

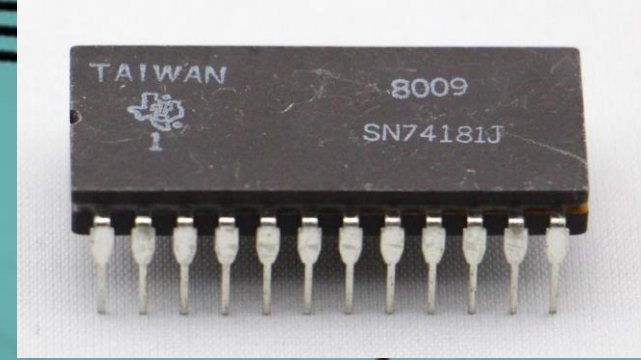


# Mikroprocesszorok építőelemei

# Tartalomjegyzék

Témakör	Főbb tartalmak	
ALU (Aritmetikai Logikai Egység)	Az ALU-hoz szükséges logikai kapuk ismertetése, Az alkalmazott logikai kapuk igazságtáblázata, Félösszeadó, Teljes összeadó, Kivonó, Programozható összeadó/kivonó, Egy gyári ALU (74181) ismertetése, Olvasható memóriás megvalósítás.	

# ALU



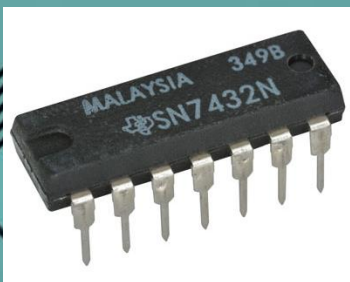
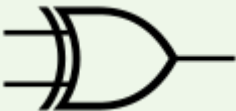
- ALU (Arithmetic Logic Unit) azaz aritmetikai-logikai egység. Feladata a számítások elvégzése. Az ALU nem más, mint egy kombinációs hálózat amely a bemenetére adott bitkombinációkra egyértelmű kimenetet ad. Ennek megfelelően felírható az ALU igazságtáblázata, ami alapján megfelelő tervezési módszerekkel logikai kapukból megalkotható.
- Az ALU egy olyan kombinációs hálózat, amelynek van:
  - két adat bemenete (operandusok),
  - egy működést befolyásoló vezérlő bemenete (ez határozza meg az ALU átviteli függvényét, például összead, AND, OR, XOR és egyéb kapcsolatokat valósít meg a két operandus között),
  - Egy túlcsordulás bemenet az előző számítást végző egység felől,
  - Egy eredmény kimenete (ezen jelenik meg az operandusokon elvégzett művelet eredménye),
  - Egy vagy több állapot kimenete (például az eredmény nulla, vagy a művelet során alul- illetve túlcsordulás keletkezett).
- Az ALU megvalósítható logikai kapuk segítségével de létrehozható egy, az igazságtáblázatot tartalmazó memóriával (ROM, PROM, EPROM) is.



# ALU (logikai kapuk)


- XOR (Kizáró VAGY) kapu:

Bemenet		Kimenet
A	B	A XOR B
0	0	0
0	1	1
1	0	1
1	1	0




- OR (VAGY) kapu:

Bemenet		Kimenet
A	B	A OR B
0	0	0
0	1	1
1	0	1
1	1	1



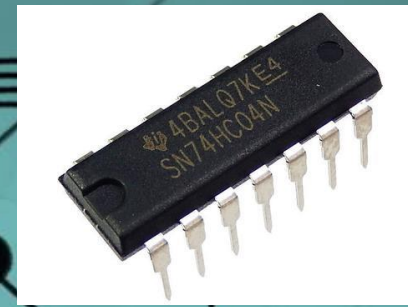
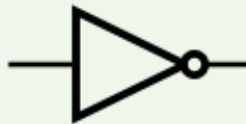
- AND (ÉS) kapu:

Bemenet		Kimenet
A	B	A AND B
0	0	0
0	1	0
1	0	0
1	1	1

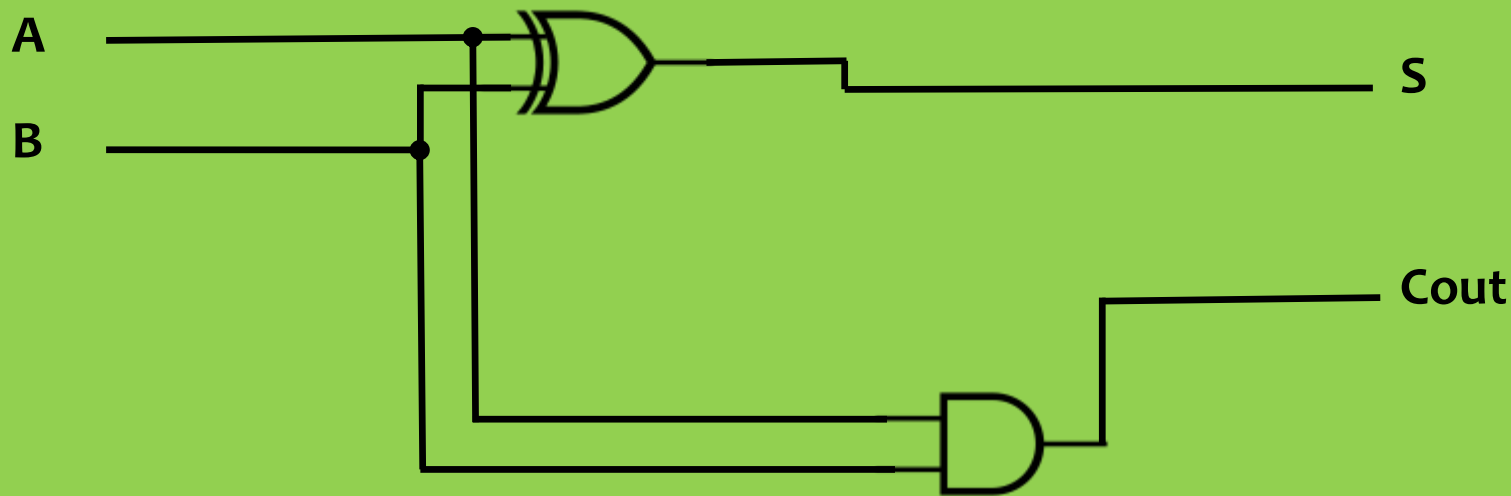


- NOT (Inverter) kapu:

Bemenet	Kimenet
A	NOT A
0	1
1	0



# ALU (1 bites félösszeadó)

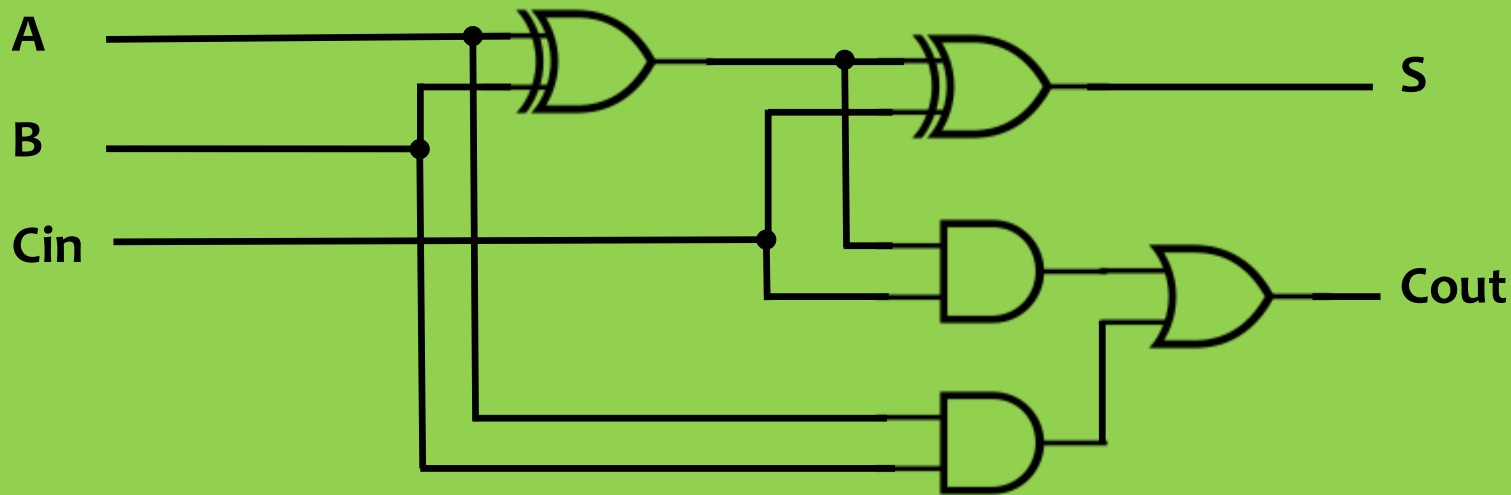


Bemenet		Kimenet	
A	B	A plus B (S)	Cout
0	0	0	0
0	1	1	0
1	0	1	0
1	1	0	1

- $S = A \text{ XOR } B$
- $\text{Cout} = A \text{ AND } B$

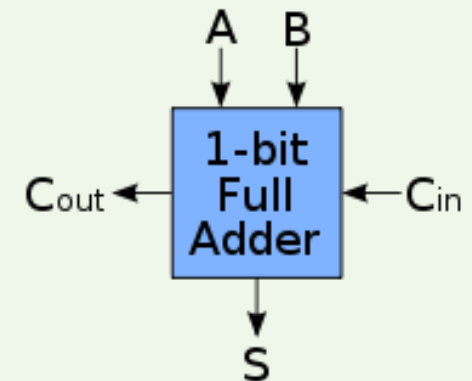
- Ez az egység két egy bites bemenet összegét és a keletkező túlcsondulást állítja elő, de nem képes kezelni a bejövő túlcsondulást (Carry input, azaz Cin).
- Két félösszeadóból és egy további VAGY kapuból létre lehet hozni egy teljes összeadó áramkört.

# ALU (1 bites teljes összeadó)

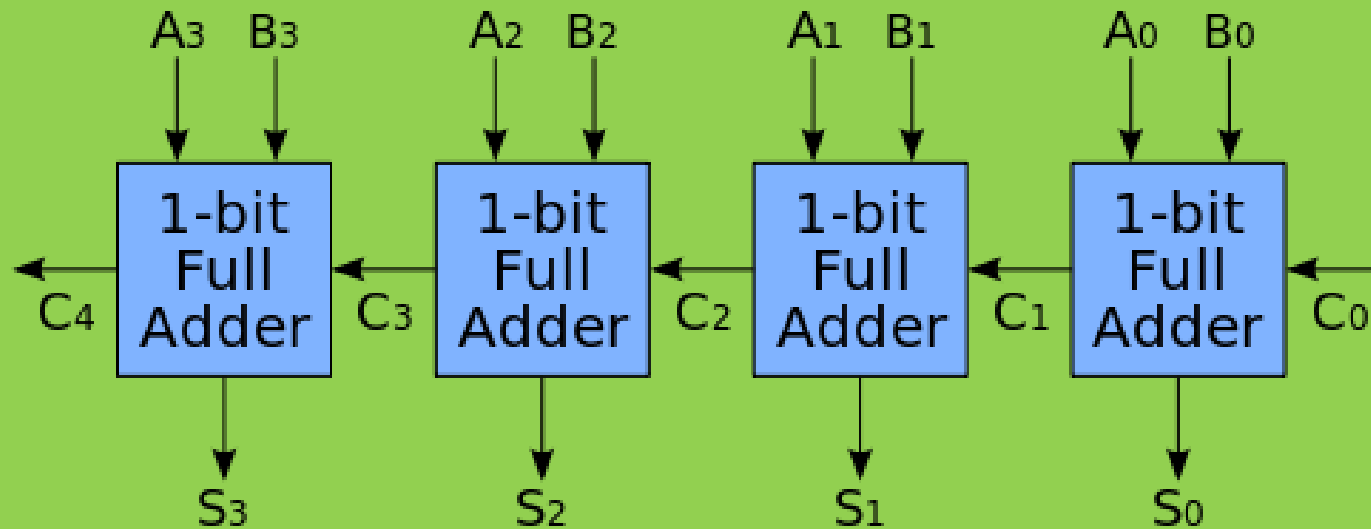


- $S = (A \text{ XOR } B) \text{ XOR } \text{Cin}$
- $\text{Cout} = (A \text{ AND } B) \text{ OR } (\text{Cin AND } (A \text{ XOR } B))$

Bemenet			Kimenet	
A	B	Cin	A plus B (S)	Cout
0	0	0	0	0
0	1	0	1	0
1	0	0	1	0
1	1	0	0	1
0	0	1	1	0
0	1	1	0	1
1	0	1	0	1
1	1	1	1	1



