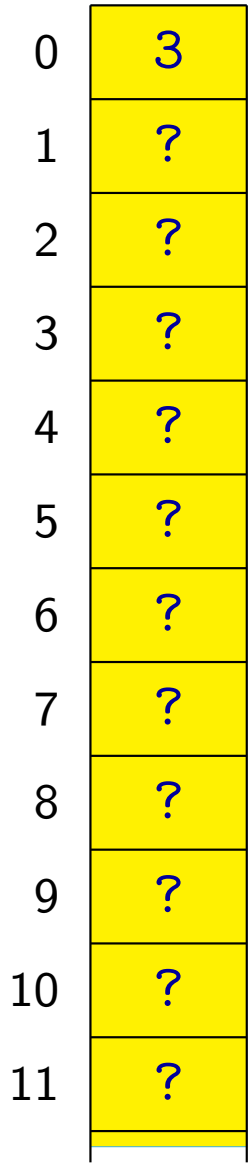


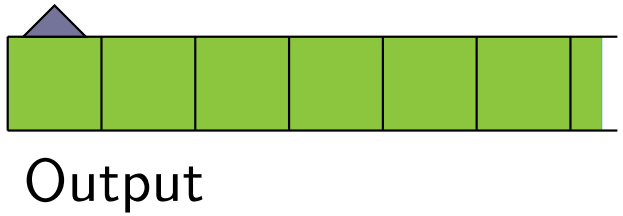


Stroj RAM

$R_0 := 3$
$R_1 := R_0$
$L_1 : R_2 := \text{READ}()$
if $(R_2 = 0)$ **goto** L_3
$[R_1] := R_2$
$R_1 := R_1 + 1$
goto L_1
$L_2 : R_1 := R_1 - 1$
$R_2 := [R_1]$
$\text{WRITE}(R_2)$
$L_3 :$ **if** $(R_1 > R_0)$ **goto** L_2
halt

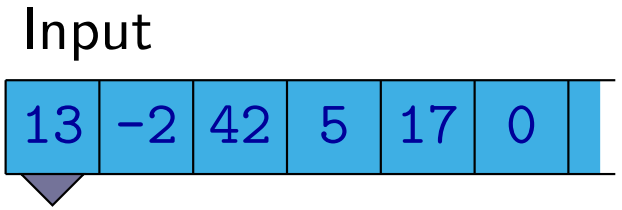


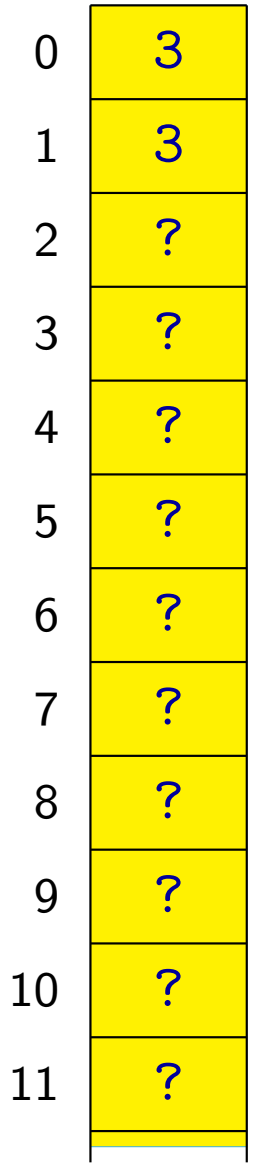


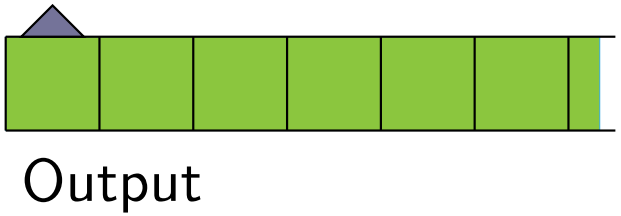


Stroj RAM

$R_0 := 3$
$R_1 := R_0$
$L_1 : R_2 := \text{READ}()$
if $(R_2 = 0)$ **goto** L_3
$[R_1] := R_2$
$R_1 := R_1 + 1$
goto L_1
$L_2 : R_1 := R_1 - 1$
$R_2 := [R_1]$
$\text{WRITE}(R_2)$
$L_3 :$ **if** $(R_1 > R_0)$ **goto** L_2
halt

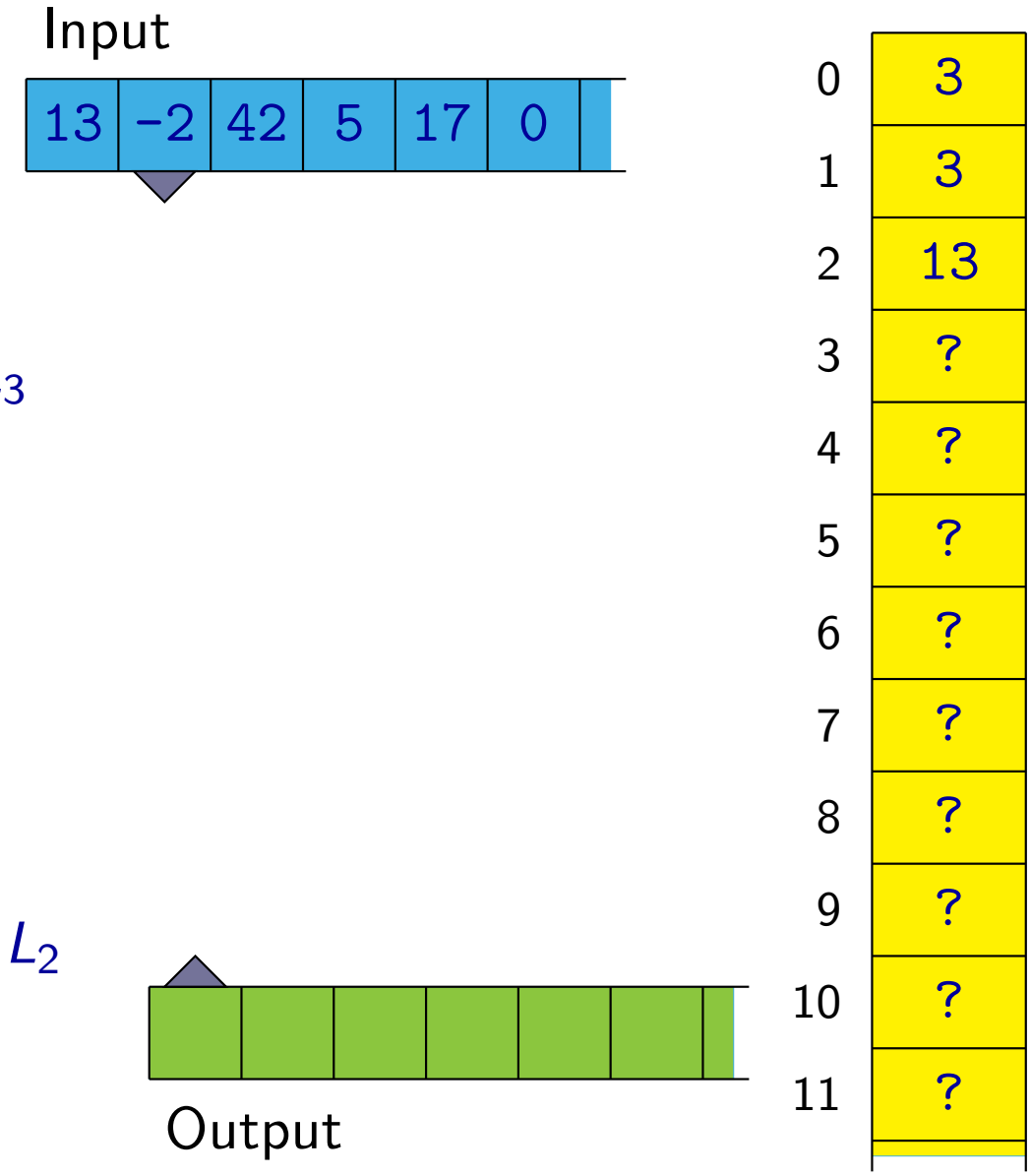






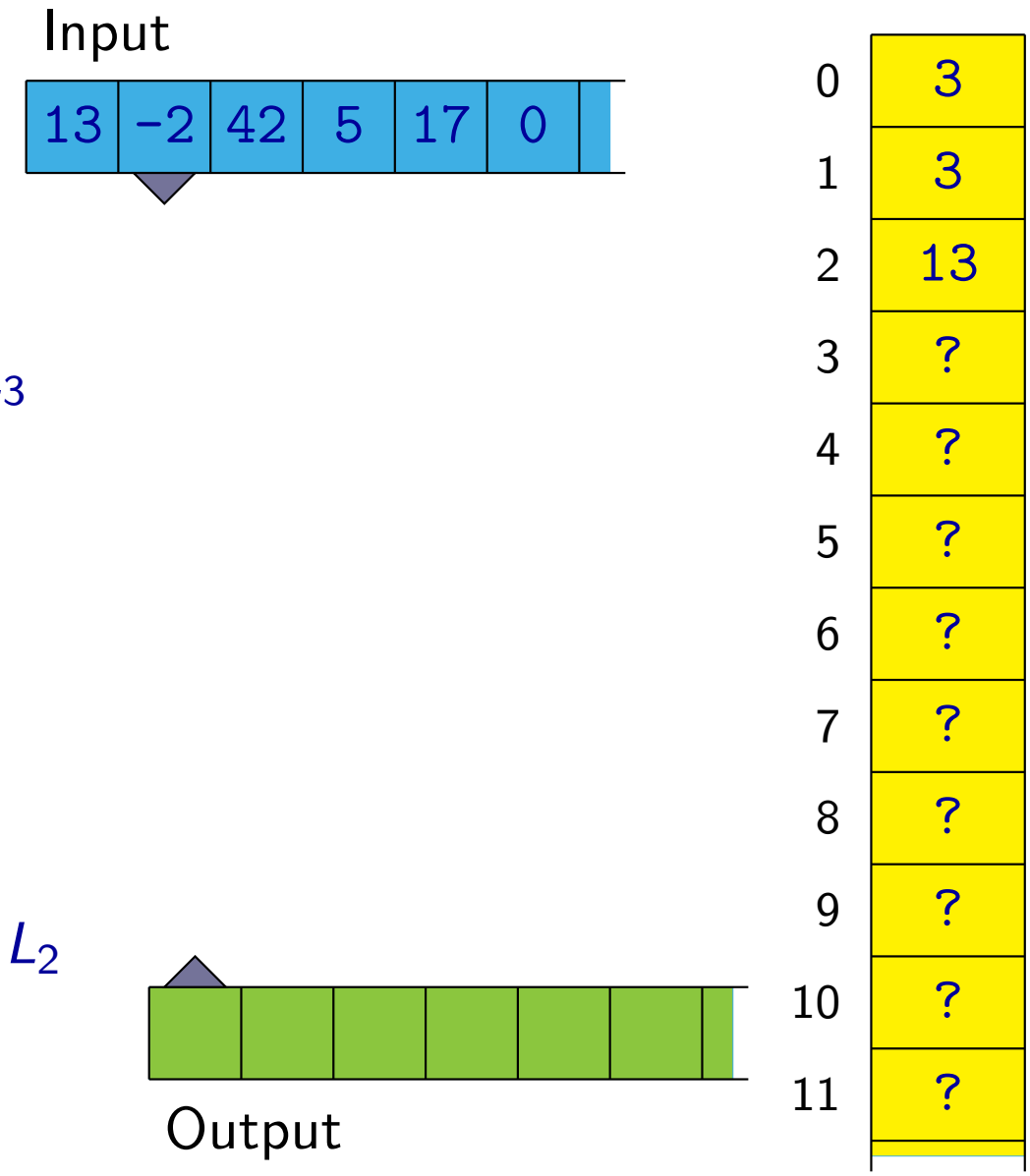
Stroj RAM

$R_0 := 3$
$R_1 := R_0$
$L_1 : R_2 := \text{READ}()$
if $(R_2 = 0)$ **goto** L_3
$[R_1] := R_2$
$R_1 := R_1 + 1$
goto L_1
$L_2 : R_1 := R_1 - 1$
$R_2 := [R_1]$
$\text{WRITE}(R_2)$
$L_3 :$ **if** $(R_1 > R_0)$ **goto** L_2
halt



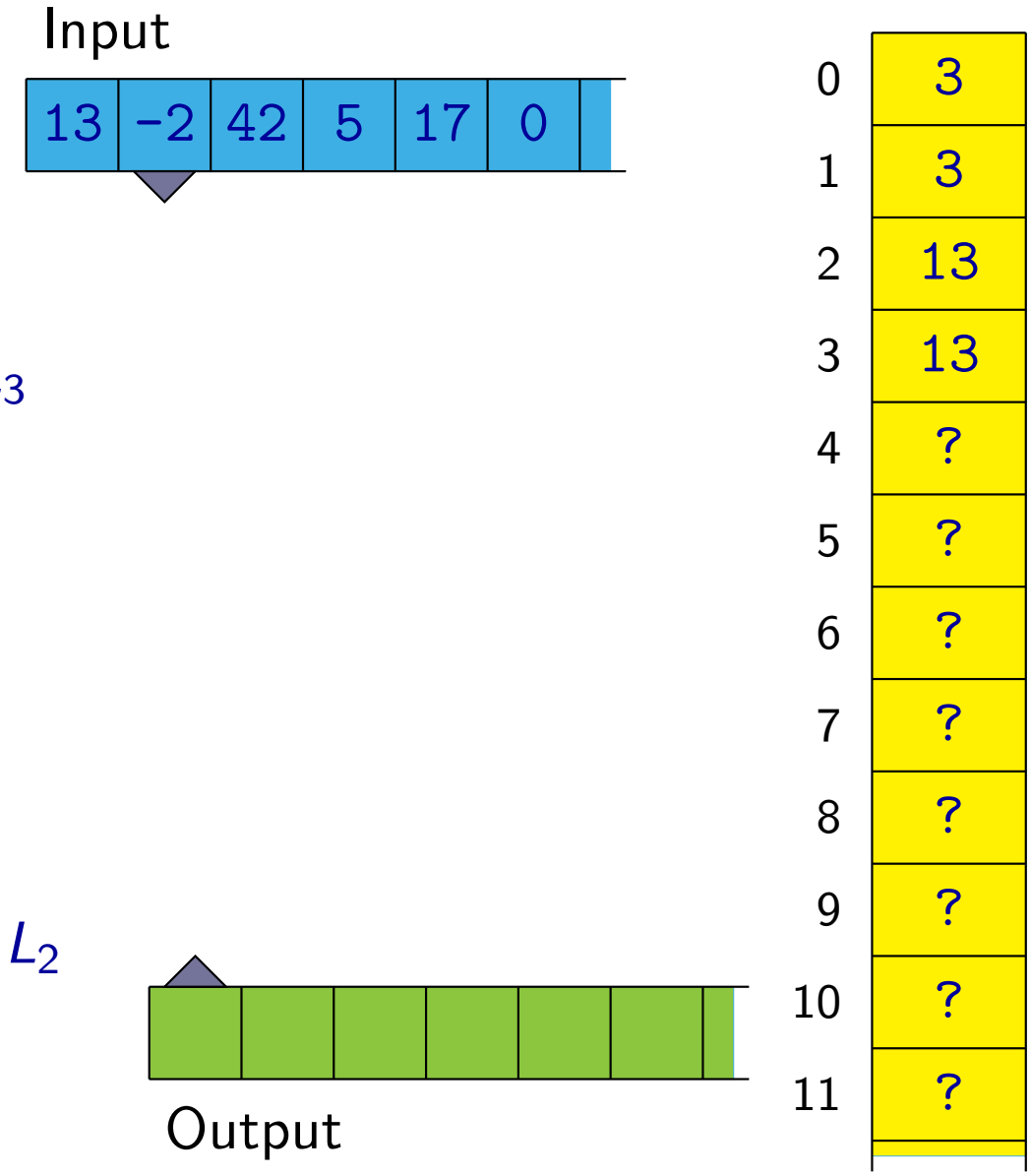
Stroj RAM

$R_0 := 3$
$R_1 := R_0$
$L_1 : R_2 := \text{READ}()$
if $(R_2 = 0)$ **goto** L_3
$[R_1] := R_2$
$R_1 := R_1 + 1$
goto L_1
$L_2 : R_1 := R_1 - 1$
$R_2 := [R_1]$
$\text{WRITE}(R_2)$
$L_3 :$ **if** $(R_1 > R_0)$ **goto** L_2
halt



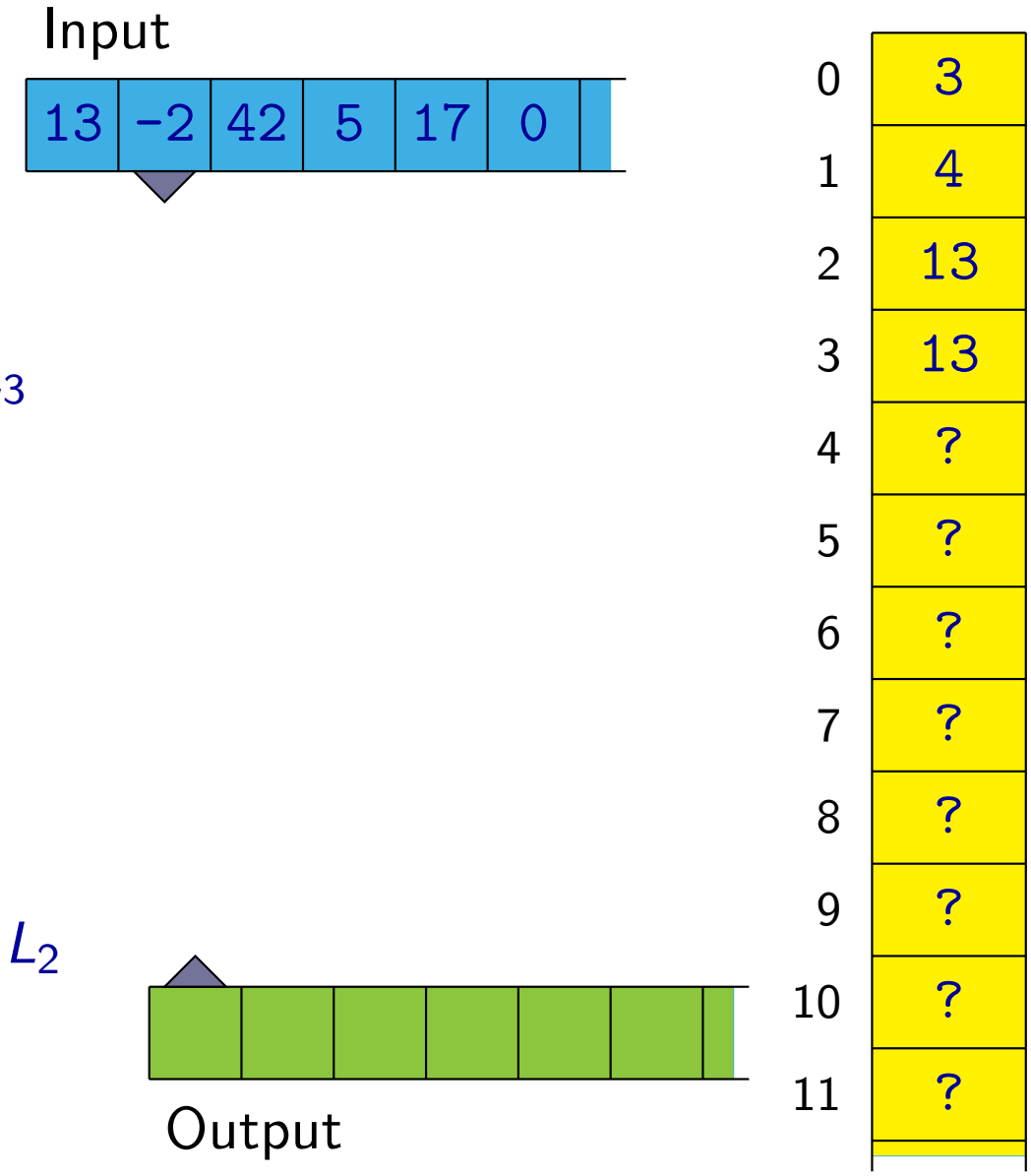
Stroj RAM

$R_0 := 3$
$R_1 := R_0$
$L_1 : R_2 := \text{READ}()$
if $(R_2 = 0)$ **goto** L_3
$[R_1] := R_2$
$R_1 := R_1 + 1$
goto L_1
$L_2 : R_1 := R_1 - 1$
$R_2 := [R_1]$
$\text{WRITE}(R_2)$
$L_3 :$ **if** $(R_1 > R_0)$ **goto** L_2
halt



