

Príklad 1: Analyzujte zložitosť pre program RAM na výpočet faktoriálu čísla n ($n!$)

Riešenie:

a) algoritmus v PL jazyku:

```

begin
  read r1 ;
  r2 B 1 ;
  if r1 ≤ 1 then write 1;
  else begin
    for r3 B 1 step 1 until r1
      r2 B r2 * r3 ;
    write r2 ;
  end
end

```

b) program pre RAM

	READ 1	}	read r ₁ ;	
	LOAD 1		}	if r ₁ ≤ 1 then
	JGZ OK			
	LOAD =1			
	STORE 2			
	JMP KONIEC			
OK:	LOAD 1	}	else	
	STORE 2			
LOOP:	SUB =1			
	JZ KONIEC			
	STORE 1			
	MUL 2			
	STORE 2			
	LOAD 1	}	write r ₂	
	JMP LOOP			
KONIEC:	WRITE 2			
	HALT			

c) uniformná zložitosť :

$$T(n) = 7 * n + 9 = O(n) \quad (\text{časová})$$

$$S(n) = 3 = O(1) \quad (\text{priestorová})$$

d) logaritmická zložitosť :

$$V \text{ i - tom kroku : } l(i!) + l(i+1) = \lg(i!) + \lg(i+1) = \lg(i!(i+1))$$

$$= \lg((i+1)!) \quad \text{výsledok násobenia}$$

pre n krokov :

$$T(n) = \sum_{i=1}^{n-1} \lg((i+1)!) \cong \sum_{i=1}^n \lg(i!) = \lg(1!) + \lg(2!) + \dots + \lg(n!) \leq n * \lg(n!)$$

$$n * \lg(n!) \sim n * \lg\left(\left(\frac{n}{e}\right)^n\right) = n^2 * \lg\left(\frac{n}{e}\right) = O(n^2 \lg n)$$

$$S(n) = l(n!) \sim l\left(\left(\frac{n}{e}\right)^n\right) = \lg\left(\left(\frac{n}{e}\right)^n\right) = n * \lg\left(\frac{n}{e}\right) = O(n \lg n)$$

Príklad 2: Analyzujte zložitosť programu RAM, ktorý rozpoznáva jazyk $L = 1^n 2^{n^2} 0$.

Riešenie:

a) algoritmus v PL jazyku :

```
begin
  r5 B 0
  read r1 ;
  for r2 B 1 step 1 until r1
    begin
      read r3 ;
      if r3 != 1 then goto KONIEC
    end ;
  for r2 B 1 step 1 until r1
    begin
      for r3 B 1 step 1 until r1
        begin
          read r4 ;
          if r4 != 2 then goto KONIEC
        end
      end
    end
  read r4 ;
  if r4 = 0 then r5 B 1 ;
  KONIEC : write r5
end
```

b) program pre RAM :

```
      LOAD =0
      STORE 5
      READ 1
      JGZ VSTUP_OK
      JMP KONIEC
VSTUP_OK: LOAD =1
      STORE 2
LAB1:   READ 3
      LOAD 3
      SUB =1
      JZ DALSIA_1
      JMP KONIEC
DALSIA_1 : LOAD 2
      SUB 1
      JZ DVOJKY
      LOAD 2
      ADD =1
      STORE 2
      JMP LAB1
DVOJKY : LOAD = 1
      STORE 2
```

```

L1 :      LOAD = 1
        STORE 3

L2 :      READ  4
        LOAD  4
        SUB   = 2
        JZ DALSIA_2
        JMP KONIEC
DALSIA_2 : LOAD  3
        SUB   1
        JZ KONIEC_L2
        LOAD  3
        ADD   = 1
        STORE 3
        JMP  L2
KONIEC_L2: LOAD  2
        SUB   1
        JZ KONIEC_L1
        LOAD  2
        ADD   = 1
        STORE 2
        JMP  L1
KONIEC_L1: READ  4
        JZ NASTAV_1
        JMP KONIEC
NASTAV_1: LOAD = 1
        STORE 5
KONIEC:  WRITE 5
        HALT