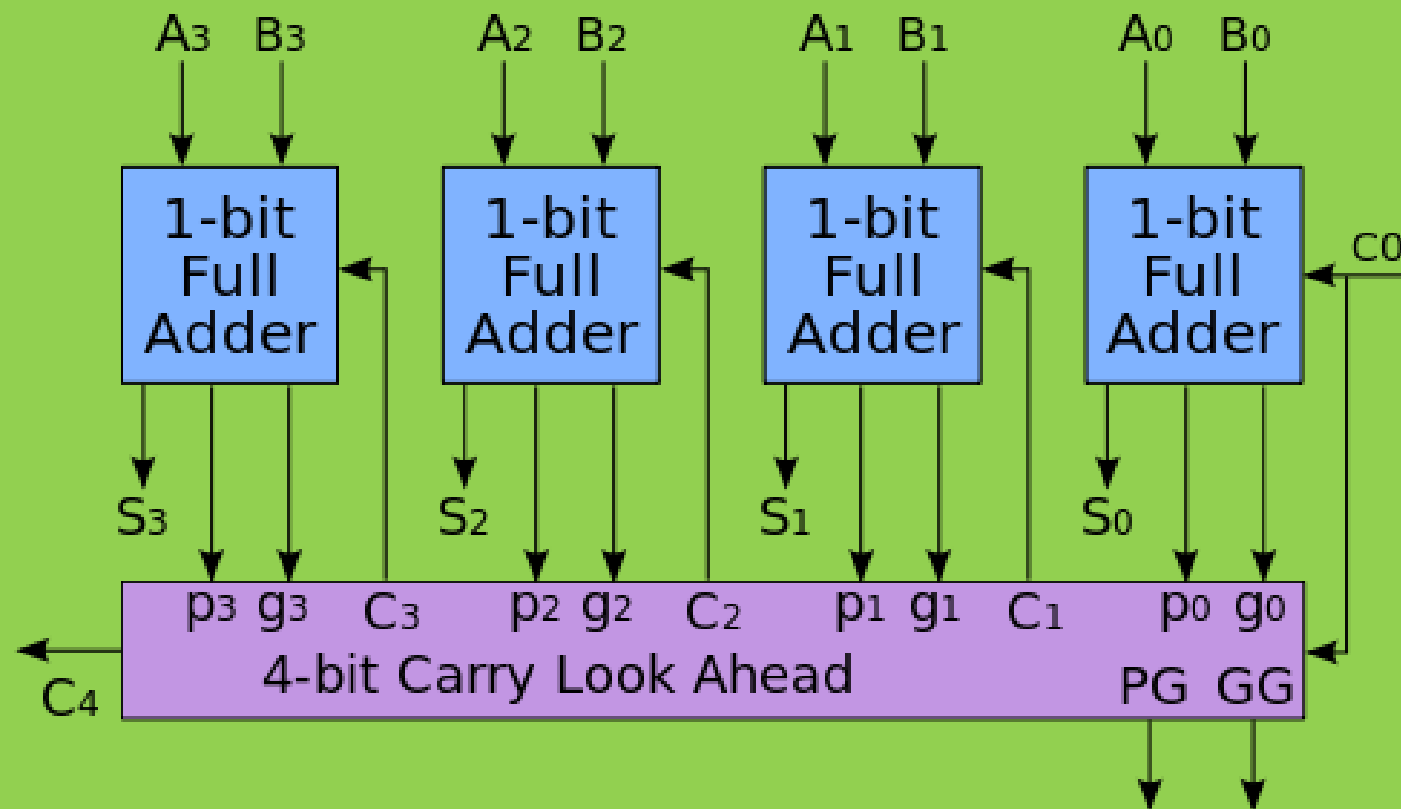
# ALU (4 bites teljes összeadó)



- A teljes összeadók sorba kapcsolásával elvileg tetszőleges bitszélességű összeadó áramkör létrehozható.
- Mivel a túlcserdülés sorosan terjed végig az összeadókon, minél több bites összeadót készítünk, annál lassabb a rendszer működése.