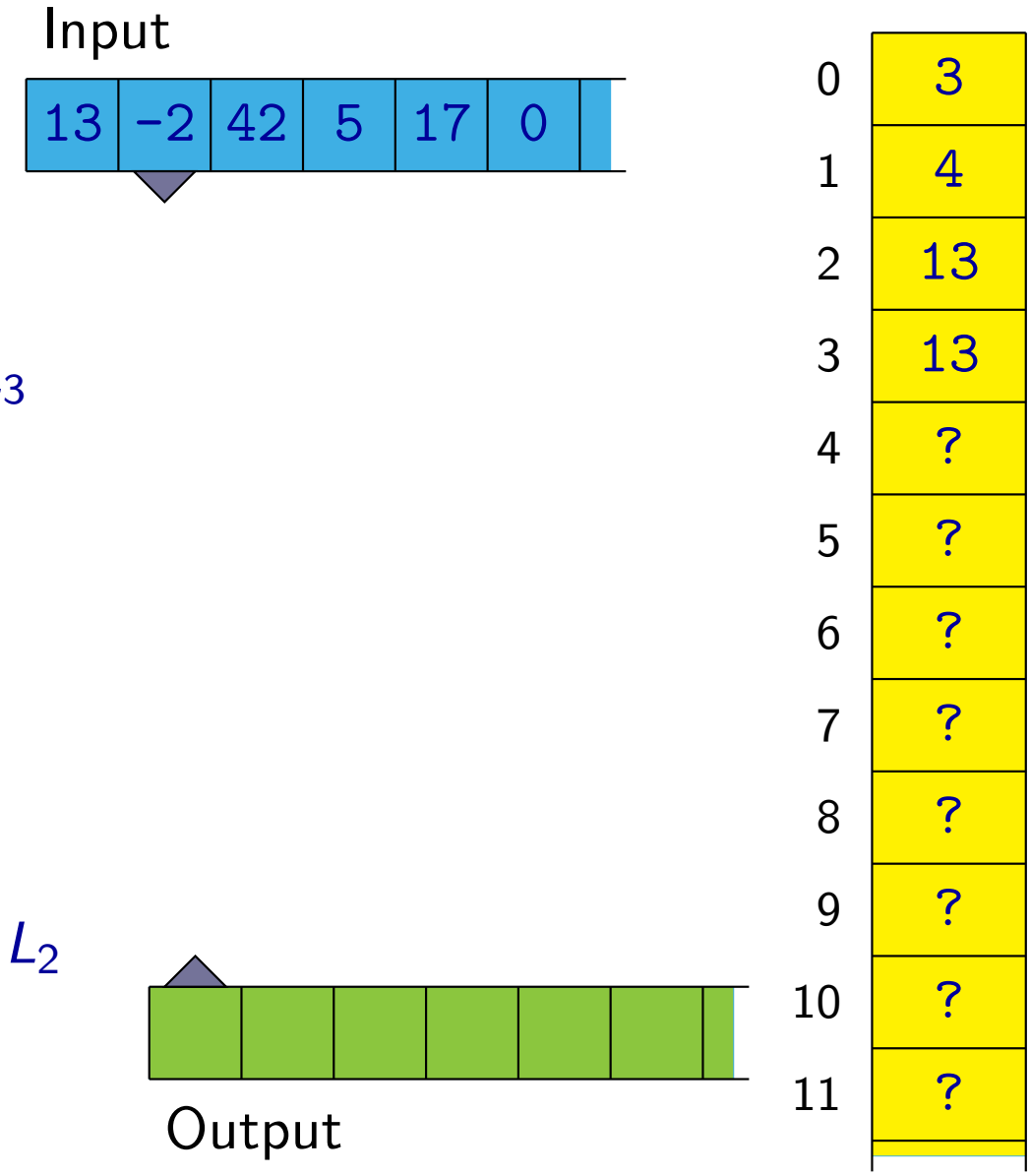
Stroj RAM

$R_0 := 3$
$R_1 := R_0$
$L_1 : R_2 := \text{READ}()$
if $(R_2 = 0)$ **goto** L_3
$[R_1] := R_2$
$R_1 := R_1 + 1$
goto L_1
$L_2 : R_1 := R_1 - 1$
$R_2 := [R_1]$
$\text{WRITE}(R_2)$
$L_3 :$ **if** $(R_1 > R_0)$ **goto** L_2
halt



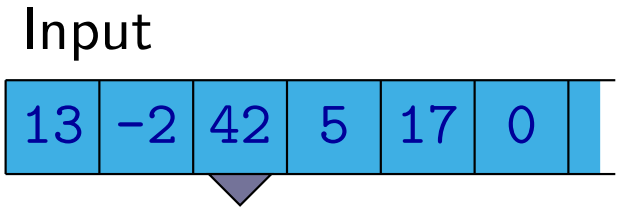
Stroj RAM

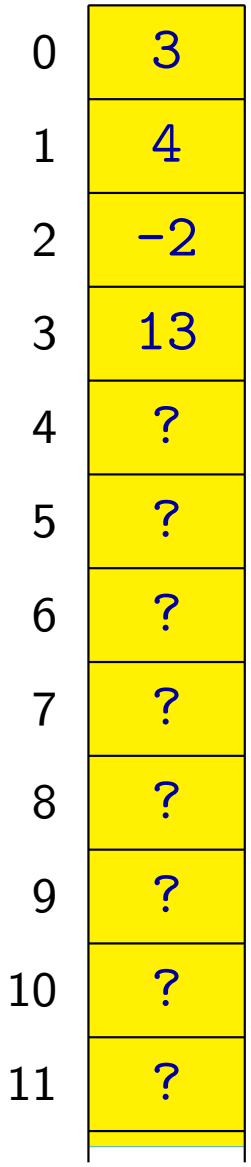
$R_0 := 3$
$R_1 := R_0$
$L_1 : R_2 := \text{READ}()$
if $(R_2 = 0)$ **goto** L_3
$[R_1] := R_2$
$R_1 := R_1 + 1$
goto L_1
$L_2 : R_1 := R_1 - 1$
$R_2 := [R_1]$
$\text{WRITE}(R_2)$
$L_3 :$ **if** $(R_1 > R_0)$ **goto** L_2
halt

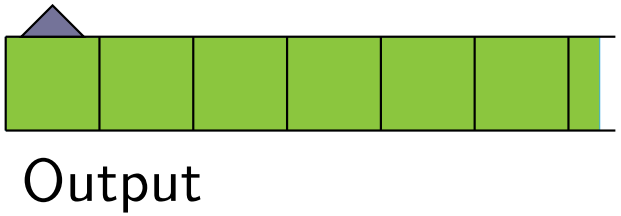


Stroj RAM

$R_0 := 3$
$R_1 := R_0$
$L_1 : R_2 := \text{READ}()$
if $(R_2 = 0)$ **goto** L_3
$[R_1] := R_2$
$R_1 := R_1 + 1$
goto L_1
$L_2 : R_1 := R_1 - 1$
$R_2 := [R_1]$
$\text{WRITE}(R_2)$
$L_3 :$ **if** $(R_1 > R_0)$ **goto** L_2
halt

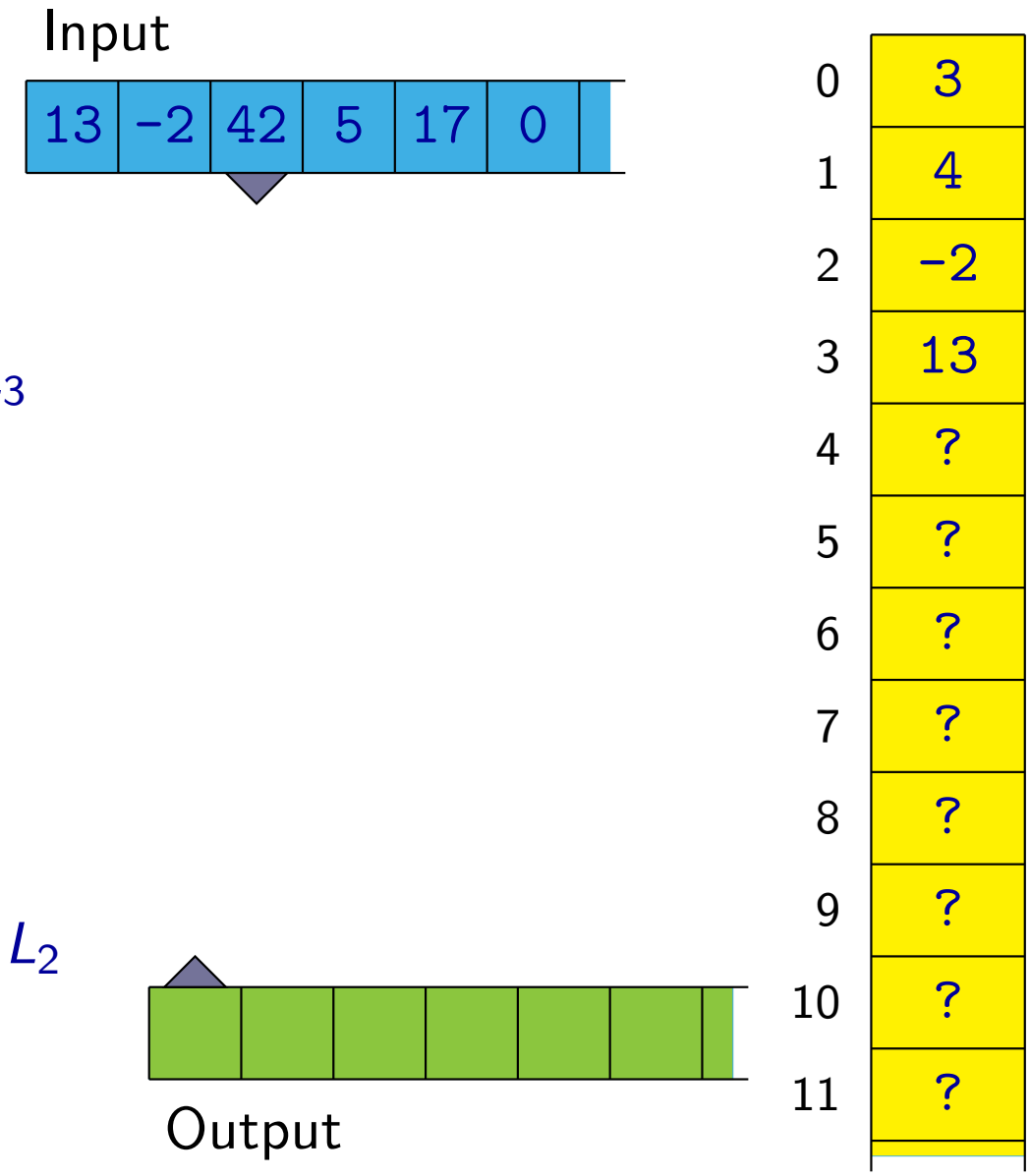






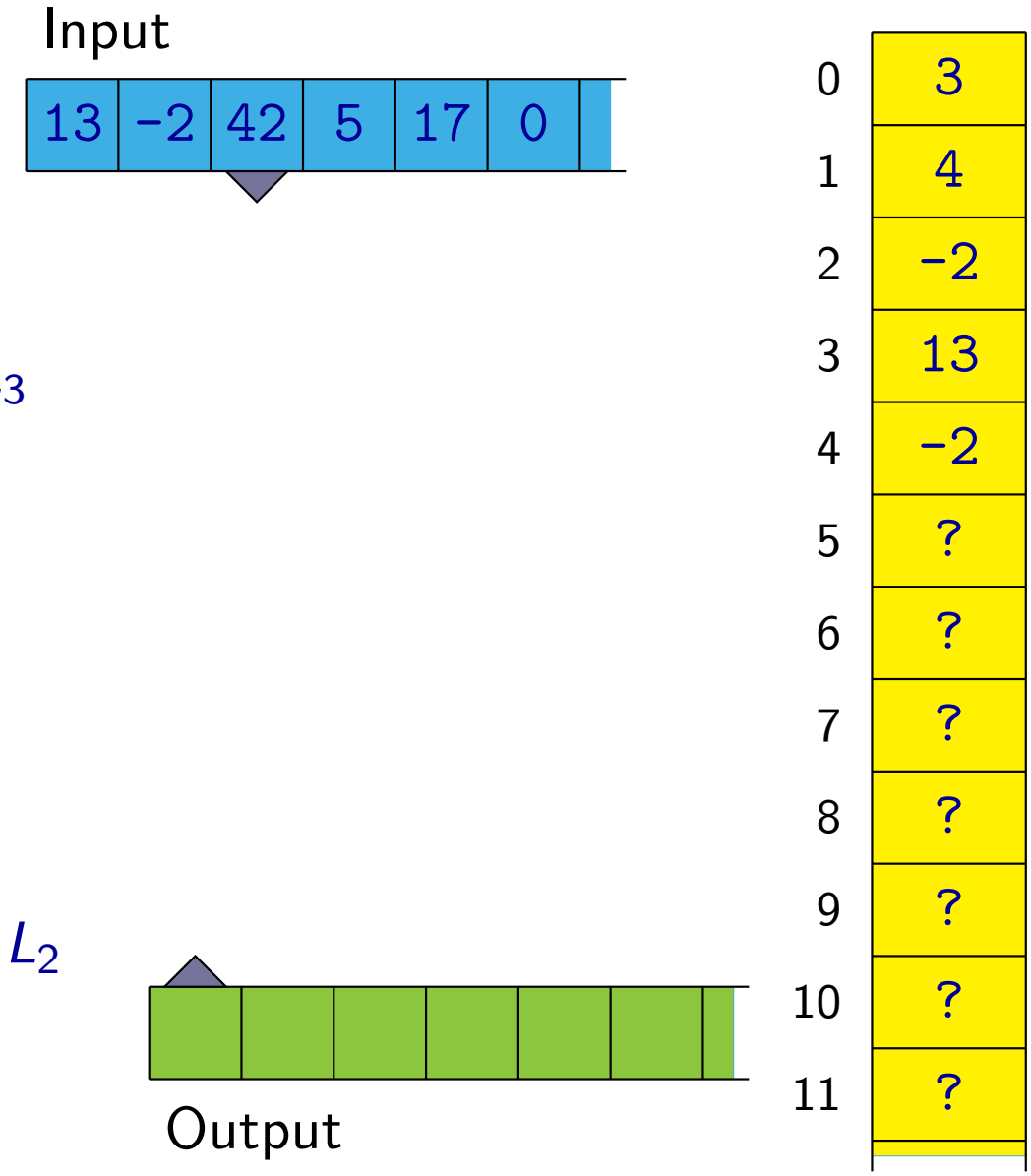
Stroj RAM

$R_0 := 3$
$R_1 := R_0$
$L_1 : R_2 := \text{READ}()$
if $(R_2 = 0)$ **goto** L_3
$[R_1] := R_2$
$R_1 := R_1 + 1$
goto L_1
$L_2 : R_1 := R_1 - 1$
$R_2 := [R_1]$
$\text{WRITE}(R_2)$
$L_3 :$ **if** $(R_1 > R_0)$ **goto** L_2
halt



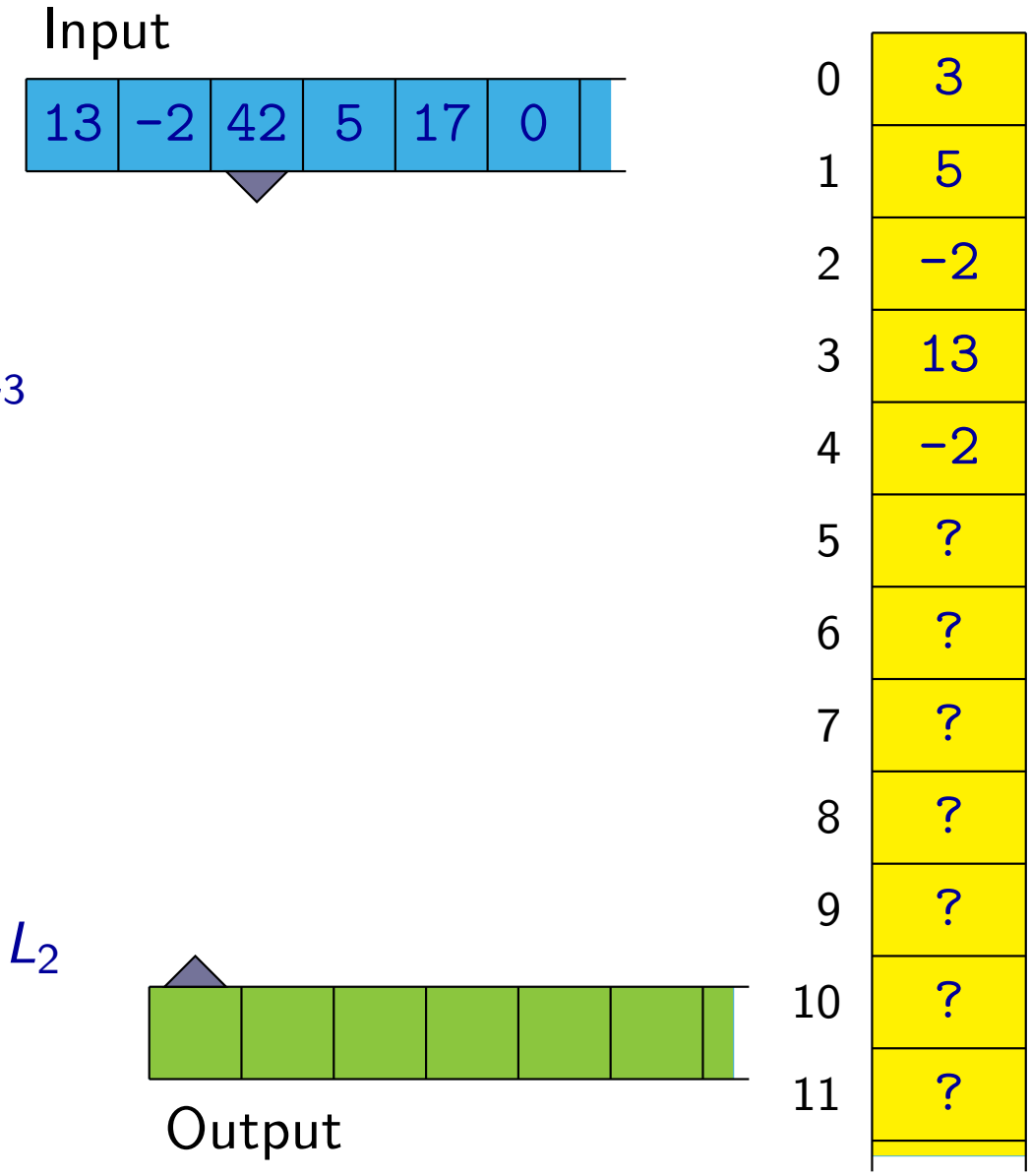
Stroj RAM

$R_0 := 3$
$R_1 := R_0$
$L_1 : R_2 := \text{READ}()$
if $(R_2 = 0)$ **goto** L_3
$[R_1] := R_2$
$R_1 := R_1 + 1$
goto L_1
$L_2 : R_1 := R_1 - 1$
$R_2 := [R_1]$
$\text{WRITE}(R_2)$
$L_3 :$ **if** $(R_1 > R_0)$ **goto** L_2
halt