```

c) uniformná časová zložitosť :

$$T_{L2}(n) = 11n$$

$$T_{L1}(n) = n * T_{L2}(n) + 9n = 11n^2 + 9n = O(n^2)$$

r_1	\mathbf{B}	13	
r_2	\mathbf{B}	r_1	(= 13)
r_3	\mathbf{B}	1	
r_4	\mathbf{B}	1	(načíta ďalší vstupný údaj)
r_3	\mathbf{B}	$r_3 * r_2$	($r_3 \mathbf{B} 13^1 * 2^{40}$)
r_2	\mathbf{B}	$r_2 * r_2$	($r_2 \mathbf{B} 13 * 13 = 13^2$)
r_4	\mathbf{B}	0	(načíta ďalší vstup)
r_2	\mathbf{B}	$r_2 * r_2$	($r_2 \mathbf{B} 13^2 * 13^2 = 13^4$)
r_4	\mathbf{B}	1	
r_3	\mathbf{B}	$r_3 * r_2$	($r_3 \mathbf{B} 13 * 13^4 = 13^5$)
r_2	\mathbf{B}	$r_2 * r_2$	($r_2 \mathbf{B} 13^4 * 13^4 = 13^8$)
r_4	\mathbf{B}	1	
r_3	\mathbf{B}	$r_3 * r_2$	($r_3 \mathbf{B} 13^5 * 13^8 = 13^{13}$)
r_2	\mathbf{B}	nie je podstatné	

c) uniformná časová zložitost :

počet kroků : $\lceil \log_2 13 \rceil + 1$

všeobecně : $\lceil \log_2 n \rceil + 1$

potom:

$$T(n) = \lceil \log_2 n \rceil + 1 = O(\log_2 n)$$

Príklad 4: Zostrojte program v PL aj RAM na výpočet súčtu $1+2 + \dots +n$ a analyzujte jeho zložitosť.

Riešenie:

a) algoritmus v PL :

```

begin
  read r1 ;
  r2 ← 0;
  for r3 ← 1 until r3 ≤ r1 step 1
    r3 ← r2 + r3 ;
  write r2
end .

```

b) program pre RAM :

```

      READ  1      read r1
      LOAD  =0     }   r2 ← 0
      STORE 2      }
      LOAD  =1     }   r3 ← 1
      STORE 3      }
OP:   LOAD  2      }
      ADD   3      }   r2 ← r2 + r3
      STORE 2      }
      LOAD  1      }
      SUB   3      }   until r3 ≤ r4
      JZ KONIEC   }
      LOAD  3      }
      ADD   =1     }   r3 ← r3 + 1
      STIRE 3      }
      JMP   OP
KONIEC: WRITE 2
      HALT

```

log. časová zložka

```

l(1) + l(n)
l(0)
l(2) + l(0)
l(1)
l(3) + l(1)
l(2) + l(∑k=1i-1 k)
l(2) + l(∑k=1i-1 k) + l(i) ←!
l(2) + l(∑k=1i k)
l(1) + l(n)
l(3) + l(n) + l(i)
1
l(3) + l(i)
l(1) + l(i)
l(3) + l(i+1)

```

c) logickoaritmetická časová zložitosť

ADDB v i-tom kroku :

$$\begin{aligned}
 l\left(\sum_{k=1}^{i-1} k\right) + l(i) &= l\left(\frac{i-1+1}{2} \cdot (i-1)\right) + l(i) = l\left(\frac{i^2-i}{2}\right) + l(i) = l\left(i \cdot \frac{i^2-1}{2}\right) = l\left(\frac{i^3-i^2}{2}\right) \cong l(i^3) = \\
 &= \log_2 i^3 = 3 \log_2 i
 \end{aligned}$$

pre n krokov :

$$\begin{aligned}
 T(n) &= \sum_{i=1}^n 3 \log_2 i = 3 \cdot \sum_{i=1}^n \log_2 i = 3 \log \prod_{i=1}^n i = 3 \log_2 (1 \cdot 2 \cdot \dots \cdot n) \leq 3 \log_2 (n^n) = \\
 &= 3n \log_2 n = O(n \log_2 n)
 \end{aligned}$$

d) priestorová logaritmickejšia zložitosť :

$$\begin{aligned}
 S(n) &= l(c(2)) = l\left(\sum_{i=1}^n i\right) = l(1+2+\dots+n) = l\left(\frac{n+1}{2} \cdot n\right) = l\left(\frac{n^2+n}{2}\right) \cong \log_2 n^2 = \\
 &= 2 \log_2 n = O(n \log_2 n)
 \end{aligned}$$

e) uniformná zložitost :

$$T(n) = 5 + n * 10 + 2 = 10 * n + 7 = O(n)$$

$$S(n) = 3 = O(1)$$

(3 registre)

Príklad 5: Napíšte program v PL aj v RAM na výpočet nájdenia maximálneho prvku vstupného reťazca prirodzených čísel dĺžky n . Analyzujte zložitosť.