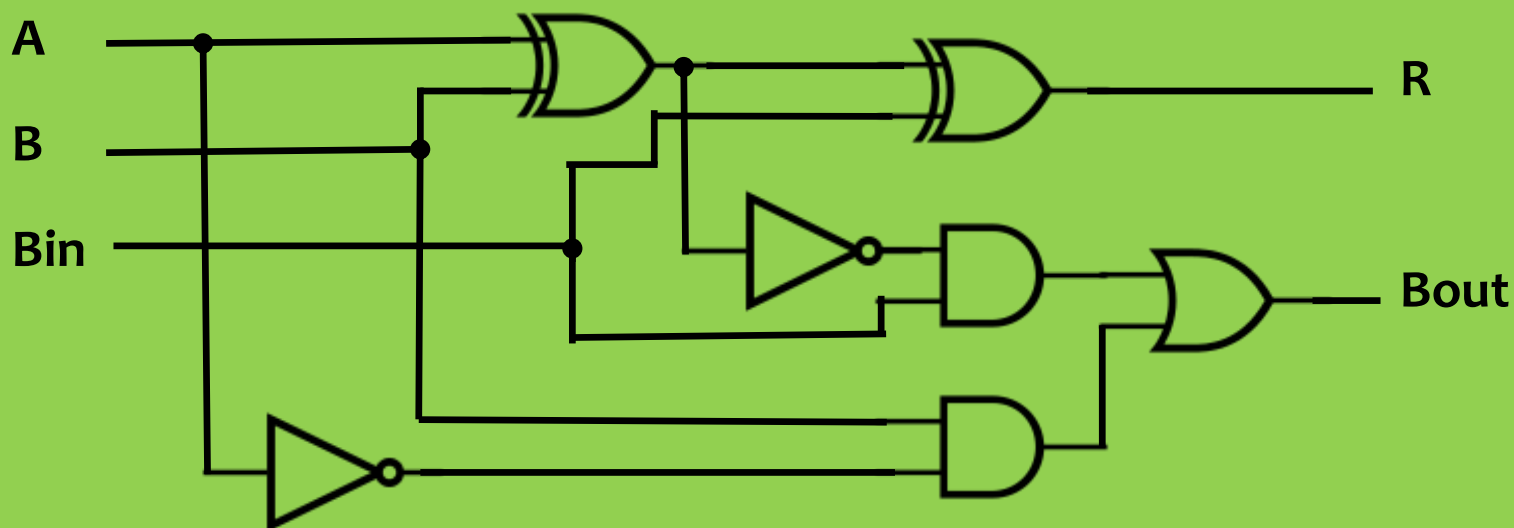
# ALU (4 bites teljes összeadó)



- A Carry Look Ahead áramkör segítségével gyorsítható a túlcserdülés terjedése. Ennek alapja, hogy az adott egységeknél keletkező túlcserdülés az adott egységek bemenetétől függ, azaz egymástól függetlenül, párhuzamosan is meghatározhatók.



# ALU (1 bites kivonás)



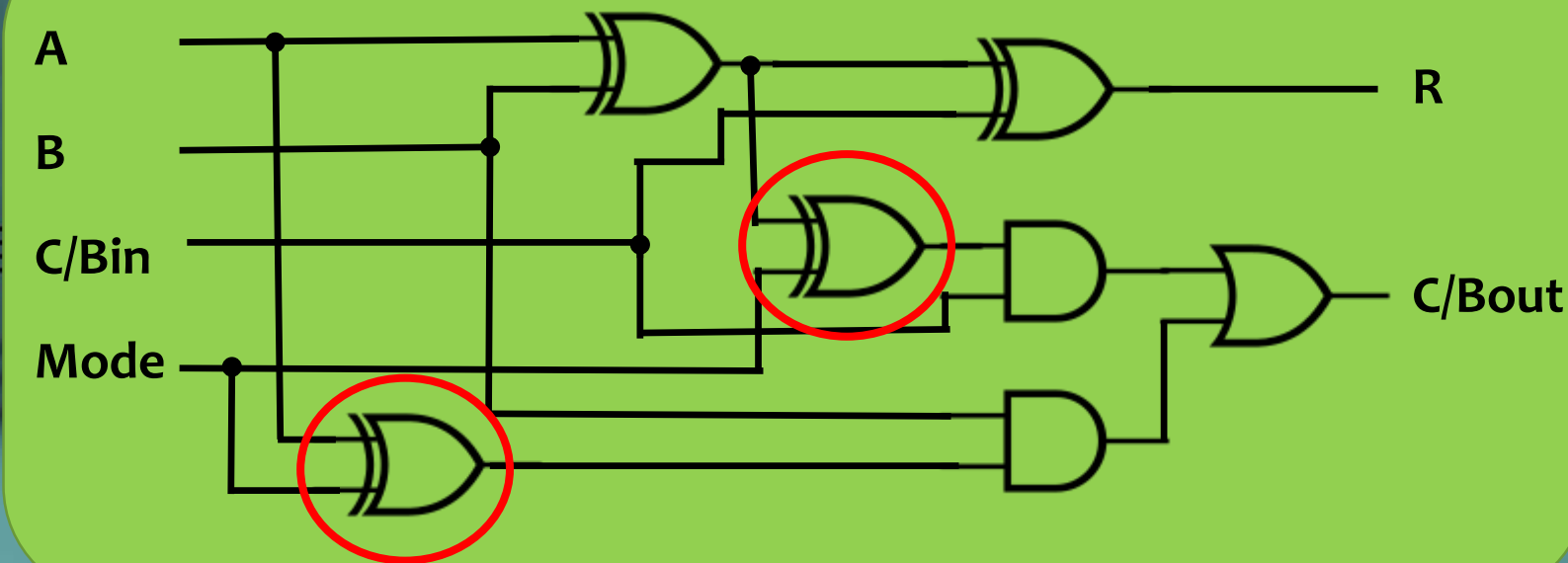
Bemenet			Kimenet	
A	B	Cin	A minus B (R)	Bout
0	0	0	0	0
0	1	0	1	1
1	0	0	1	0
1	1	0	0	0
0	0	1	1	1
0	1	1	0	1
1	0	1	0	0
1	1	1	1	1