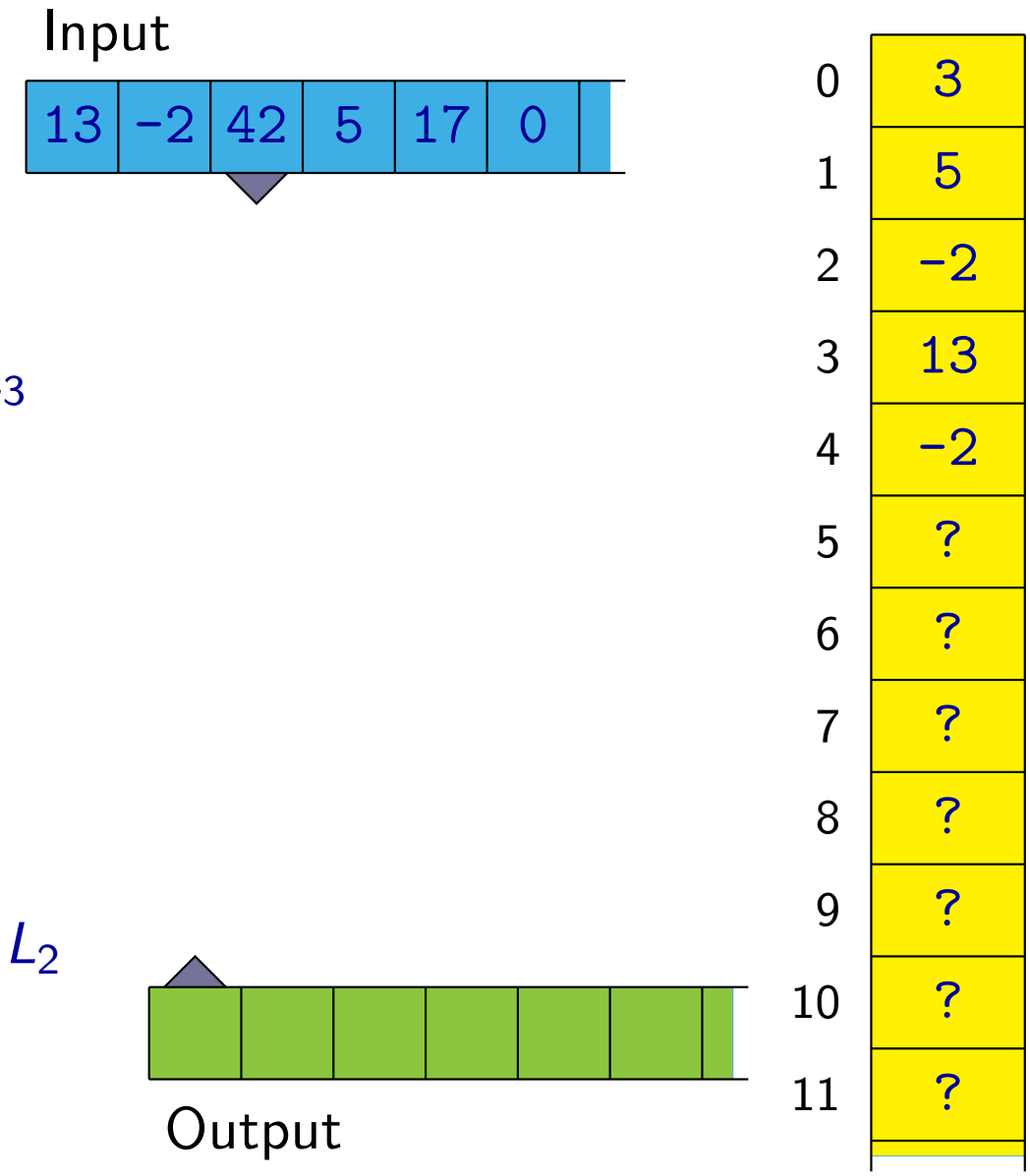
Stroj RAM

$R_0 := 3$
$R_1 := R_0$
$L_1 : R_2 := \text{READ}()$
if $(R_2 = 0)$ **goto** L_3
$[R_1] := R_2$
$R_1 := R_1 + 1$
goto L_1
$L_2 : R_1 := R_1 - 1$
$R_2 := [R_1]$
$\text{WRITE}(R_2)$
$L_3 :$ **if** $(R_1 > R_0)$ **goto** L_2
halt



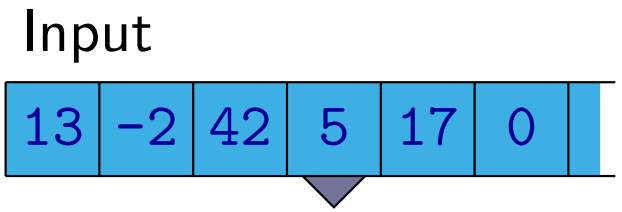
Stroj RAM

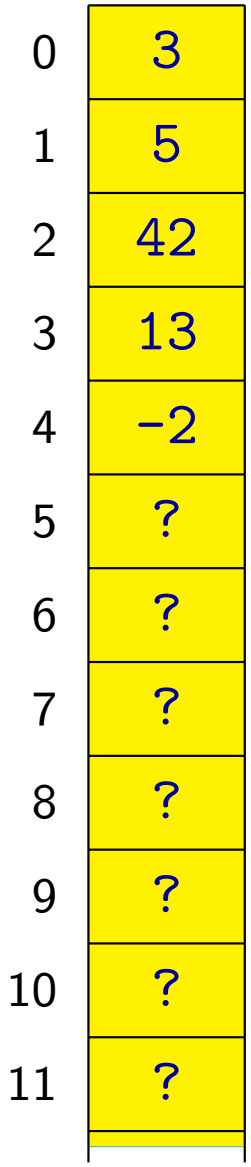
$R_0 := 3$
$R_1 := R_0$
$L_1 : R_2 := \text{READ}()$
if $(R_2 = 0)$ **goto** L_3
$[R_1] := R_2$
$R_1 := R_1 + 1$
goto L_1
$L_2 : R_1 := R_1 - 1$
$R_2 := [R_1]$
$\text{WRITE}(R_2)$
$L_3 :$ **if** $(R_1 > R_0)$ **goto** L_2
halt

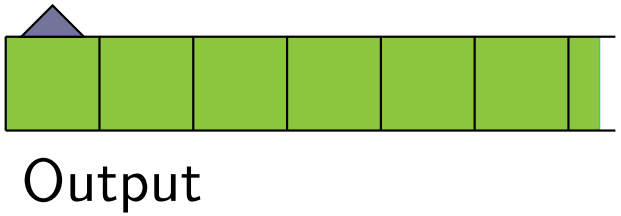


Stroj RAM

$R_0 := 3$
$R_1 := R_0$
$L_1 : R_2 := \text{READ}()$
if $(R_2 = 0)$ **goto** L_3
$[R_1] := R_2$
$R_1 := R_1 + 1$
goto L_1
$L_2 : R_1 := R_1 - 1$
$R_2 := [R_1]$
$\text{WRITE}(R_2)$
$L_3 :$ **if** $(R_1 > R_0)$ **goto** L_2
halt

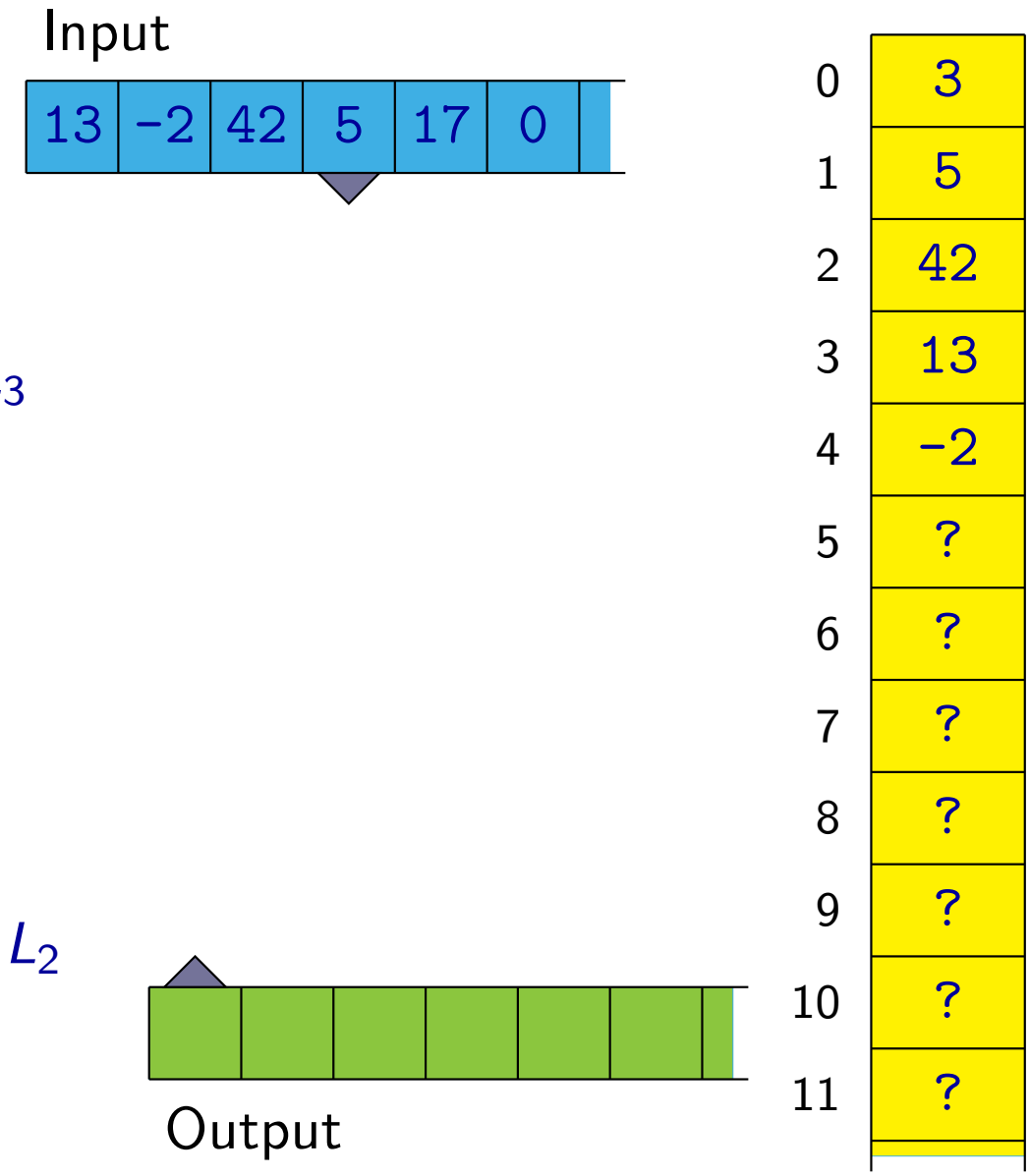






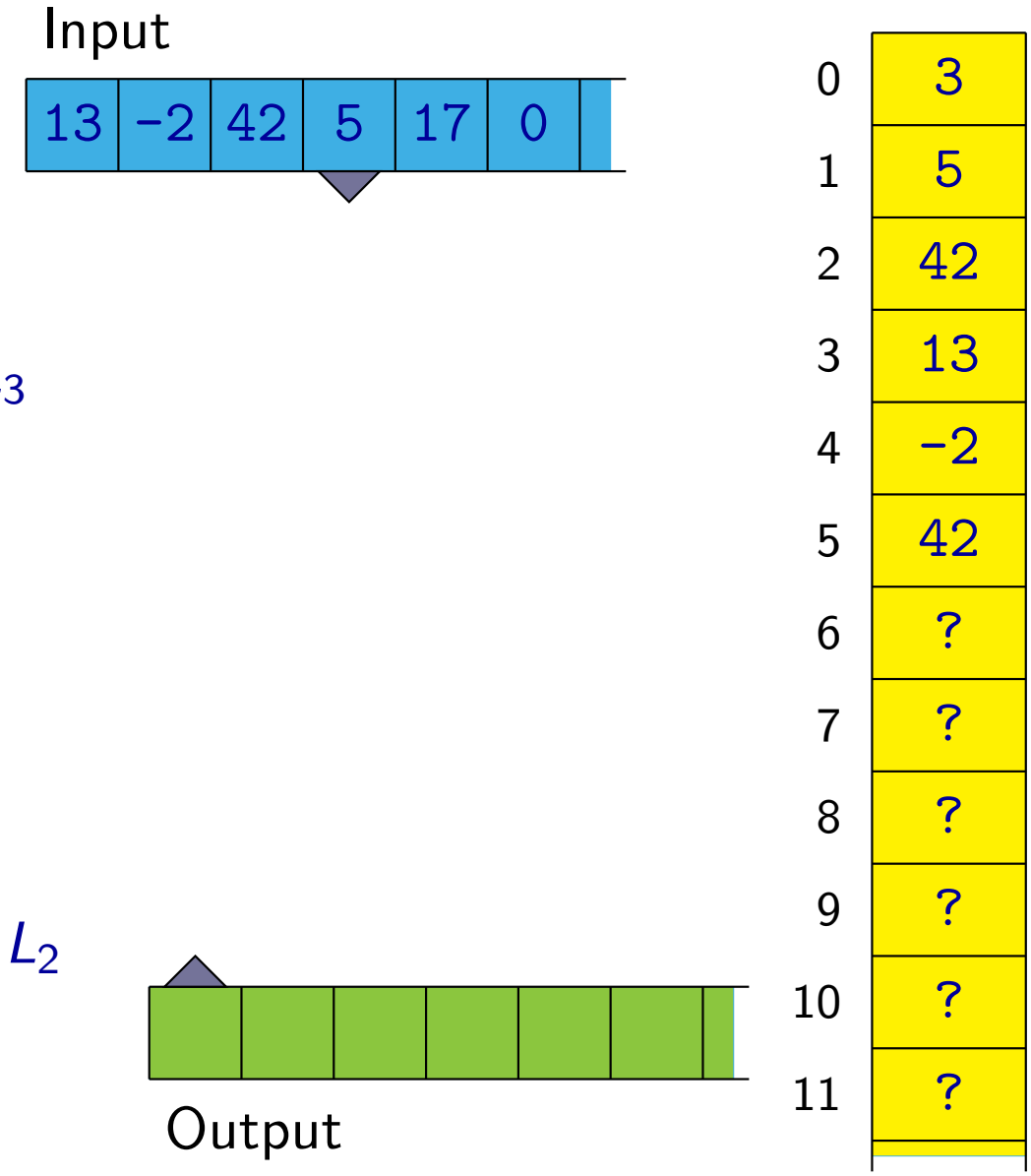
Stroj RAM

$R_0 := 3$
$R_1 := R_0$
$L_1 : R_2 := \text{READ}()$
if $(R_2 = 0)$ **goto** L_3
$[R_1] := R_2$
$R_1 := R_1 + 1$
goto L_1
$L_2 : R_1 := R_1 - 1$
$R_2 := [R_1]$
$\text{WRITE}(R_2)$
$L_3 :$ **if** $(R_1 > R_0)$ **goto** L_2
halt



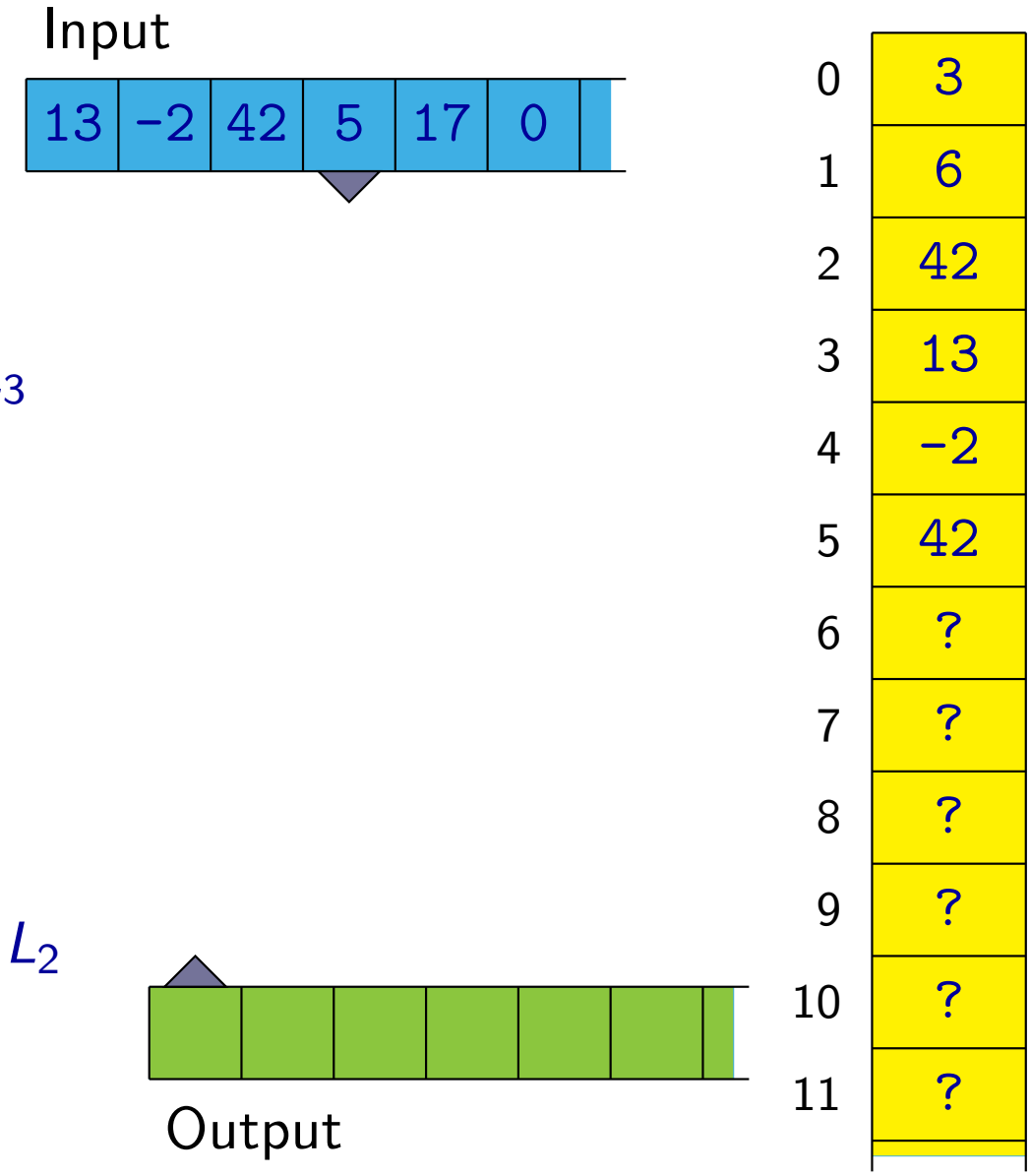
Stroj RAM

$R_0 := 3$
$R_1 := R_0$
$L_1 : R_2 := \text{READ}()$
if $(R_2 = 0)$ **goto** L_3
$[R_1] := R_2$
$R_1 := R_1 + 1$
goto L_1
$L_2 : R_1 := R_1 - 1$
$R_2 := [R_1]$
$\text{WRITE}(R_2)$
$L_3 :$ **if** $(R_1 > R_0)$ **goto** L_2
halt



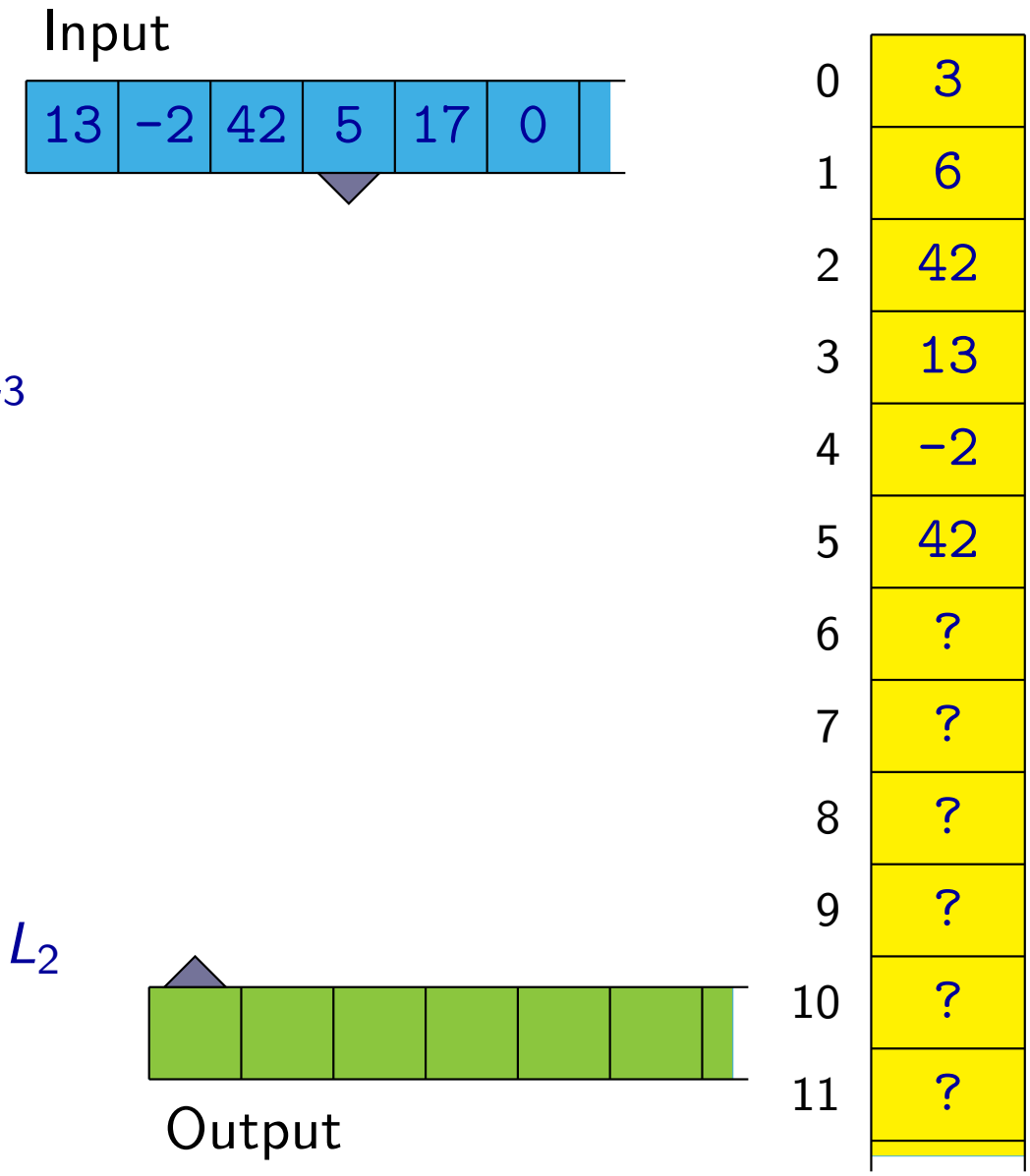
Stroj RAM

$R_0 := 3$
$R_1 := R_0$
$L_1 : R_2 := \text{READ}()$
if $(R_2 = 0)$ **goto** L_3
$[R_1] := R_2$
$R_1 := R_1 + 1$
goto L_1
$L_2 : R_1 := R_1 - 1$
$R_2 := [R_1]$
$\text{WRITE}(R_2)$
$L_3 :$ **if** $(R_1 > R_0)$ **goto** L_2
halt