Riešenie:

a) algoritmus v PL :

```

begin
  read r2 ;
  r3 ← r2;
  for r1 ← 1 until n
    begin
      read r2
      if c(r2) > r3 then r3 ← r2
    end
  write r3
end .

```

registre : r₁-počítadlo
 r₂ -vstup
 r₁- maximum

b) RAM program :

	READ 2	}	read r ₂	(a ₁)
	LOAD 2		r ₃ ← r ₂	
	STORE 3			
	LOAD =1	}	r ₁ ← 1	
	STORE 1			
A:	LOAD 1	}	r ₁ ← r ₁ + 1	
	ADD = 1			
	STORE 1			
	LOAD = n	}	if r ₁ = n then goto B	
	SUB 1			
	JZ B			
	READ 2	}	read r ₂	(a ₂)
	LOAD 3			
	SUB 2		if r ₃ > r ₃ then goto A	
	JGTZ A	}		
	LOAD 2			
	STORE 3		else r ₃ ← r ₂	
	JMP A			
B:	WRITE 3			
	HALT			

c) zložitosť

$$T(n) = 5 + 13(n - 1) = O(n); S(n) = 3$$

logaritmická časová :

SUB – významná inštrukcia

$$T(i) = 1(i) + 1\left(\sum_{k=1}^{i-1} k\right) = 1(i) + 1\left(\frac{i^2 - i}{2}\right) = \log_2\left(\frac{i^3 - i^2}{2}\right) \cong \log_2 i^3 = 3\log_2 i$$

$$T(n) = \sum_{i=1}^n \log_2(i) \leq n \log_2 n = O(n \log_2 n)$$

Príklad 7: Napíšte program v PL a RAM na výpočet súčtu $1 + 3 + \dots + (2n + 1)$, n je vstupný údaj. Analyzujte zložitosť.

Riešenie:

a) algoritmus v PL :

```

begin
  read r1;
  r2 ← 1;
  r3 ← 1;
  for r4 ← 2 step 1 to n
    begin
      r2 ← r2 + 2;
      r3 ← r3 + r2;
    end;
  write r3;
end.

```

registre :

r_1 - n
 r_2 - nasledujúci sčítanec
 r_3 - súčet
 r_4 - počítadlo

b) program pre RAM

	READ 1		read r ₁
	LOAD =1		
	STORE 2		r ₂ ← 1
	STORE 3		r ₃ ← 1
	STORE 4		r ₄ ← 1
LOOP :	LOAD 4	}	r ₄ ← r ₄ + 1
	ADD =1		
	STORE 4		
	SUB 1	}	if r ₄ = n goto KONIEC
	JZ KONIEC		
	LOAD 2	}	else r ₂ ← r ₂ + 2
	ADD =2		
	STORE 2		
	LOAD 3	}	r ₃ ← r ₃ + r ₂
	ADD 2		
	STORE 3		
	JMP LOOP		
KONIEC :	WRITE 3		
	HALT		

c) zložitosť :

uniformná :

$$T(n) = 7 + 12n = O(n)$$

$$S(n) = 4 = O(1)$$

logaritmickej :

i -ty krok :

$$T(i) = l(i) + l\left(\sum_{k=1}^{i-1} k\right) = l(i) + l\left(\frac{i^2 - i}{2}\right) =$$

$$T(n) = 3n \ln n = O(n \ln n) \quad = l\left(i \frac{i^2 - i}{2}\right) = l\left(\frac{i^3 - i^2}{2}\right) = l(i^3) = 3 \ln(i)$$

$$S(n) = 4 \ln n = O(\ln n)$$

Príklad 8: Analyzujte zložitosť lineárneho RAM programu na výpočet determinantu rozmerov $n \times n$.

Riešenie:

Lineárny RAM program :

$n = 2$

$$\begin{vmatrix} a_{11} & a_{12} \\ a_{21} & a_{22} \end{vmatrix}$$

$p_1 \leftarrow a_{11} * a_{22};$
 $p_2 \leftarrow a_{21} * a_{12};$
 $v \leftarrow p_1 - p_2;$

$n = 3$

$$\begin{vmatrix} a_{11} & a_{12} & a_{13} \\ a_{21} & a_{22} & a_{23} \\ a_{31} & a_{32} & a_{33} \end{vmatrix}$$

$D_{11}^2;$
 $D_{12}^2;$
 $D_{13}^2;$

$p_1 \leftarrow a_{11} * D_{11}^2;$
 $p_2 \leftarrow a_{12} * D_{12}^2;$
 $p_3 \leftarrow a_{13} * D_{13}^2;$
 $v \leftarrow p_1 - p_2;$
 $v \leftarrow v + p_3;$

$n = 4$

zložitosť $D^2 = 3$

zložitosť $D^3 = 3 * D^2 + 3 +$
 $2 = 3 * D^2 + 2 * 3 - 1 =$
 $3(D^2 + 2) - 1$

$$\begin{vmatrix} a_{11} & a_{12} & a_{13} & a_{14} \\ a_{21} & a_{22} & a_{23} & a_{24} \\ a_{31} & a_{32} & a_{33} & a_{34} \\ a_{41} & a_{42} & a_{43} & a_{44} \end{vmatrix}$$

$D_{11}^3;$
 $D_{12}^3;$
 $D_{13}^3;$
 $D_{14}^3;$

$p_1 \leftarrow a_{11} * D_{11}^3;$
 $p_2 \leftarrow a_{12} * D_{12}^3;$
 $p_3 \leftarrow a_{13} * D_{13}^3;$
 $p_4 \leftarrow a_{14} * D_{14}^3;$
 $v \leftarrow p_1 - p_2;$
 $v \leftarrow v + p_3;$
 $v \leftarrow v - p_4;$

zložitosť $D^4 = 4 * D^3 + 4 +$
 $3 = 4 * D^3 + 2 * 4 - 1 = 4(D^3$
 $+ 2) - 1$

obecne pre zložitosť výpočtu determinantu $n \times n$ platí :

$$D^n = \begin{cases} 3, & \text{ak } n = 2 \\ n(D^{n-1} + 2) - 1, & \text{ak } n > 2 \end{cases}$$

Výpočet D^n :

$$\begin{aligned}
 D^n &= n(D^{n-1} + 2) - 1 = n((n-1)(D^{n-2} + 2) - 1 + 2) - 1 = \\
 &= n((n-1)(D^{n-2} + 2) + 1) - 1 = n((n-1)((n-2)(D^{n-3} + 2) - 1) + 1) - 1 = \\
 &\bullet \\
 &\bullet \\
 &\bullet \\
 &= n((n-1)((n-2)(\dots 3(D^2 + 2) - 1) + 1) \dots + 1) - 1 = \\
 &= -1 + n + n(n-1)((n-2)(\dots 3(3+2) + 1) \dots) + 1 =
 \end{aligned}$$

$$= -1 + n + (n-1)n + \dots + n(n-1) + \dots + 4 + n(n-1)(n-2) \dots 4.3.(3+2) =$$

$$= -1 + n! \sum_{i=4}^n \frac{i}{i!} + \frac{5n!}{2}$$

$$n! \sum_{i=4}^n \frac{i}{i!} = \sum_{i=1}^n \frac{i}{i!} - \frac{1}{1!} - \frac{2}{2!} - \frac{3}{3!} = \sum_{i=1}^n \frac{i}{i!} - \frac{5}{2}$$

$$\lim_{n \rightarrow \infty} \frac{i}{i!} = e$$

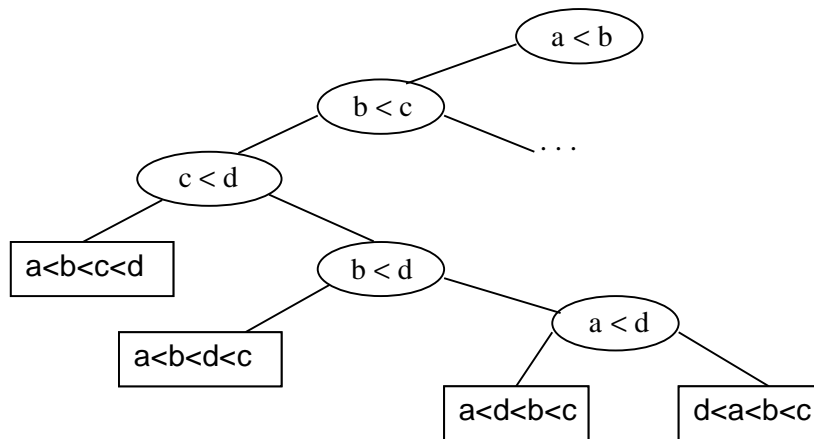
Potom :

$$D^n = -1 + n! \left(e - \frac{5}{2} \right) + n! \frac{5}{2} = e.n! - 1$$

$$T(n) = O(n!)$$

Príklad 9: Zostrojte rozhodovací strom pre usporiadanie 4 čísel a, b, c, d.