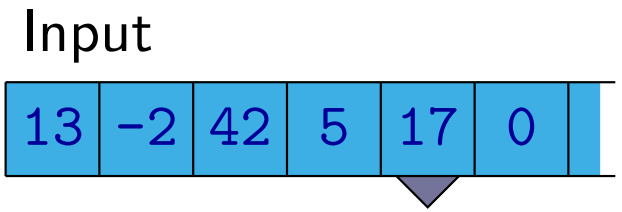
Stroj RAM

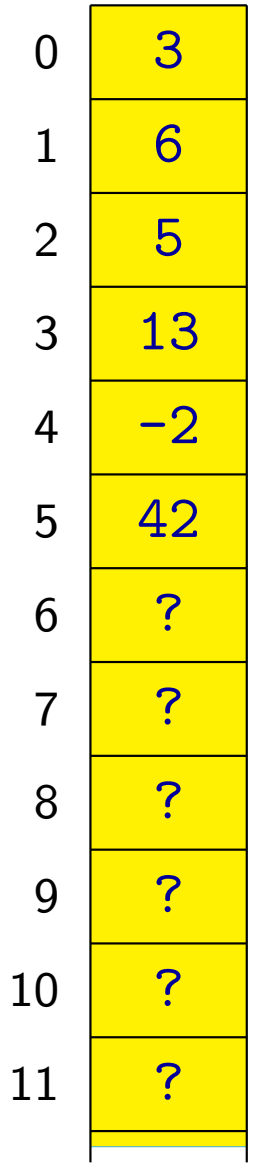
$R_0 := 3$
$R_1 := R_0$
$L_1 : R_2 := \text{READ}()$
if $(R_2 = 0)$ **goto** L_3
$[R_1] := R_2$
$R_1 := R_1 + 1$
goto L_1
$L_2 : R_1 := R_1 - 1$
$R_2 := [R_1]$
$\text{WRITE}(R_2)$
$L_3 :$ **if** $(R_1 > R_0)$ **goto** L_2
halt

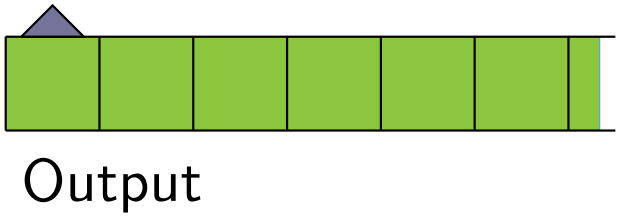


Stroj RAM

$R_0 := 3$
$R_1 := R_0$
$L_1 : R_2 := \text{READ}()$
if $(R_2 = 0)$ **goto** L_3
$[R_1] := R_2$
$R_1 := R_1 + 1$
goto L_1
$L_2 : R_1 := R_1 - 1$
$R_2 := [R_1]$
$\text{WRITE}(R_2)$
$L_3 :$ **if** $(R_1 > R_0)$ **goto** L_2
halt

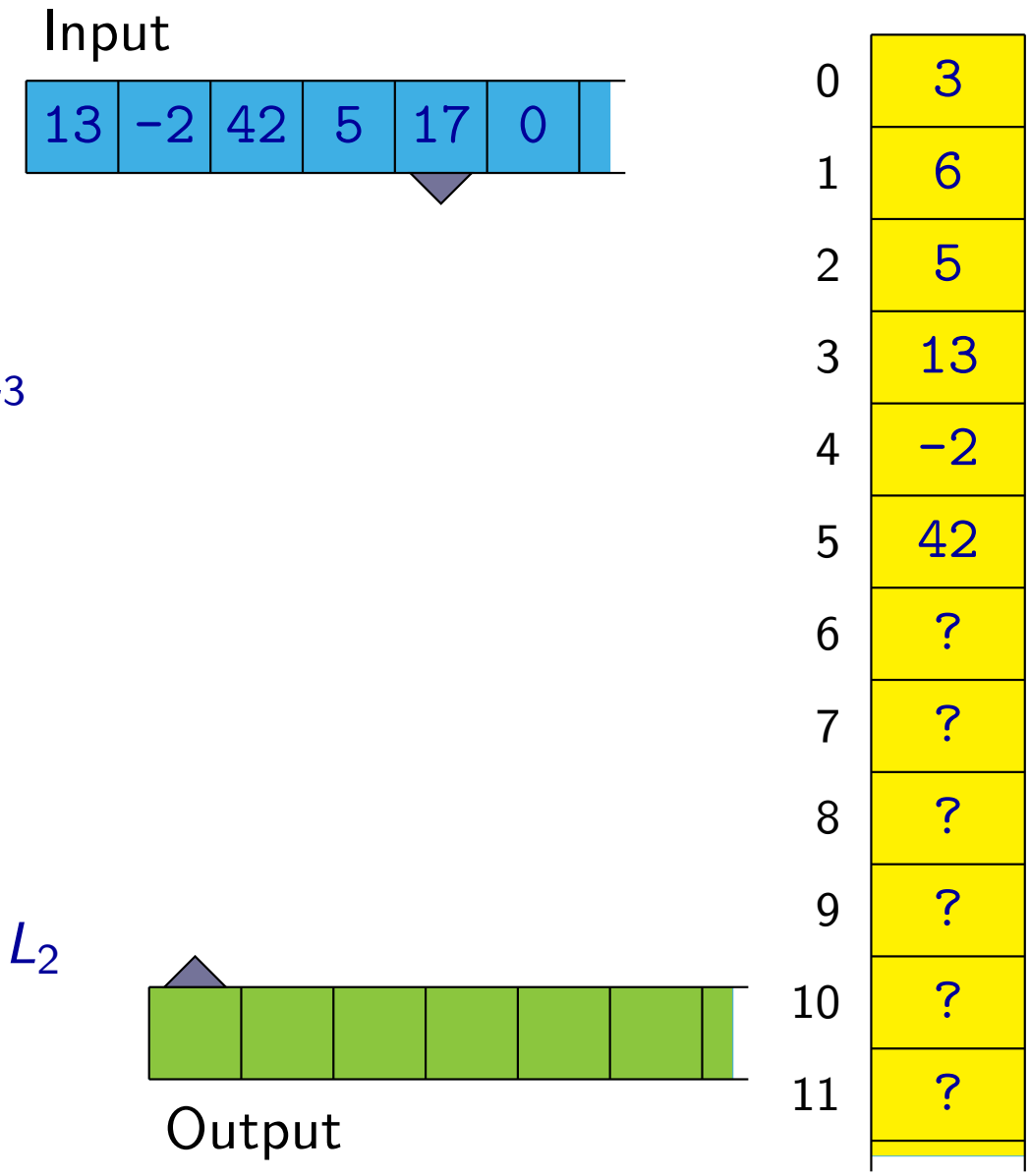






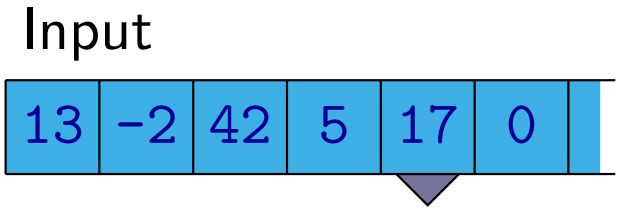
Stroj RAM

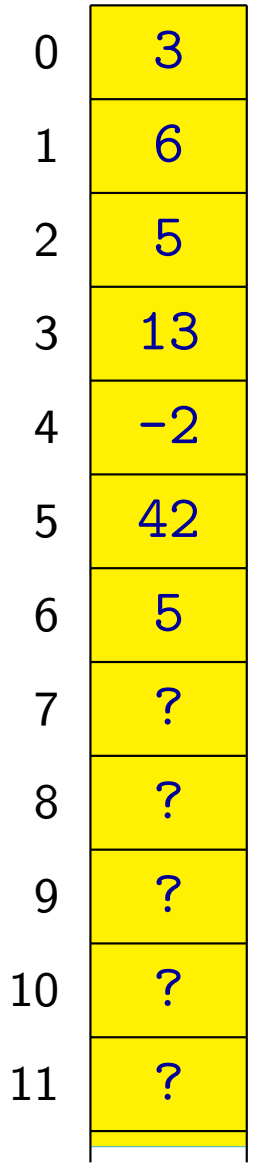
$R_0 := 3$
$R_1 := R_0$
$L_1 : R_2 := \text{READ}()$
if $(R_2 = 0)$ **goto** L_3
$[R_1] := R_2$
$R_1 := R_1 + 1$
goto L_1
$L_2 : R_1 := R_1 - 1$
$R_2 := [R_1]$
$\text{WRITE}(R_2)$
$L_3 :$ **if** $(R_1 > R_0)$ **goto** L_2
halt

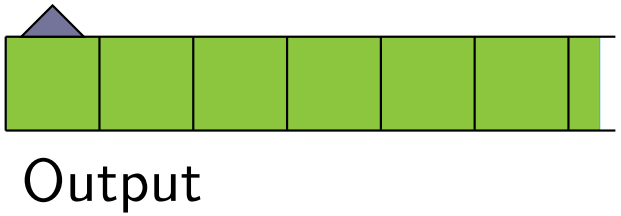


Stroj RAM

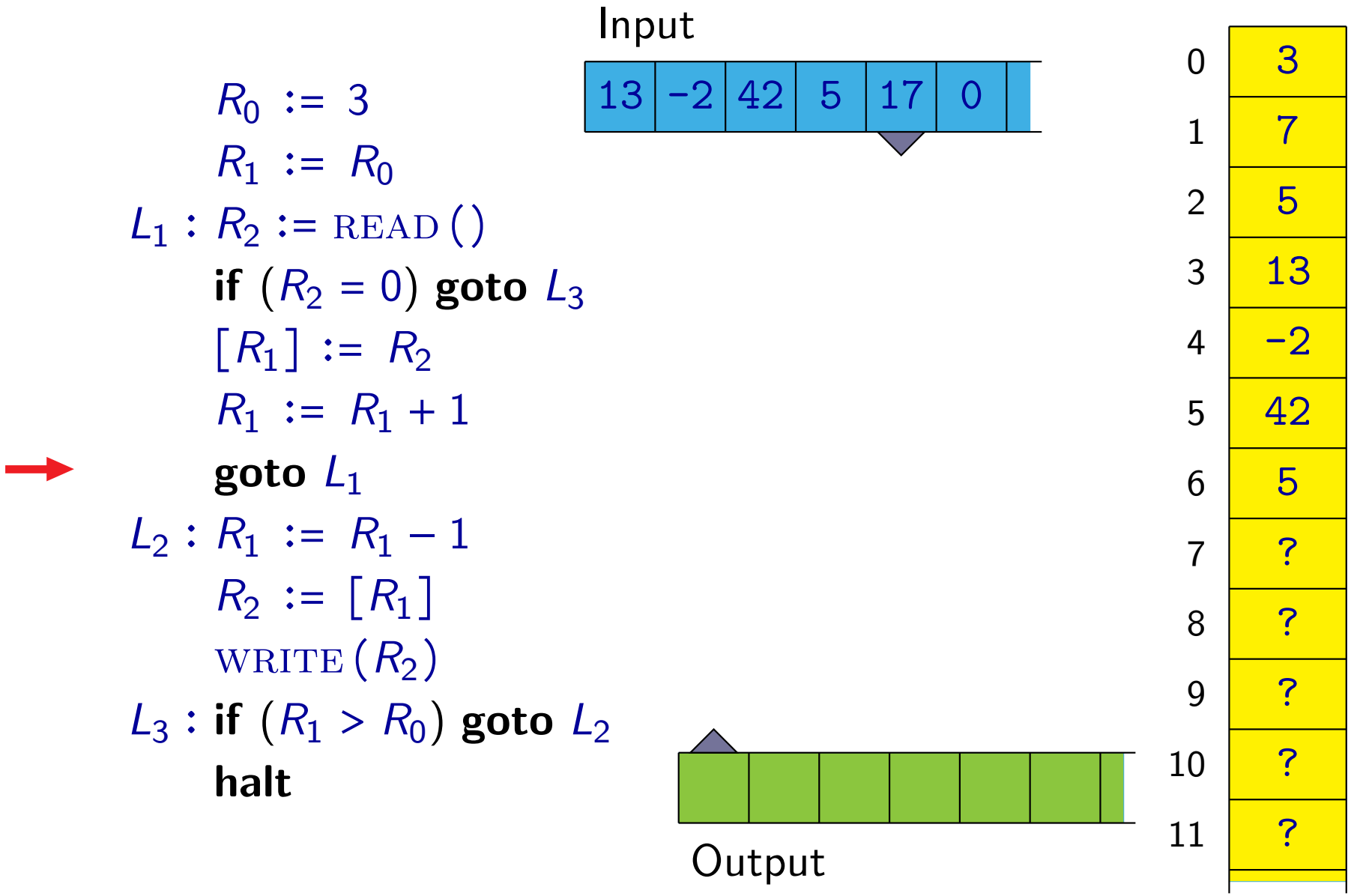
$R_0 := 3$
$R_1 := R_0$
$L_1 : R_2 := \text{READ}()$
if $(R_2 = 0)$ **goto** L_3
$[R_1] := R_2$
$R_1 := R_1 + 1$
goto L_1
$L_2 : R_1 := R_1 - 1$
$R_2 := [R_1]$
$\text{WRITE}(R_2)$
$L_3 :$ **if** $(R_1 > R_0)$ **goto** L_2
halt







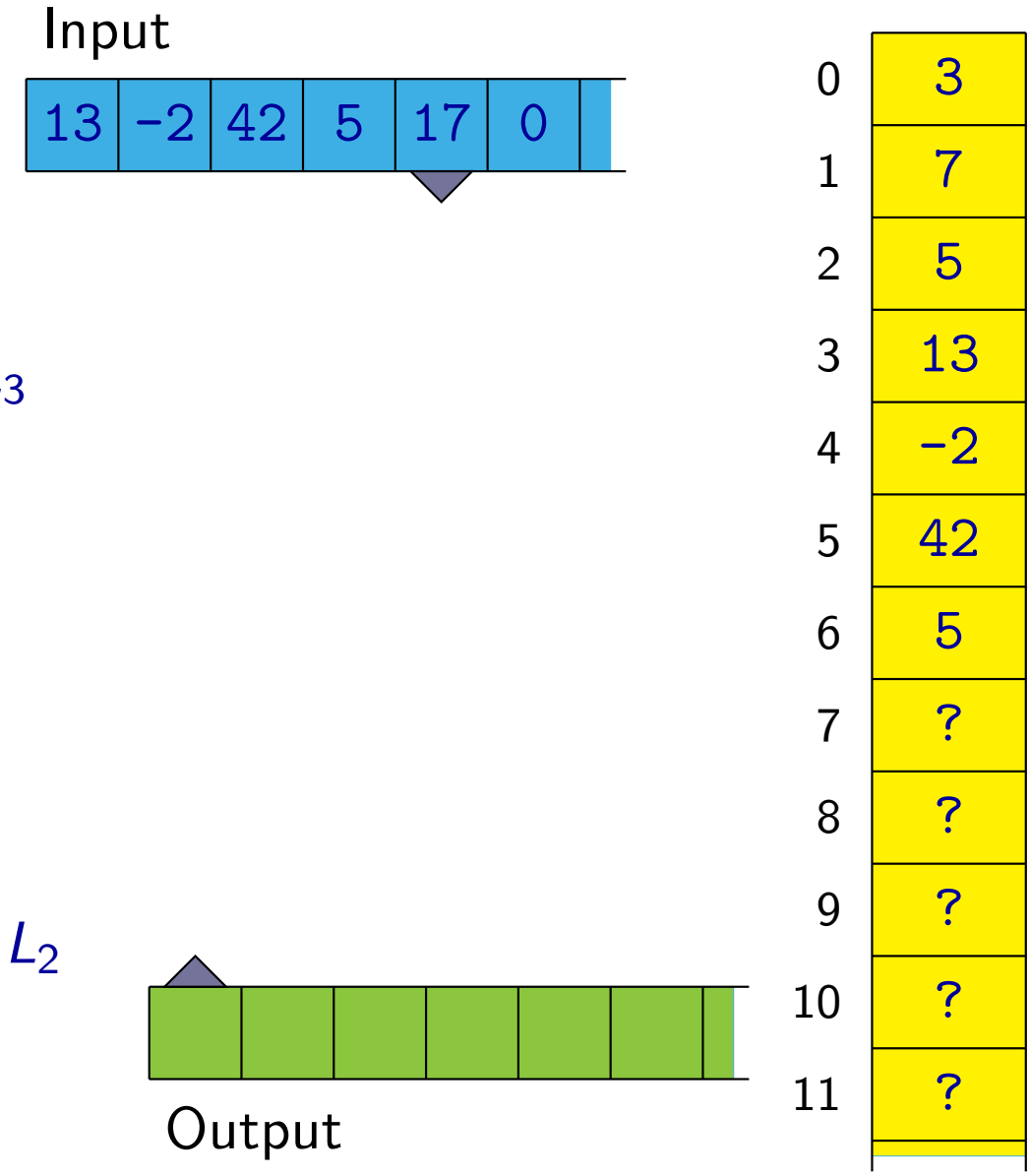
Stroj RAM



$R_0 := 3$
$R_1 := R_0$
$L_1 : R_2 := \text{READ}()$
if $(R_2 = 0)$ **goto** L_3
$[R_1] := R_2$
$R_1 := R_1 + 1$
goto L_1
$L_2 : R_1 := R_1 - 1$
$R_2 := [R_1]$
$\text{WRITE}(R_2)$
$L_3 :$ **if** $(R_1 > R_0)$ **goto** L_2
halt

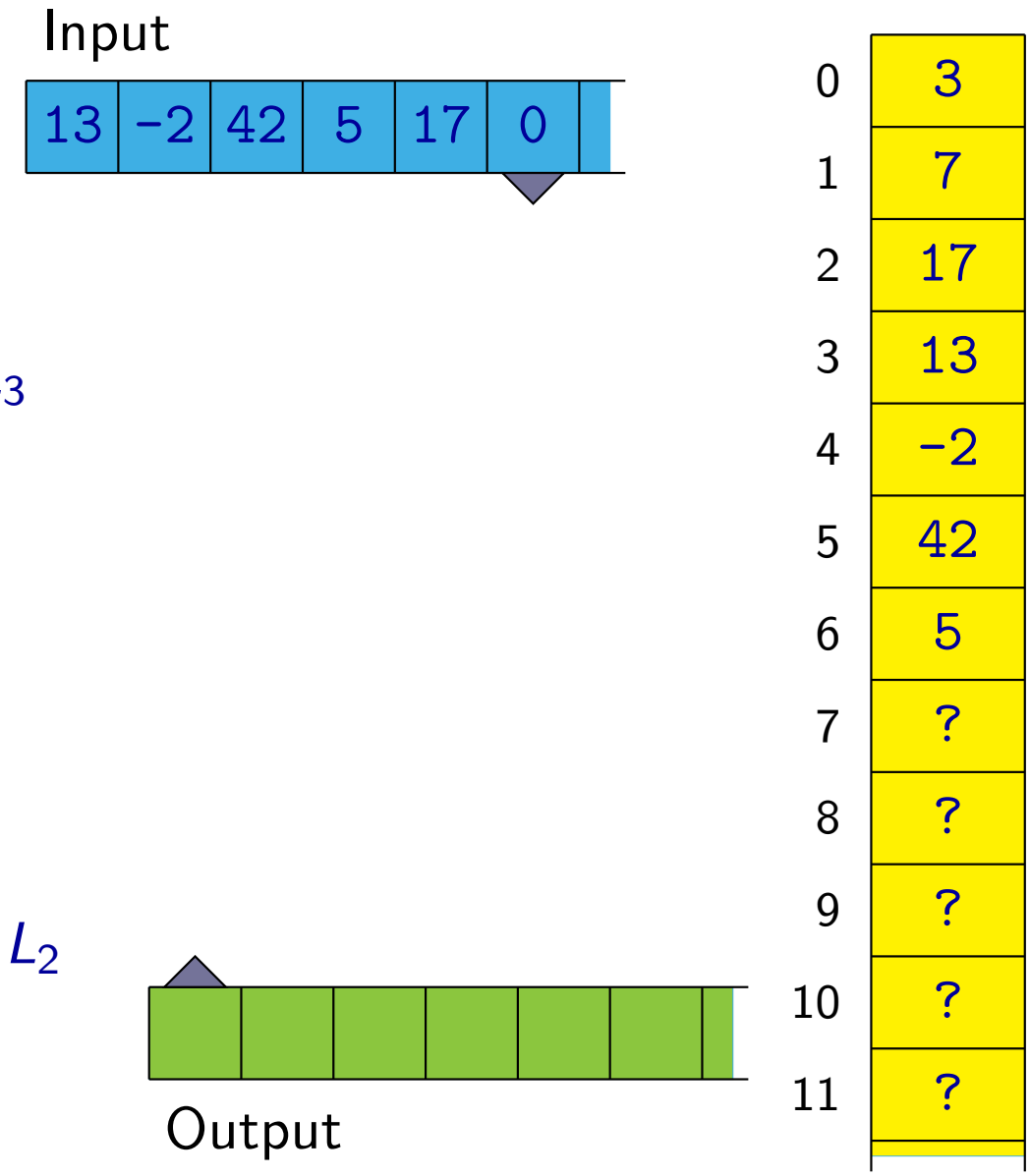
Stroj RAM

$R_0 := 3$
$R_1 := R_0$
$L_1 : R_2 := \text{READ}()$
if $(R_2 = 0)$ **goto** L_3
$[R_1] := R_2$
$R_1 := R_1 + 1$
goto L_1
$L_2 : R_1 := R_1 - 1$
$R_2 := [R_1]$
$\text{WRITE}(R_2)$
$L_3 :$ **if** $(R_1 > R_0)$ **goto** L_2
halt



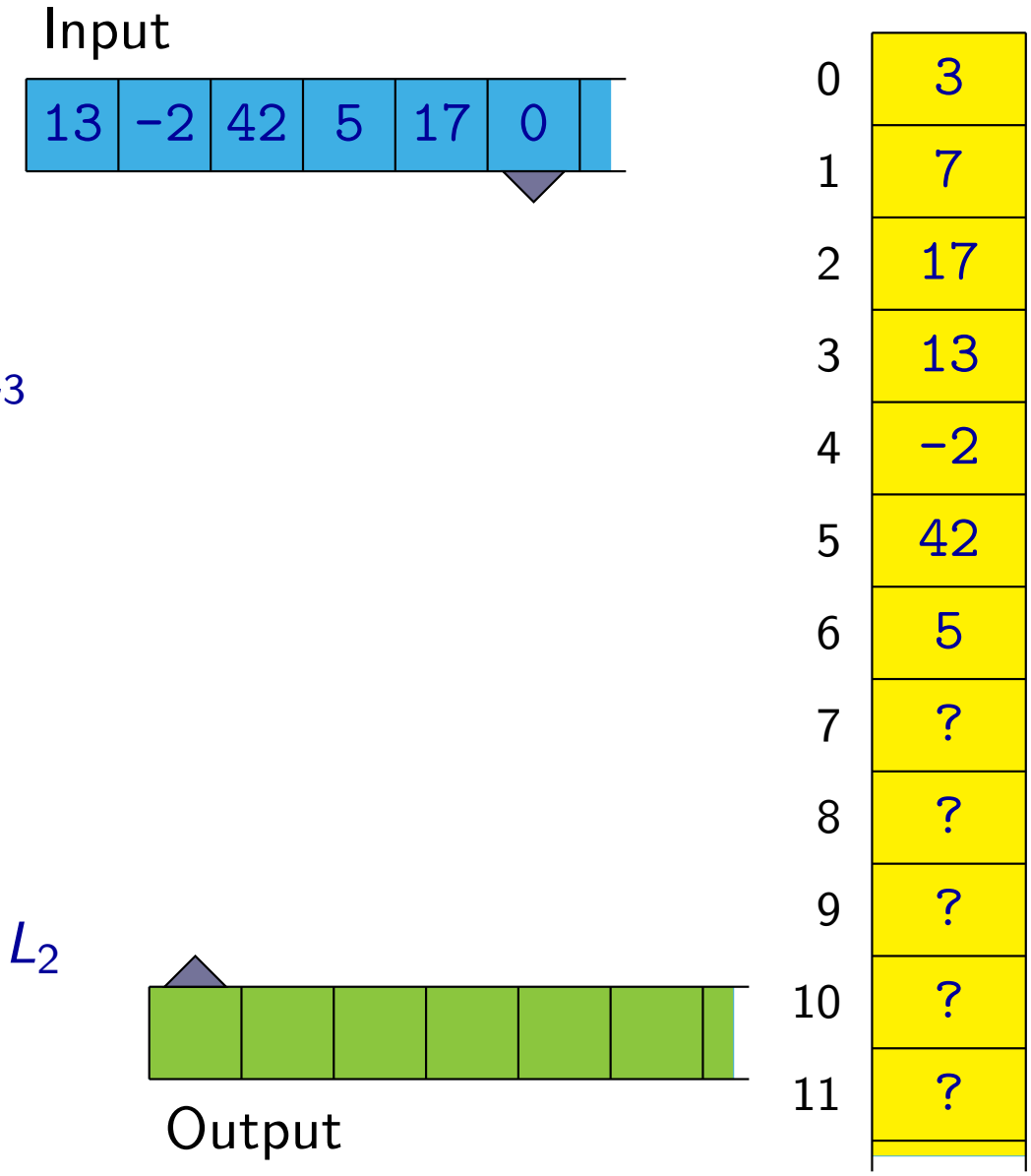
Stroj RAM

$R_0 := 3$
$R_1 := R_0$
$L_1 : R_2 := \text{READ}()$
if $(R_2 = 0)$ **goto** L_3
$[R_1] := R_2$
$R_1 := R_1 + 1$
goto L_1
$L_2 : R_1 := R_1 - 1$
$R_2 := [R_1]$
$\text{WRITE}(R_2)$
$L_3 :$ **if** $(R_1 > R_0)$ **goto** L_2
halt



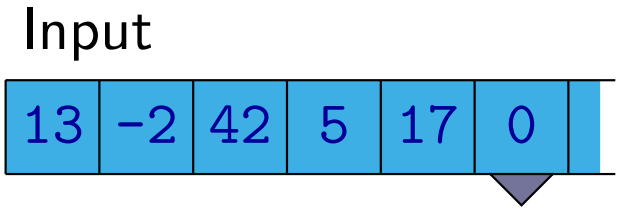
Stroj RAM

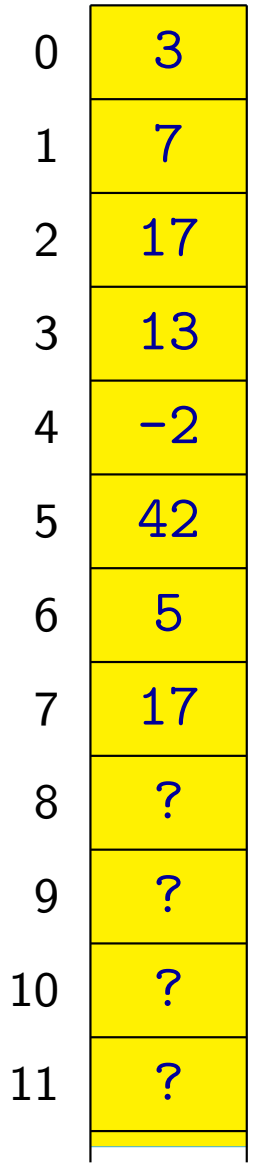
$R_0 := 3$
$R_1 := R_0$
$L_1 : R_2 := \text{READ}()$
if $(R_2 = 0)$ **goto** L_3
$[R_1] := R_2$
$R_1 := R_1 + 1$
goto L_1
$L_2 : R_1 := R_1 - 1$
$R_2 := [R_1]$
$\text{WRITE}(R_2)$
$L_3 :$ **if** $(R_1 > R_0)$ **goto** L_2
halt

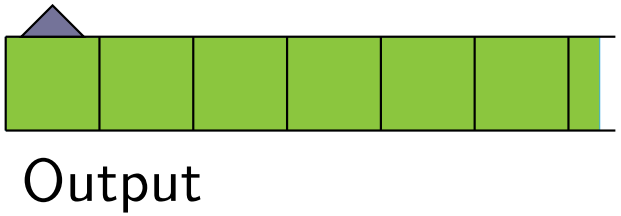


Stroj RAM

$R_0 := 3$
$R_1 := R_0$
$L_1 : R_2 := \text{READ}()$
if $(R_2 = 0)$ **goto** L_3
$[R_1] := R_2$
$R_1 := R_1 + 1$
goto L_1
$L_2 : R_1 := R_1 - 1$
$R_2 := [R_1]$
$\text{WRITE}(R_2)$
$L_3 :$ **if** $(R_1 > R_0)$ **goto** L_2
halt

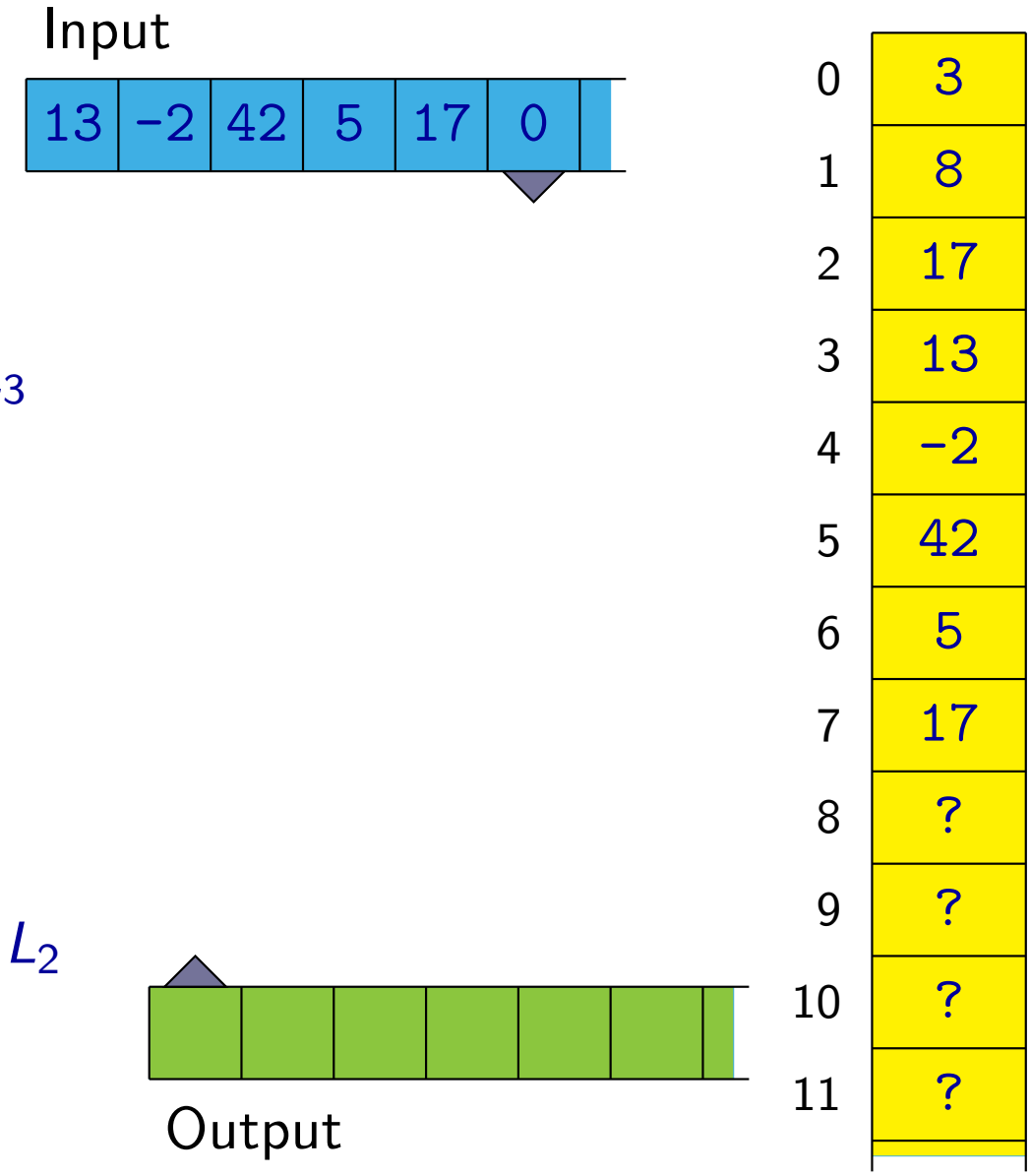






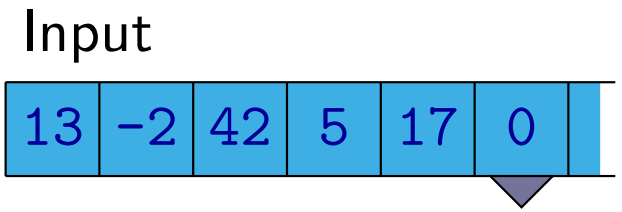
Stroj RAM

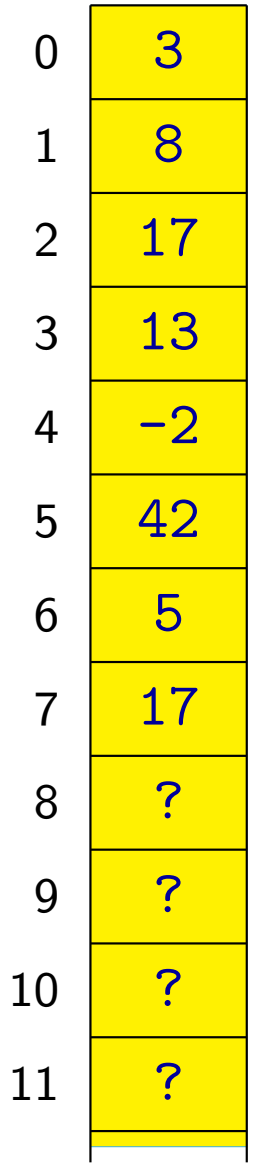
$R_0 := 3$
$R_1 := R_0$
$L_1 : R_2 := \text{READ}()$
if $(R_2 = 0)$ **goto** L_3
$[R_1] := R_2$
$R_1 := R_1 + 1$
goto L_1
$L_2 : R_1 := R_1 - 1$
$R_2 := [R_1]$
$\text{WRITE}(R_2)$
$L_3 :$ **if** $(R_1 > R_0)$ **goto** L_2
halt

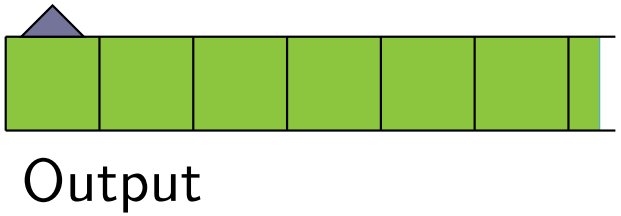


Stroj RAM

$R_0 := 3$
$R_1 := R_0$
$L_1 : R_2 := \text{READ}()$
if $(R_2 = 0)$ **goto** L_3
$[R_1] := R_2$
$R_1 := R_1 + 1$
goto L_1
$L_2 : R_1 := R_1 - 1$
$R_2 := [R_1]$
$\text{WRITE}(R_2)$
$L_3 :$ **if** $(R_1 > R_0)$ **goto** L_2
halt

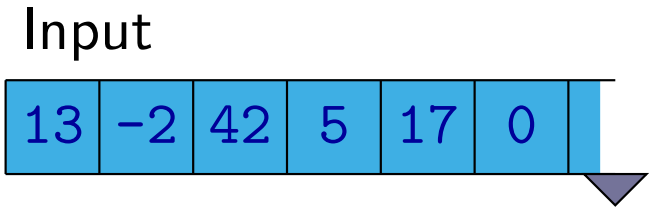


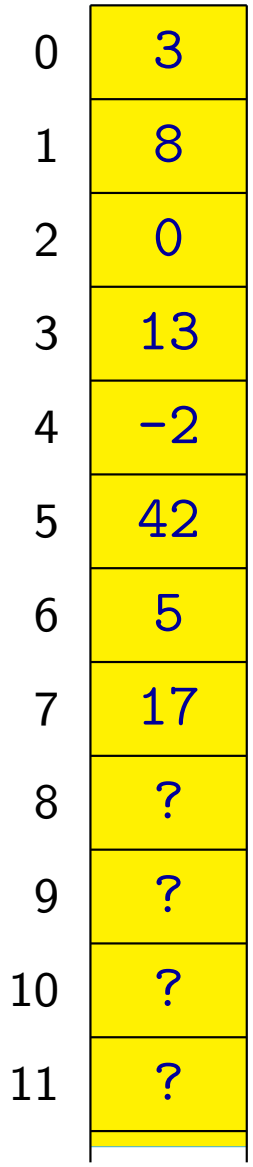


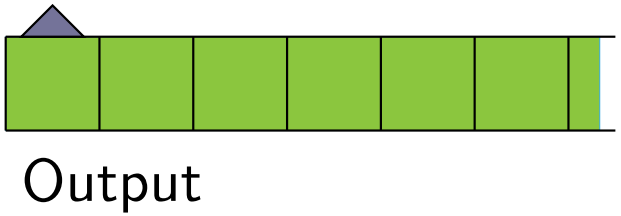


Stroj RAM

$R_0 := 3$
$R_1 := R_0$
$L_1 : R_2 := \text{READ}()$
if $(R_2 = 0)$ **goto** L_3
$[R_1] := R_2$
$R_1 := R_1 + 1$
goto L_1
$L_2 : R_1 := R_1 - 1$
$R_2 := [R_1]$
$\text{WRITE}(R_2)$
$L_3 :$ **if** $(R_1 > R_0)$ **goto** L_2
halt

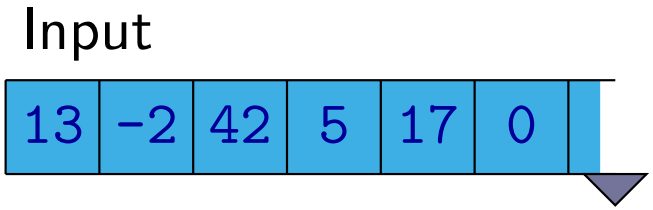


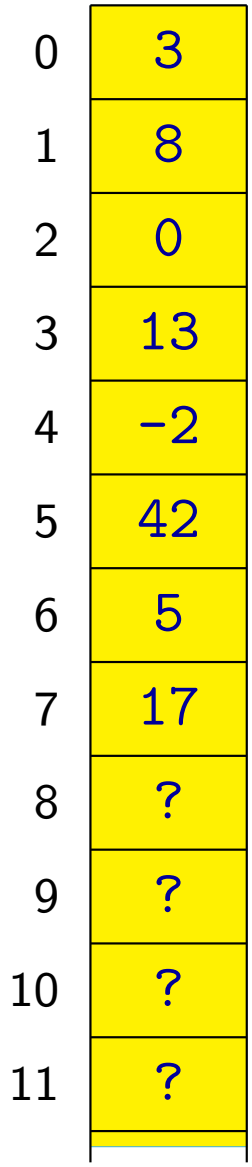


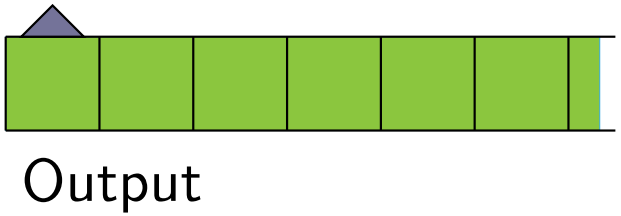


Stroj RAM

$R_0 := 3$
$R_1 := R_0$
$L_1 : R_2 := \text{READ}()$
if $(R_2 = 0)$ **goto** L_3
$[R_1] := R_2$
$R_1 := R_1 + 1$
goto L_1
$L_2 : R_1 := R_1 - 1$
$R_2 := [R_1]$
$\text{WRITE}(R_2)$
$L_3 :$ **if** $(R_1 > R_0)$ **goto** L_2
halt