Riešenie:



Zložitosť rozhodovacieho stromu = výška stromu.

Obvykle meriame maximum porovnaní potrebných na to, aby sme sa dostali z koreňa na list, čo je viac ako výška stromu. Teda :

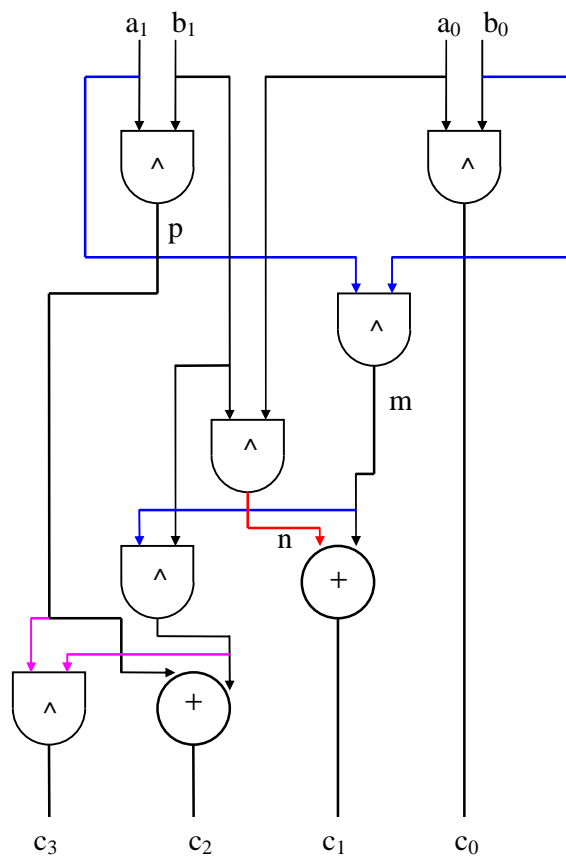
$$\begin{aligned} O_c(n) &= n! && \text{minimálny počet listov stromu} \\ T(n) &= O(n \cdot \ln n) \\ \text{pre } n &= 4 \\ T(4) &= O(4 \cdot \ln 4) \end{aligned}$$

Príklad 10: Napíšte binárny program a logickú schému pre násobenie dvoch dvojbitových čísel $[a_1a_0]$, $[b_1b_0]$.

Vieme, že výsledok bude maximálne 4 bitový, t.j. : $[a_1a_0][b_1b_0] = [c_3c_2c_1c_0]$

$$\begin{array}{lcl}
 c_0 \leftarrow a_0 \wedge b_0 & & \begin{array}{r} \underline{a_1} \underline{a_0} \cdot b_1 b_0 \\ a_1 b_0 \ a_0 b_0 \\ + \underline{a_1} \underline{b_1} \underline{a_0} \underline{b_1} \underline{\hspace{1cm}} \\ a_1 b_1 [a_0 b_1 + a_1 b_0] a_0 b_0 \end{array} \\
 m \leftarrow a_1 \wedge b_0 & & \underbrace{\hspace{1cm}} \underbrace{\hspace{1cm}} \underbrace{\hspace{1cm}} \underbrace{\hspace{1cm}} \\
 n \leftarrow a_0 \wedge b_1 & & c_3 \quad c_2 \quad c_1 \quad c_0 \\
 c_1 \leftarrow m \oplus n & & \downarrow \\
 p \leftarrow a_1 \wedge b_1 & & \text{(ak je prechod cez 10)} \\
 c_2 \leftarrow (m \wedge n) \oplus p & & \\
 c_3 \leftarrow (m \wedge n) \wedge p & &
 \end{array}$$

Logická schéma :



Zložitosť: $O_B(n)$ krokov

Príklad 11: Simulujte viacpáskovým Turingovým strojom RAM inštrukciu ADD *20.