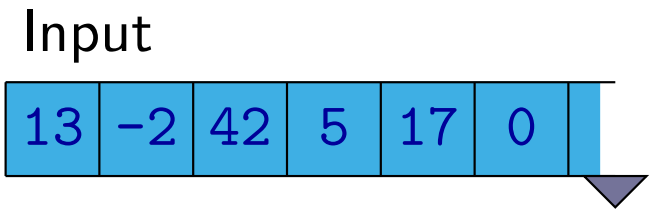


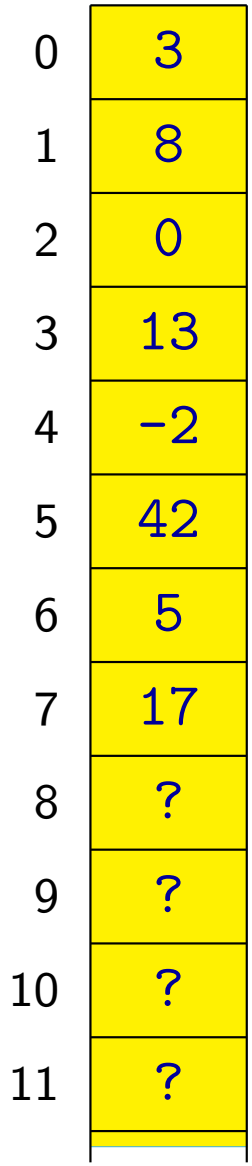


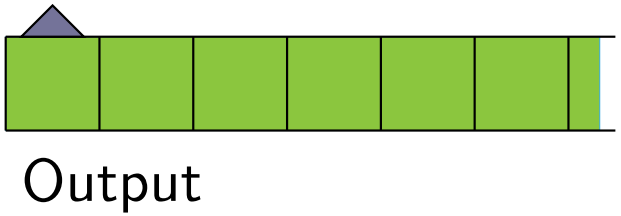


Stroj RAM

$R_0 := 3$
$R_1 := R_0$
$L_1 : R_2 := \text{READ}()$
if $(R_2 = 0)$ **goto** L_3
$[R_1] := R_2$
$R_1 := R_1 + 1$
goto L_1
$L_2 : R_1 := R_1 - 1$
$R_2 := [R_1]$
$\text{WRITE}(R_2)$
$L_3 :$ **if** $(R_1 > R_0)$ **goto** L_2
halt

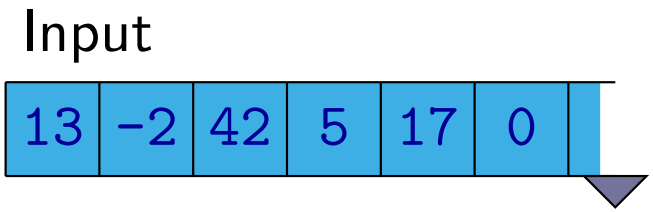


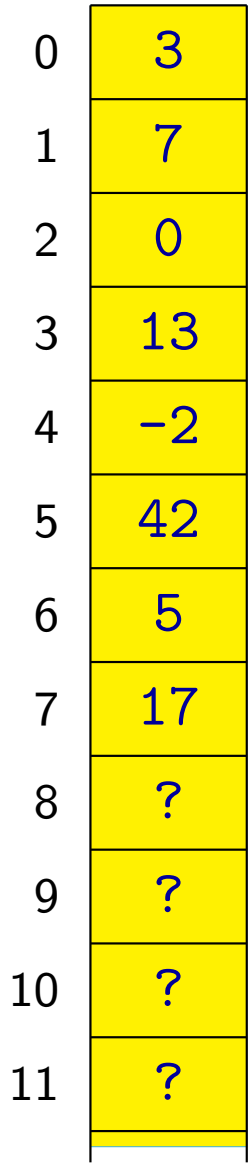


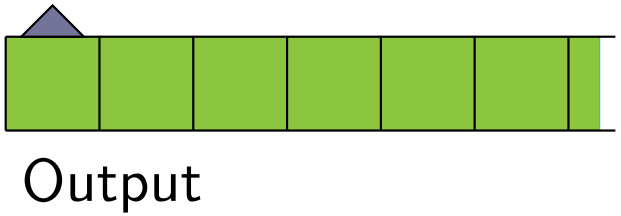


Stroj RAM

$R_0 := 3$
$R_1 := R_0$
$L_1 : R_2 := \text{READ}()$
if $(R_2 = 0)$ **goto** L_3
$[R_1] := R_2$
$R_1 := R_1 + 1$
goto L_1
$L_2 : R_1 := R_1 - 1$
$R_2 := [R_1]$
$\text{WRITE}(R_2)$
$L_3 :$ **if** $(R_1 > R_0)$ **goto** L_2
halt

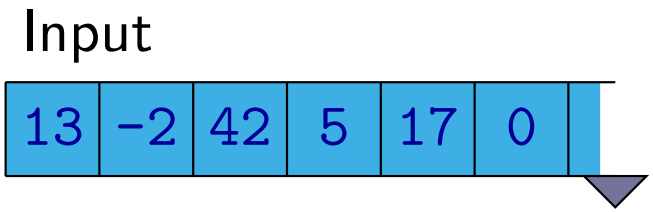


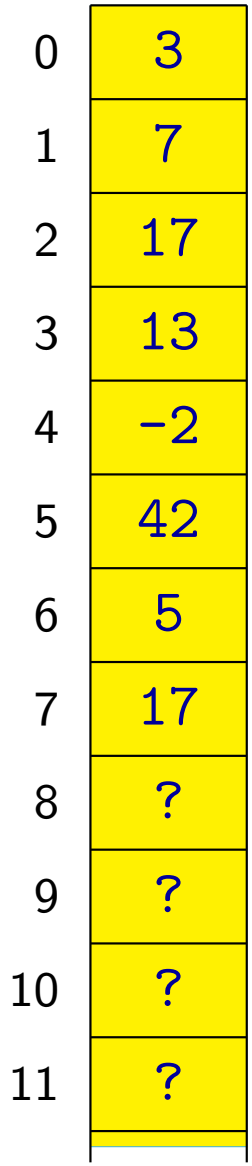


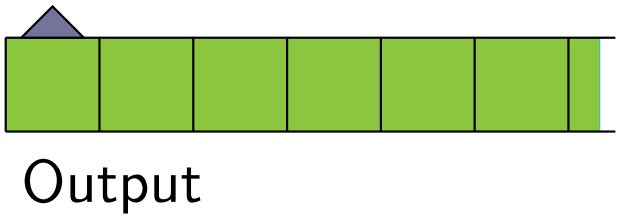


Stroj RAM

$R_0 := 3$
$R_1 := R_0$
$L_1 : R_2 := \text{READ}()$
if $(R_2 = 0)$ **goto** L_3
$[R_1] := R_2$
$R_1 := R_1 + 1$
goto L_1
$L_2 : R_1 := R_1 - 1$
$R_2 := [R_1]$
$\text{WRITE}(R_2)$
$L_3 :$ **if** $(R_1 > R_0)$ **goto** L_2
halt

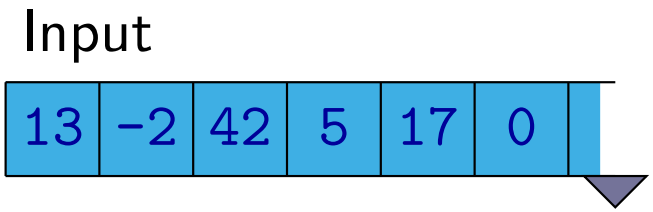


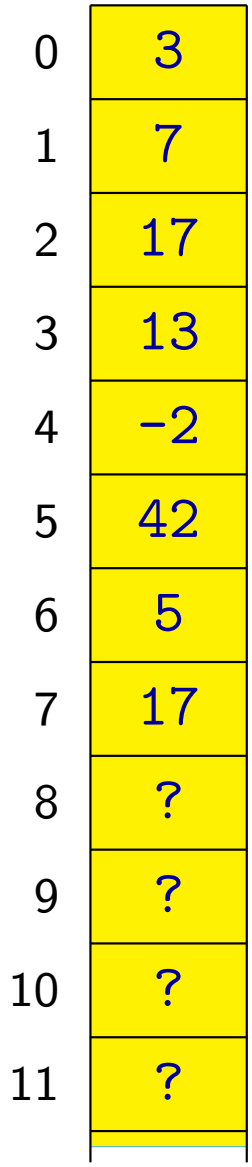


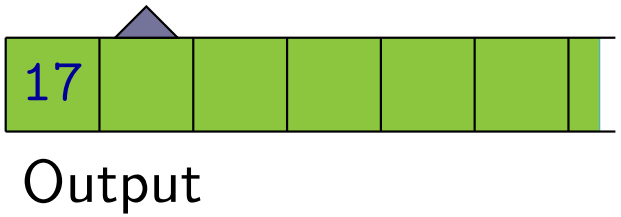


Stroj RAM

$R_0 := 3$
$R_1 := R_0$
$L_1 : R_2 := \text{READ}()$
if $(R_2 = 0)$ **goto** L_3
$[R_1] := R_2$
$R_1 := R_1 + 1$
goto L_1
$L_2 : R_1 := R_1 - 1$
$R_2 := [R_1]$
$\text{WRITE}(R_2)$
$L_3 :$ **if** $(R_1 > R_0)$ **goto** L_2
halt

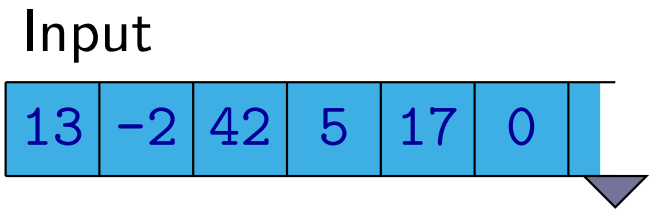


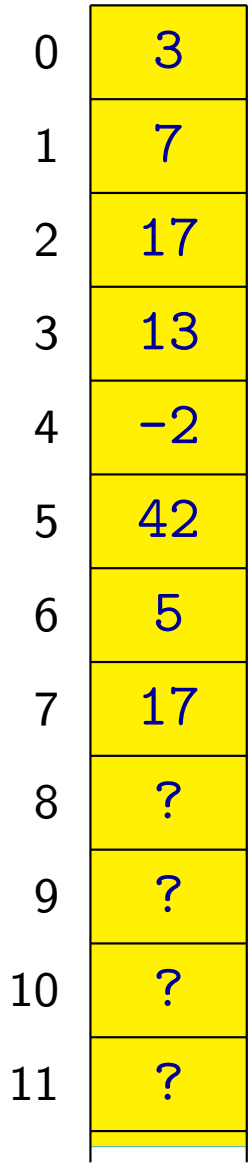


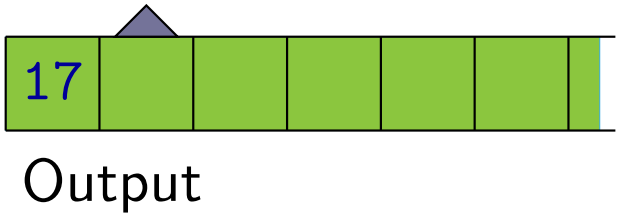


Stroj RAM

$R_0 := 3$
$R_1 := R_0$
$L_1 : R_2 := \text{READ}()$
if $(R_2 = 0)$ **goto** L_3
$[R_1] := R_2$
$R_1 := R_1 + 1$
goto L_1
$L_2 : R_1 := R_1 - 1$
$R_2 := [R_1]$
$\text{WRITE}(R_2)$
$L_3 :$ **if** $(R_1 > R_0)$ **goto** L_2
halt

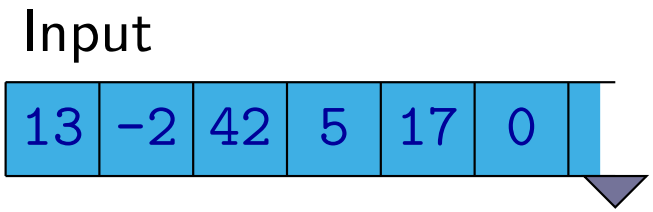


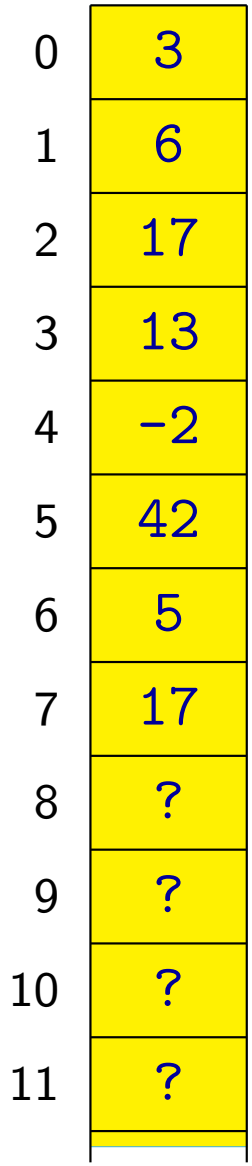


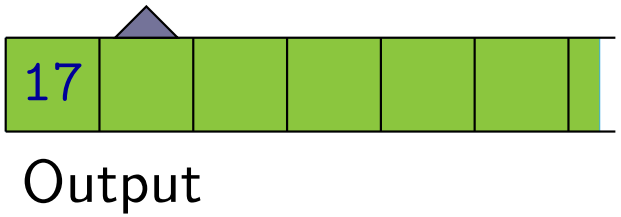


Stroj RAM

$R_0 := 3$
$R_1 := R_0$
$L_1 : R_2 := \text{READ}()$
if $(R_2 = 0)$ **goto** L_3
$[R_1] := R_2$
$R_1 := R_1 + 1$
goto L_1
$L_2 : R_1 := R_1 - 1$
$R_2 := [R_1]$
$\text{WRITE}(R_2)$
$L_3 :$ **if** $(R_1 > R_0)$ **goto** L_2
halt

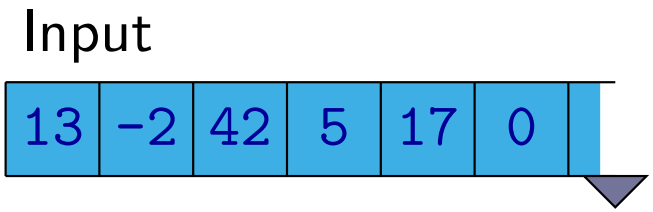


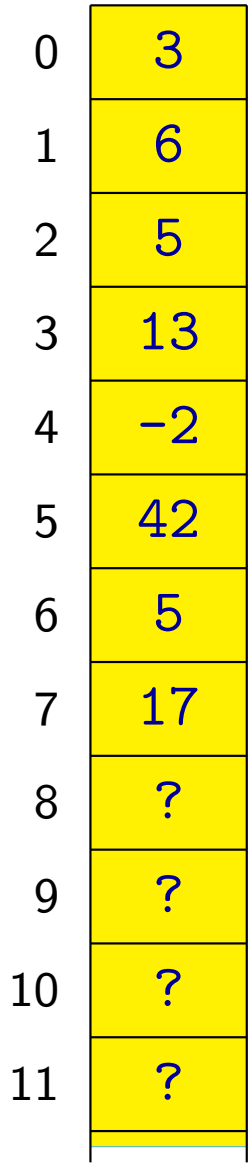


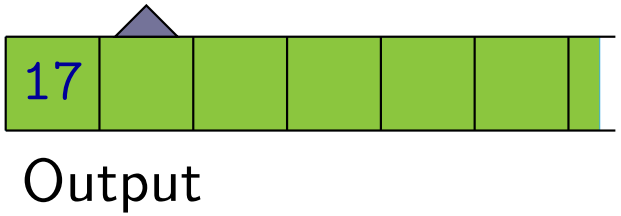


Stroj RAM

$R_0 := 3$
$R_1 := R_0$
$L_1 : R_2 := \text{READ}()$
if $(R_2 = 0)$ **goto** L_3
$[R_1] := R_2$
$R_1 := R_1 + 1$
goto L_1
$L_2 : R_1 := R_1 - 1$
$R_2 := [R_1]$
$\text{WRITE}(R_2)$
$L_3 :$ **if** $(R_1 > R_0)$ **goto** L_2
halt

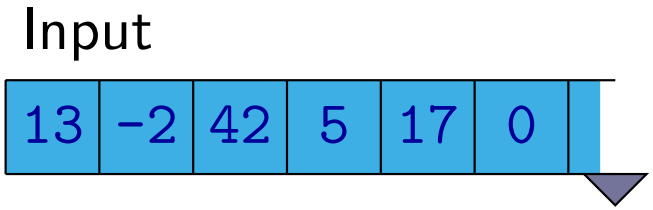


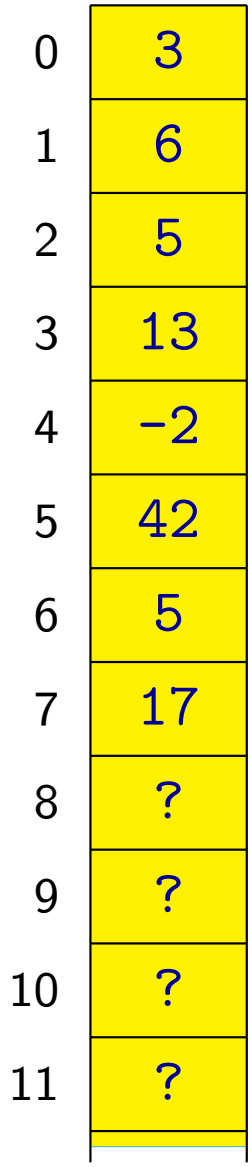


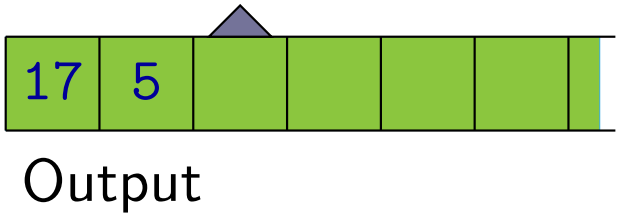


Stroj RAM

$R_0 := 3$
$R_1 := R_0$
$L_1 : R_2 := \text{READ}()$
if $(R_2 = 0)$ **goto** L_3
$[R_1] := R_2$
$R_1 := R_1 + 1$
goto L_1
$L_2 : R_1 := R_1 - 1$
$R_2 := [R_1]$
$\text{WRITE}(R_2)$
$L_3 :$ **if** $(R_1 > R_0)$ **goto** L_2
halt

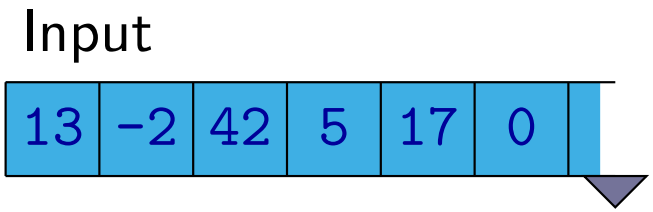


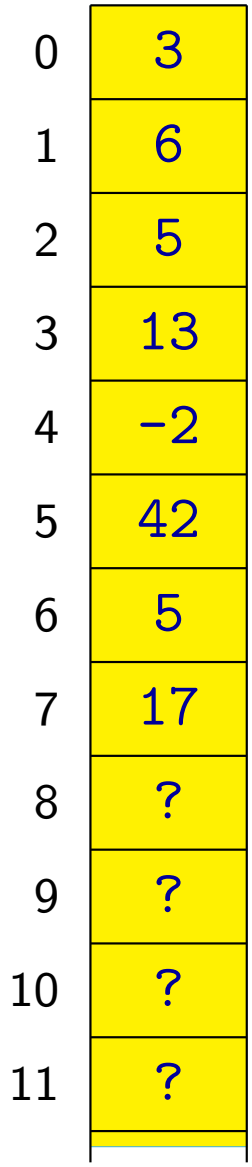


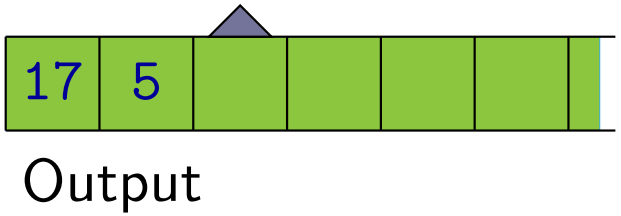


Stroj RAM

$R_0 := 3$
$R_1 := R_0$
$L_1 : R_2 := \text{READ}()$
if $(R_2 = 0)$ **goto** L_3
$[R_1] := R_2$
$R_1 := R_1 + 1$
goto L_1
$L_2 : R_1 := R_1 - 1$
$R_2 := [R_1]$
$\text{WRITE}(R_2)$
$L_3 :$ **if** $(R_1 > R_0)$ **goto** L_2
halt

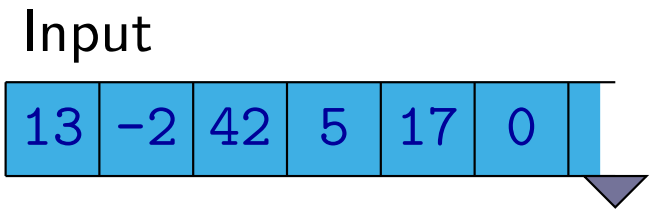


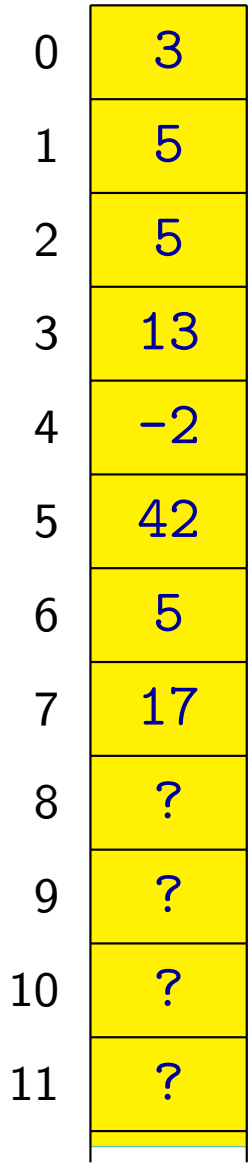


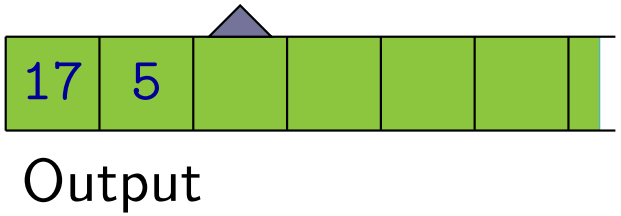


Stroj RAM

$R_0 := 3$
$R_1 := R_0$
$L_1 : R_2 := \text{READ}()$
if $(R_2 = 0)$ **goto** L_3
$[R_1] := R_2$
$R_1 := R_1 + 1$
goto L_1
$L_2 : R_1 := R_1 - 1$
$R_2 := [R_1]$
$\text{WRITE}(R_2)$
$L_3 :$ **if** $(R_1 > R_0)$ **goto** L_2
halt

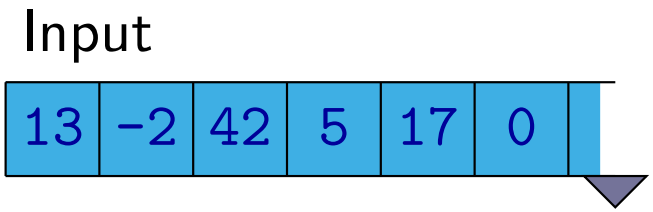


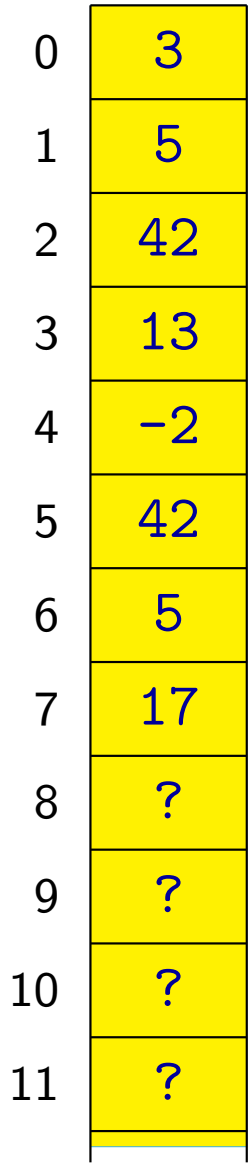


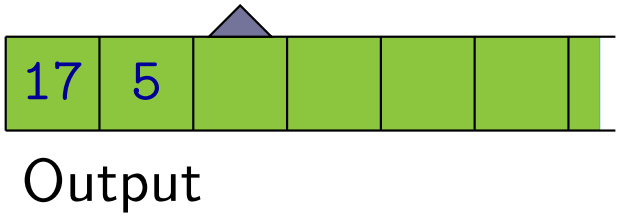


Stroj RAM

$R_0 := 3$
$R_1 := R_0$
$L_1 : R_2 := \text{READ}()$
if $(R_2 = 0)$ **goto** L_3
$[R_1] := R_2$
$R_1 := R_1 + 1$
goto L_1
$L_2 : R_1 := R_1 - 1$
$R_2 := [R_1]$
$\text{WRITE}(R_2)$
$L_3 :$ **if** $(R_1 > R_0)$ **goto** L_2
halt

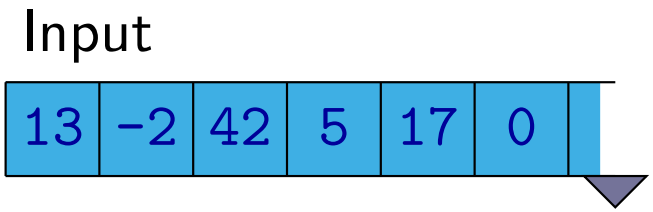


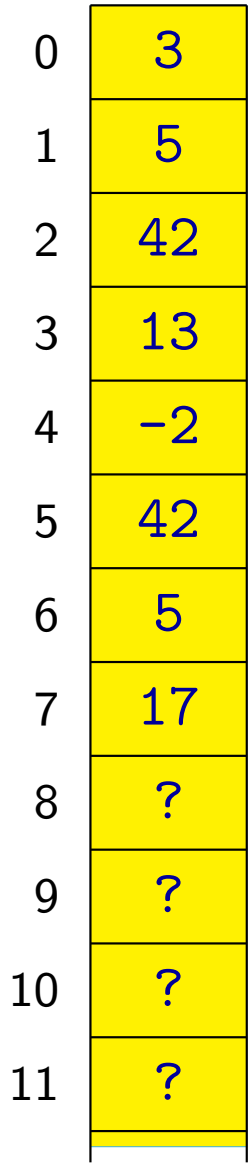


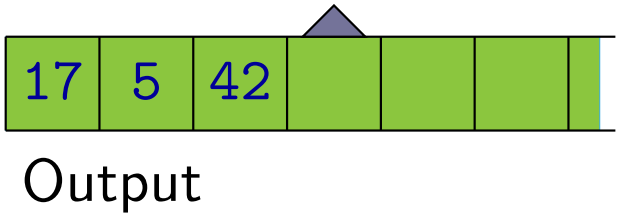


Stroj RAM

$R_0 := 3$
$R_1 := R_0$
$L_1 : R_2 := \text{READ}()$
if $(R_2 = 0)$ **goto** L_3
$[R_1] := R_2$
$R_1 := R_1 + 1$
goto L_1
$L_2 : R_1 := R_1 - 1$
$R_2 := [R_1]$
$\text{WRITE}(R_2)$
$L_3 :$ **if** $(R_1 > R_0)$ **goto** L_2
halt

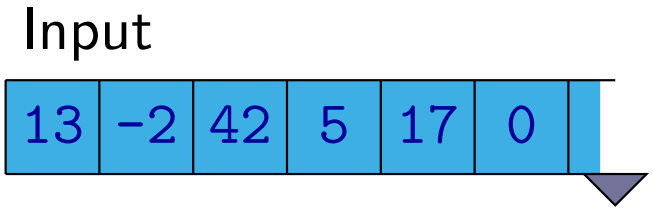


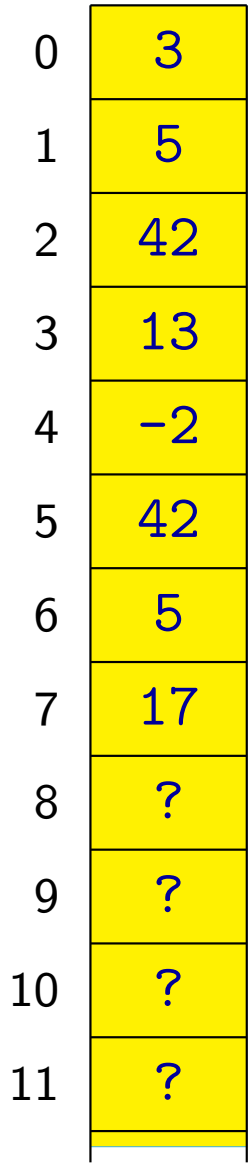


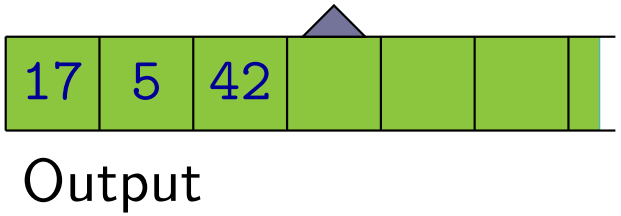


Stroj RAM

$R_0 := 3$
$R_1 := R_0$
$L_1 : R_2 := \text{READ}()$
if $(R_2 = 0)$ **goto** L_3
$[R_1] := R_2$
$R_1 := R_1 + 1$
goto L_1
$L_2 : R_1 := R_1 - 1$
$R_2 := [R_1]$
$\text{WRITE}(R_2)$
$L_3 :$ **if** $(R_1 > R_0)$ **goto** L_2
halt

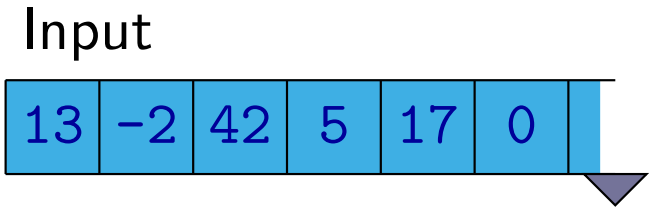


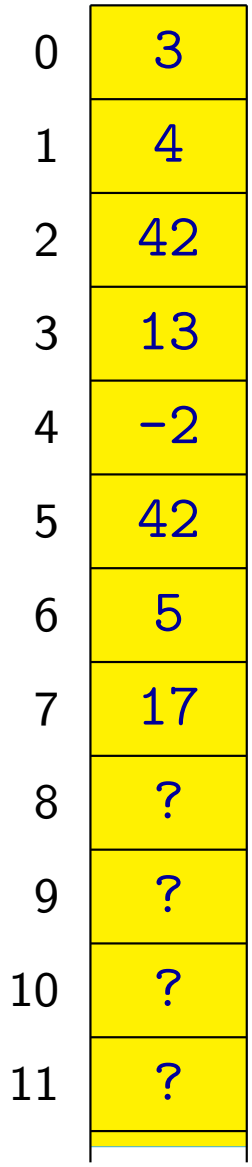


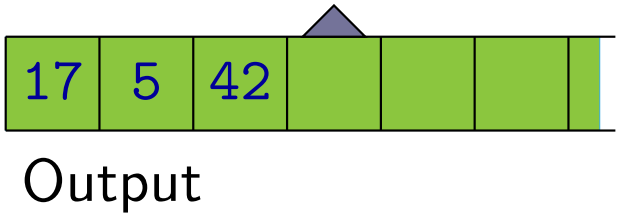


Stroj RAM

$R_0 := 3$
$R_1 := R_0$
$L_1 : R_2 := \text{READ}()$
if $(R_2 = 0)$ **goto** L_3
$[R_1] := R_2$
$R_1 := R_1 + 1$
goto L_1
$L_2 : R_1 := R_1 - 1$
$R_2 := [R_1]$
$\text{WRITE}(R_2)$
$L_3 :$ **if** $(R_1 > R_0)$ **goto** L_2
halt

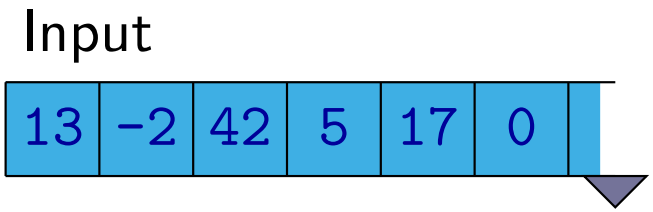


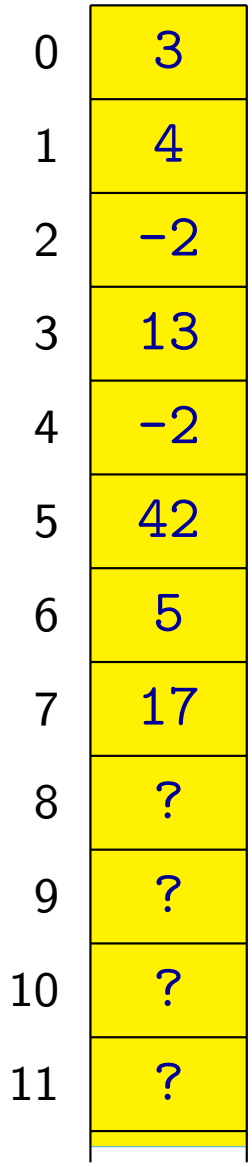


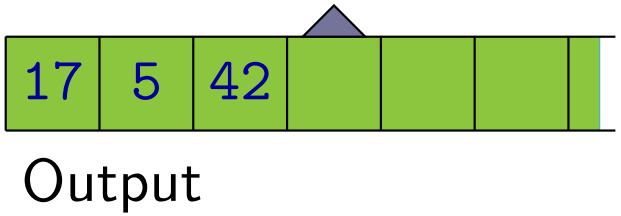


Stroj RAM

$R_0 := 3$
$R_1 := R_0$
$L_1 : R_2 := \text{READ}()$
if $(R_2 = 0)$ **goto** L_3
$[R_1] := R_2$
$R_1 := R_1 + 1$
goto L_1
$L_2 : R_1 := R_1 - 1$
$R_2 := [R_1]$
$\text{WRITE}(R_2)$
$L_3 :$ **if** $(R_1 > R_0)$ **goto** L_2
halt

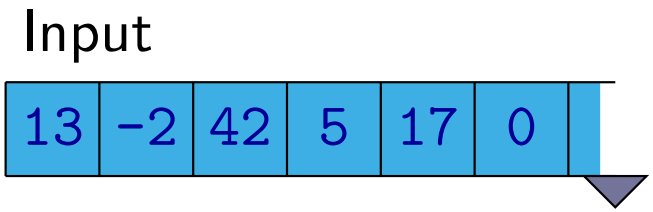


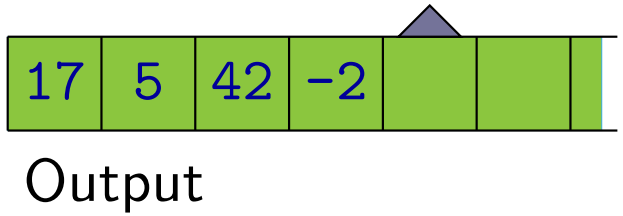


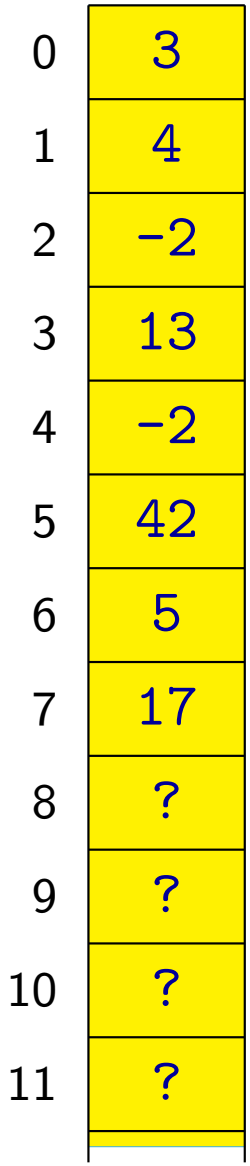


Stroj RAM

$R_0 := 3$
$R_1 := R_0$
$L_1 : R_2 := \text{READ}()$
if $(R_2 = 0)$ **goto** L_3
$[R_1] := R_2$
$R_1 := R_1 + 1$
goto L_1
$L_2 : R_1 := R_1 - 1$
$R_2 := [R_1]$
$\text{WRITE}(R_2)$
$L_3 :$ **if** $(R_1 > R_0)$ **goto** L_2
halt

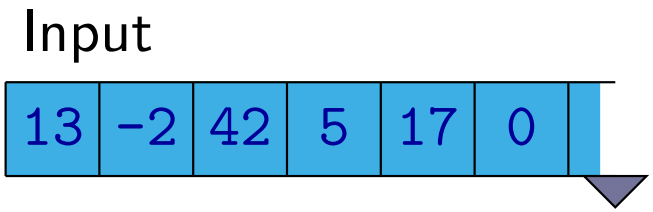


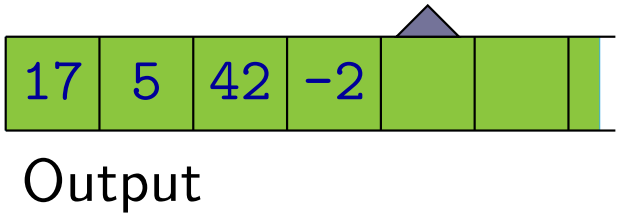


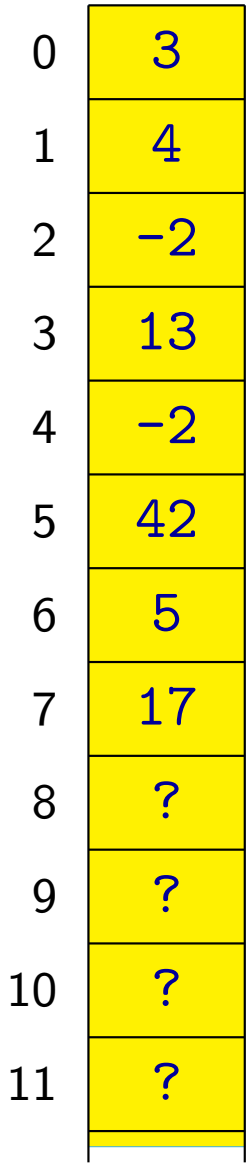


Stroj RAM

$R_0 := 3$
$R_1 := R_0$
$L_1 : R_2 := \text{READ}()$
if $(R_2 = 0)$ **goto** L_3
$[R_1] := R_2$
$R_1 := R_1 + 1$
goto L_1
$L_2 : R_1 := R_1 - 1$
$R_2 := [R_1]$
$\text{WRITE}(R_2)$
$L_3 :$ **if** $(R_1 > R_0)$ **goto** L_2
halt