Obece je RAM reprezentovaný Turingovým strojom nasledovne :

Páska č.1 :

#	#	i_1	#	c_1	#	#	i_2	#	c_1	#	#	..	i_k	#	c_k	#	#	b
---	---	-------	---	-------	---	---	-------	---	-------	---	---	----	-------	---	-------	---	---	---

Pričom :

i_1 je register 1

c_1 je obsah registra 1

páska č. 2 obsahuje akumulátor

páska č.3 je pracovná

Postup :

1. Na páske č.1 TS hľadáme register 20 , t.j. postupnosť ##10110#. Ak nájdeme, umiestnime c_{20} , ktoré nasleduje, na pásku č.3 TS. Ak nenájdeme i_{20} , koniec, pretože obsah je žiadny a nepriama adresácia nemôže byť vykonaná. Nech $c_{20} = 48$
2. Na páske č.1 hľadáme register č. 48, t.j. i_{48} . Ak nájdeme, umiestnime c_{48} na pásku č. 3 TS. Ak nenájdeme, umiestnime 0 na pásku č. 3 TS.
3. Pričítame číslo z pásky č. 3, c_{48} ku obsahu akumulátora, ktorý je umiestnený na páske č.2.

Príklad 12: Simulujte viacpáskový Turingovým strojom RAM inštrukciu STORE 30.

Postup :

1. Na páske č.1 hľadáme i_{30} , t.j. $##11110\#$.
2. Ak nájdeme, preskočíme obsah i_{20} , c_{30} a všetko ostatné napravo skopírujeme na pásku č.3 TS. Vrátime sa na políčko c_{30} a z pásky č.2, kde je uchovaný akumulátor skopírujeme jeho obsah do c_{30} . Potom z pásky č. 3 skopírujeme zvyšok naspäť.
3. Ak nenájdeme i_{30} , vytlačíme $11110\#$ a zatým obsah akumulátora ukončený $##$.

Zložitosť :

RAM výpočet s logaritmickou zložitou k , vyžaduje maximálne $O(k^2)$ krokov TS.

Príklad 13: Simulujte RAM ADD *i na RASP stroji. Nech n je počet inštrukcií programu.

Postup :

1. Dočasne uložíme obsah akumulátora do r_1 .
2. Vložíme obsah registra do r_{i+1} do akumulátora.
3. Pričítame n ku akumulátoru.
4. uchováme súčet v adresnom poli inštrukcie ADD.
5. obnovíme akumulátor z r_1 .
6. Použijeme inštrukciu ADD z kroku 4.

Ako to vyzerá : ADD *i

	RASP reg.	obsah	význam
napr. 100		STORE	} STRE 1 ,krok 1 .po
101		1	
102		LOAD	} krok 2.
103		u+i	
104		ADD	} krok 3.
105		=n	
106		STORE	} krok 4.,ešte nevieme adresné pole ADD,
107		111	
108		LOAD	} krok 5.
109		1	
110		ADD	} krok 6., b je obsah i –teho registra RAM teraz
111		b	

Zložitosť : Každá RAM inštrukcia vyžaduje maximálne 6 RASP inštrukcií.

$$T_{RASP}(n)=6.T_{RAM}(n)$$

Príklad 14: Simulácia RASP –RAM. Simulujte na RAM stroji RASP inštrukciu ADD i.

V RAM si rezervujeme nasledujúce registre :

r_1 – pre nepriamu adresáciu

r_2 – pre počítadlo inštrukcií RASP

r_3 – pre uchovanie obsahu akumulátora RASP

ADD	RASP počet inš.		Pretože sme rezervovali 3 registre,
-----			PASP register i sa uchová v RAM
i			registri i+3

RAM program je cyklus modelovania :

a : LOAD 2 }
 ADD =1 } zvýši o jeden počítadlo inštrukcií, aby sme sa dostali
 STORE 2 } operand ADD.
 LOAD *2 }
 ADD =3 } vložíme operand i do akumulátora, pridáme konštantu 3 a
 STORE 1 } uchováme v r_4 .
 LOAD 3 }
 ADD *1 } vložíme obsah RASP akumulátora, pričítame obsah registra
 STORE 3 } i+3 a výsledok uchováme v r_3 .
 LOAD 2 }
 ADD =1 } zvýšenie poč. inštrukcií v 1 , t.j. ukazuje na ďalšie RASP
 STORE 2 } inštrukcie
 JMP a

Zložitosť :Každá RASP inštrukcia vyžaduje maximálne 13 inštrukcií RAM, teda

$$T_{RAM}(n)=13.T_{RASP}(n)$$