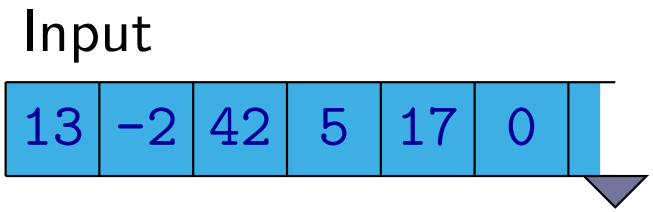


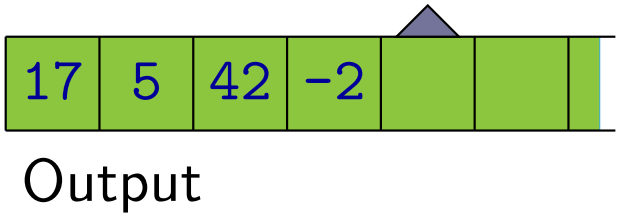


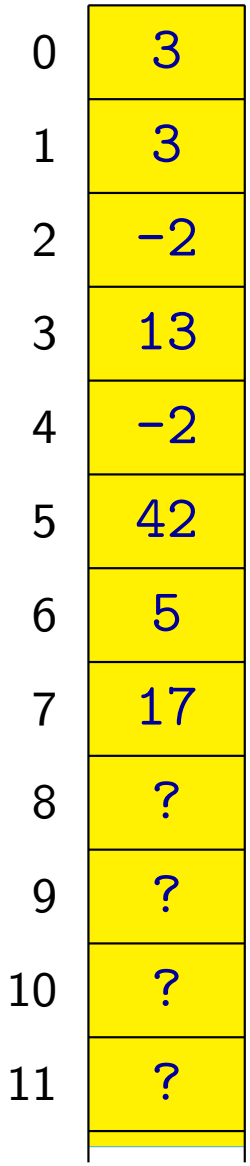


Stroj RAM

$R_0 := 3$
$R_1 := R_0$
$L_1 : R_2 := \text{READ}()$
if $(R_2 = 0)$ **goto** L_3
$[R_1] := R_2$
$R_1 := R_1 + 1$
goto L_1
$L_2 : R_1 := R_1 - 1$
$R_2 := [R_1]$
$\text{WRITE}(R_2)$
$L_3 :$ **if** $(R_1 > R_0)$ **goto** L_2
halt

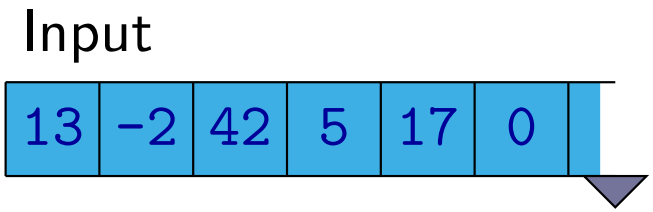


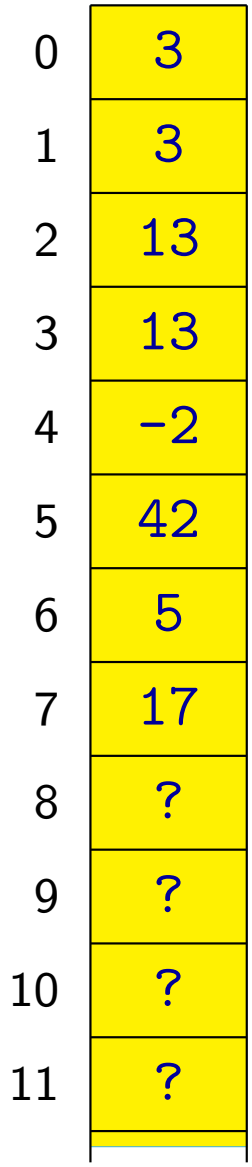


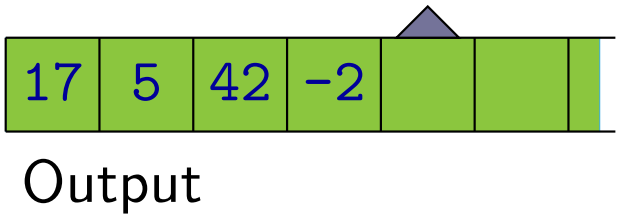


Stroj RAM

$R_0 := 3$
$R_1 := R_0$
$L_1 : R_2 := \text{READ}()$
if $(R_2 = 0)$ **goto** L_3
$[R_1] := R_2$
$R_1 := R_1 + 1$
goto L_1
$L_2 : R_1 := R_1 - 1$
$R_2 := [R_1]$
$\text{WRITE}(R_2)$
$L_3 :$ **if** $(R_1 > R_0)$ **goto** L_2
halt

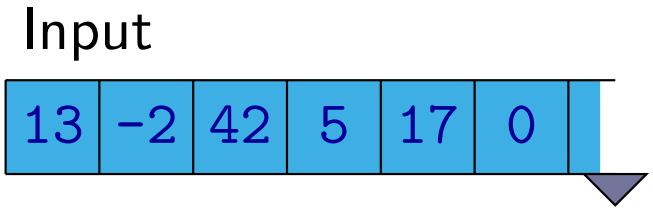


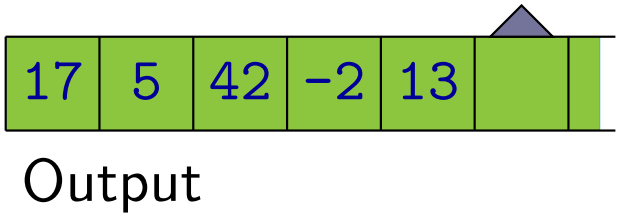


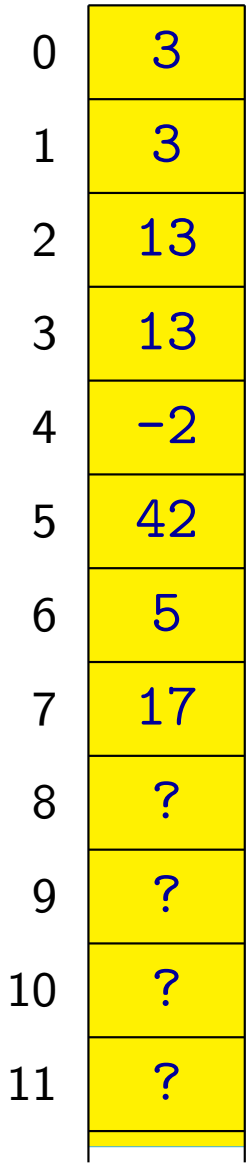


Stroj RAM

$R_0 := 3$
$R_1 := R_0$
$L_1 : R_2 := \text{READ}()$
if $(R_2 = 0)$ **goto** L_3
$[R_1] := R_2$
$R_1 := R_1 + 1$
goto L_1
$L_2 : R_1 := R_1 - 1$
$R_2 := [R_1]$
$\text{WRITE}(R_2)$
$L_3 :$ **if** $(R_1 > R_0)$ **goto** L_2
halt

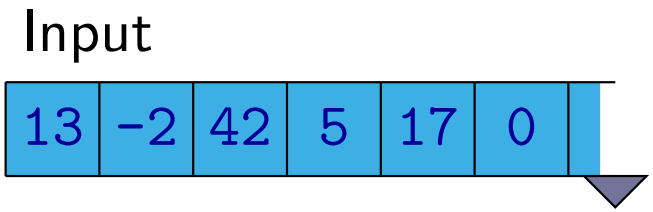


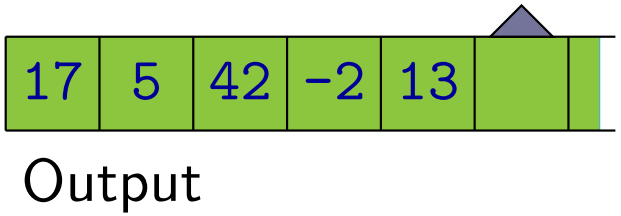


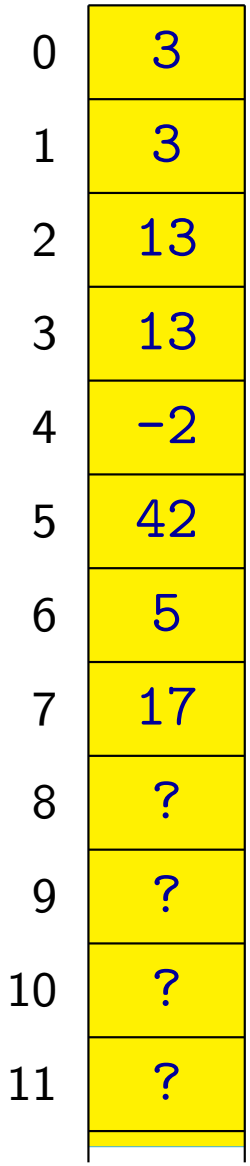


Stroj RAM

$R_0 := 3$
$R_1 := R_0$
$L_1 : R_2 := \text{READ}()$
if $(R_2 = 0)$ **goto** L_3
$[R_1] := R_2$
$R_1 := R_1 + 1$
goto L_1
$L_2 : R_1 := R_1 - 1$
$R_2 := [R_1]$
$\text{WRITE}(R_2)$
$L_3 :$ **if** $(R_1 > R_0)$ **goto** L_2
halt







Stroj RAM

Rozdíly oproti skutečnému počítači:

- Velikost paměti není omezena (adresa může být libovolné přirozené číslo).
- Velikost obsahu jednotlivých buněk není omezena (buňka může obsahovat libovolné celé číslo).
- Čte data sekvenčně ze vstupu, který je tvořen sekvencí celých čísel. Ze vstupu lze pouze číst.
- Zapisuje data sekvenčně na výstup, který je tvořen sekvencí celých čísel. Na výstup je možné pouze zapisovat.

Stroj RAM

- Operace jako přístup k buňce paměti na adrese menší než nula nebo dělení nulou vedou k chybě — výpočet se (neúspěšně) zastaví.

- Co se týká počátečního obsahu paměti, jsou dvě možnosti, jak ho definovat:
  - Všechny buňky jsou inicializovány hodnotou 0.
  - Čtení obsahu buňky, do které nebylo dosud nic zapsáno, způsobí chybu.

    Buňky na začátku obsahují speciální hodnotu (označenou zde symbolem '?'), která reprezentuje to, že buňka nebyla dosud inicializována.

- Uvažují se i varianty strojů RAM, kde buňky paměti (a vstupu a výstupu) neobsahují celá čísla (tj. prvky množiny $\mathbb{Z}$), ale mohou obsahovat jen přirozená čísla (tj. prvky množiny $\mathbb{N}$).

  Například operace odčítání ($R_i := R_j - R_k$) se pak chová tak, že pokud by výsledkem mělo být záporné číslo, je jako výsledek operace přiřazena hodnota 0.

Stroj RAM

- Různé varianty strojů RAM se mohou lišit tím, jaké konkrétní operace v aritmetických instrukcích podporují nebo naopak nepodporují.

  Například:

  - podpora bitových operací (and, or, not, xor, ...), bitový posunů, ...
  - varianta stroje RAM, která nemá operace násobení a dělení

- Mohli bychom také uvažovat variantu stroje RAM, kde místo instrukcí tvaru

  $$\textbf{if } (R_i \ rel \ R_j) \textbf{ goto } \ell \qquad \text{nebo} \qquad \textbf{if } (R_i \ rel \ c) \textbf{ goto } \ell$$

  jsou všechny podmíněné skoky jen tvaru

  $$\textbf{if } (R_i \ rel \ 0) \textbf{ goto } \ell$$

  Místo všech relací $\{=, \neq, \leq, \geq, <, >\}$ může být podporována jen nějaká podmnožina z nich, např. $\{=, >\}$.

Stroj RAM

- V některých variantách stroje RAM nemají vstup a výstup podobu sekvence čísel.

  Místo toho pracuje stroj z hlediska vstupu a výstupu s páskami obsahujícími sekvence symbolů z nějaké dané abecedy, např. $\{0, 1\}$.

  Stroj má pak například instrukce, které mu umožňují větvit výpočet podle symbolu přečteného ze vstupu.

  Vnitřní paměť ovšem i v této variantě pracuje s čísly.

- Pokud má stroj jako výsledek dávat jen odpověď Ano/Ne (tj. přijmout nebo nepřijmout daný vstup), nemusí mít výstupní pásku.

  Instrukce **halt** je pak nahrazena instrukcemi **accept** a **reject**.

Stroj RAM

- Ve standardní definici stroje RAM se většinou neuvažují instrukce skoku na adresu instrukce uloženou v buňce paměti, tj. instrukce typu

  goto R_i

  Stroj RAM bychom mohli rozšířit o tento druh instrukcí.

- Jako standardní se u stroje RAM bere to, že kód programu není uložen v pracovní paměti, ale má zvláštní samostatnou paměť, která je jen pro čtení.

  V průběhu výpočtu se tedy kód programu nemůže měnit.

Stroj RASP

- Druh stroje podobný stroji RAM, kde je ovšem program uložen v pracovní paměti (instrukce jsou kódovány čísly) a je možné ho průběhu výpočtu měnit, se označuje jako stroj **RASP** (**random-access stored program**).

  Stroj RASP tak umožňuje provádět činnost sebemodifikujících se programů.

Turingův stroj simulující činnost stroje RAM

Není těžké si rozmyslet, že činnost libovolného Turingova stroje je možné simulovat pomocí stroje RAM.

Promyslet si, že i naopak činnost každého stroje RAM je možné simulovat Turingovým strojem, je o něco komplikovanější.

Při popisu toho, jak simulovat činnost stroje RAM pomocí Turingova stroje, budeme postupovat po menších krocích:

- Ukážeme, jak činnost stroje RAM ve variantě, kterou jsme si popsali, simulovat variantou stroje RAM s poněkud jednodušími instrukcemi.
- Ukážeme, jak činnost této jednodušší varianty stroje RAM simulovat vícepáskovým Turingovým strojem.
- Už dříve jsme viděli, jak činnosti vícepáskového Turingovat stroje simulovat pomocí jednopáskového Turingovat stroje.

Jednodušší varianta stroje RAM

Tato jednodušší varianta stroje RAM bude mít kromě pracovní paměti tři **registry**:

- **registr A** — téměř všechny instrukce pracují s tímto registrem, výsledky všech operací se ukládají do tohoto registru

  Poznámka: Tento druh registru se často označuje jako **akumulátor**.

- **registr B** — tento registr slouží k uložení druhého operandu pro aritmetické instrukce (první operand je vždy v akumulátoru)

- **registr C** — tento registr slouží k uložení adresy, na kterou bude zapisovat instrukce store

Jednodušší varianta stroje RAM

Přehled instrukcí:

$A := c$	– přiřazení konstanty
$B := A$	– přiřazení do registru B
$C := A$	– přiřazení do registru C
$A := [A]$	– load (čtení z paměti)
$[C] := A$	– store (zápis do paměti)
$A := A \; op \; B$	– aritmetické instrukce, $op \in \{+, -, *, /\}$
if $(A \; rel \; 0)$ **goto** ℓ	– podmíněný skok, $rel \in \{=, \neq, \leq, \geq, <, >\}$
goto ℓ	– nepodmíněný skok
$A :=$ READ $()$	– čtení ze vstupu
WRITE (A)	– zápis na výstup
halt	– zastavení programu

Jednodušší varianta stroje RAM

Například instrukce

$$R_7 := R_3 + R_6$$

může být nahrazena posloupností instrukcí:

$$A := 7$$
$$C := A$$
$$A := 6$$
$$A := [A]$$
$$B := A$$
$$A := 3$$
$$A := [A]$$
$$A := A + B$$
$$[C] := A$$

Jednodušší varianta stroje RAM

Například instrukce

$$[R_{15}] := R_9$$

může být nahrazena posloupností instrukcí:

$$A := 15$$
$$A := [A]$$
$$C := A$$
$$A := 9$$
$$A := [A]$$
$$[C] := A$$

Jednodušší varianta stroje RAM

Například instrukce

$$\textbf{if } (R_4 \geq R_{11}) \textbf{ goto } \ell$$

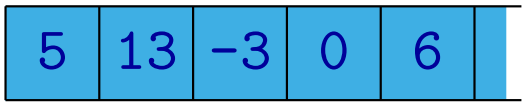
může být nahrazena posloupností instrukcí:

$$A := 11$$
$$A := [A]$$
$$B := A$$
$$A := 4$$
$$A := [A]$$
$$A := A - B$$
$$\textbf{if } (A \geq 0) \textbf{ goto } \ell$$

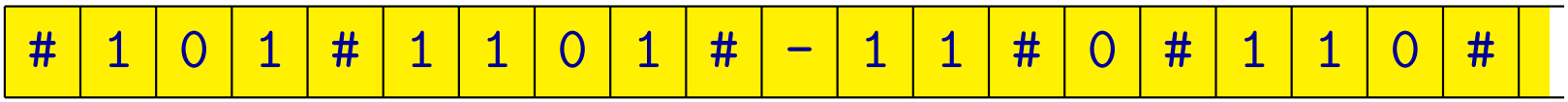
Turingův stroj simulující činnost stroje RAM

Turingův stroj pracuje se slovy nad nějakou abecedou, zatímco stroj RAM s čísly. Čísla ale můžeme zapisovat jako sekvence symbolů a naopak symboly nějaké abecedy můžeme zapisovat jako čísla.

Například následující vstup stroje RAM



může být v případě Turingova stroje reprezentován jako



Turingův stroj simulující činnost stroje RAM

Turingův stroj simulující činnost stroje RAM bude mít několik pásek:

- Pásku, na které bude uložen obsah pracovní paměti stroje RAM.
- Tři pásky, na kterých budou uloženy hodnoty registrů A, B a C.
  (Hodnoty registrů A, B a C budou na těchto páskách zapsány binárně bez vedoucích nul a zleva a zprava budou ohraničeny symboly #.)
- Pásku reprezentující vstupní pásku stroje RAM.
- Pásku reprezentující výstupní pásku stroje RAM.
- Jednu pomocnou pásku používanou při implementaci simulace jednotlivých instrukcí.

Turingův stroj simulující činnost stroje RAM

Turingův stroj si bude v řídící jednotce pamatovat, která instrukce stroje RAM se právě provádí.

Provedení většiny instrukcí není složité:

- $A := c$

  zapíše jednotlivé bity konstanty c na pásku registru A

- $B := A$ nebo $C := A$

  zkopíruje obsah pásky registru A na pásku registru B nebo C

- **goto** ℓ

  změní se jen stav řídící jednotky Turingova stroje

- **if** $(A \; rel \; 0)$ **goto** ℓ, kde $rel \in \{=, \neq, \leq, \geq, <, >\}$

  snadno se otestuje obsah registru A a podle výsledku se změní stav řídící jednotky Turingova stroje

Turingův stroj simulující činnost stroje RAM

- $A :=$ READ $()$

  zkopírování hodnoty (ohraničené znaky "#") ze vstupní pásky na pásku registru A

- WRITE (A)

  zkopírování hodnoty registru A na výstupní pásku.

- **halt**

  výpočet se zastaví

Turingův stroj simulující činnost stroje RAM

Také aritmetické instrukce jsou poměrně jednoduché, i když o něco složitější než předchozí instrukce:

- $A := A \; op \; B$, kde $op \in \{+, -, *, /\}$

  Příslušnou operaci (např. sčítání nebo odčítání) provede Turingův stroj bit po bitu, výsledek je ukládán do registru A.

Poznámka: Násobení a dělení je možné realizovat pomocí série sčítání, odčítání a bitových posunů.

Při implementaci násobení a dělení může být potřeba použít pomocnou pásku k ukládání mezivýsledků.

Turingův stroj simulující činnost stroje RAM

Asi nejsložitější je realizace pracovní paměti stroje RAM.

Jednou z možností je pamatovat si jen obsah těch buněk, se kterými stroj RAM v průběhu své činnosti někdy pracoval.

Příklad: Stroj RAM zatím pracoval jen s buňkami 2, 3 a 6:

- Buňka 2 obsahuje hodnotu 11.
- Buňka 3 obsahuje hodnotu -1.
- Buňka 6 obsahuje hodnotu 2.

Obsah pásky Turingova stroje reprezentující buňky paměti stroje RAM bude následující:

$	#	1	0	:	1	0	1	1	#	1	1	:	-	1	#	1	1	0	:	1	0	#	$	

Turingův stroj simulující činnost stroje RAM

Instrukce load, tj. $A := [A]$:

- Turingův stroj bude hledat příslušnou adresu uloženou v registru A na pásce reprezentující obsah paměti stroje RAM.
  (Pokud ji nenajdeme, přidá ji na konec, s tím, že obsahuje hodnotu 0.)
- Příslušnou hodnotu zkopíruje na pásku registru A.

Turingův stroj simulující činnost stroje RAM

Instrukce store, tj. $[C] := A$:

- Podobně jako u instrukce load se najde příslušné místo na pásce reprezentující pracovní paměť, kde se nachází obsah buňky, jejíž adresa je v registru C.
- Zbytek pásky s obsahem paměti stroje RAM se zkopíruje na pomocnou pásku.
- Na příslušné místo se zkopíruje obsah pásky registru A.
- Zbytek pásky, který byl zkopírován na pomocnou pásku, se zkopíruje zpět (za nově zapsanou hodnotu).