- $R = (A \text{ XOR } B) \text{ XOR } \text{Cin}$
- $\text{Cout} = ((\text{NOT } A) \text{ AND } B) \text{ OR } (\text{Cin AND (NOT(A XOR B))})$

- Két inverter segítségével az összeadó áramkör kivonó áramkörre alakítható.

- Kivonás esetében áthozat (Borrow, azaz B) bitet alkalmazunk.

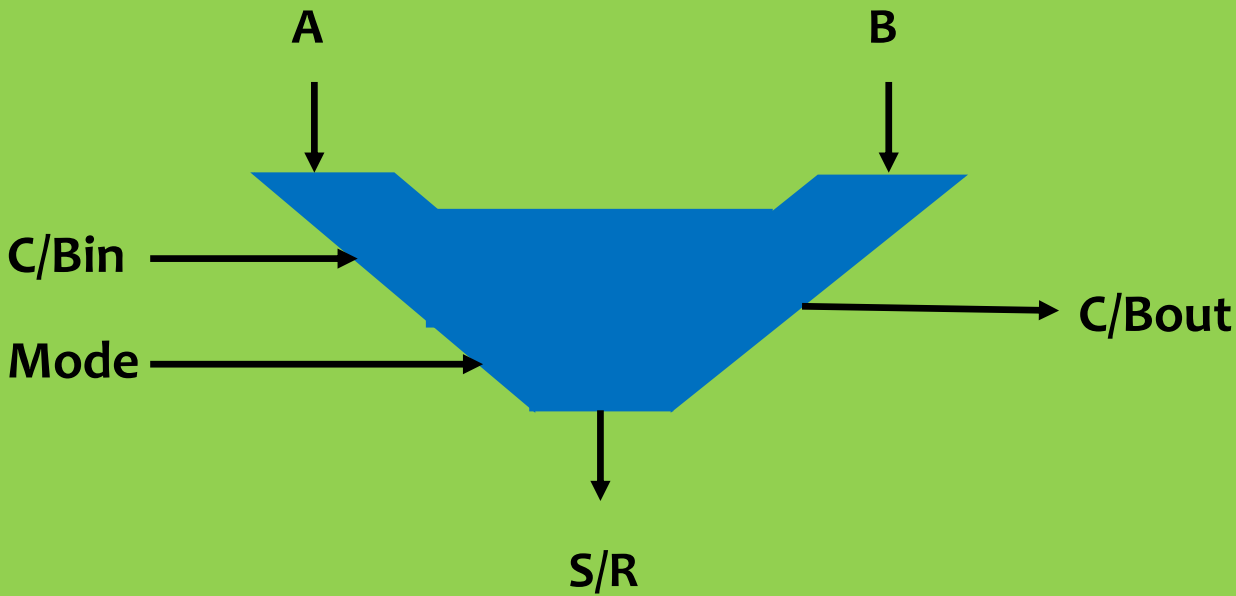
# ALU (1 bites összeadó/kivonó)



Bemenet				Kimenet			
A	B	Cin	Mode	A plus B (S)	Cout	A miús B (R)	Bout
0	0	0	0	0	0	-	-
1	0	1	0	1	0	-	-
0	0	0	0	1	0	-	-
1	0	1	0	0	1	-	-
0	1	0	0	1	0	-	-
1	1	1	0	0	1	-	-
0	1	0	0	0	1	-	-
1	1	1	0	1	1	-	-
0	0	0	1	-	-	0	0
0	1	0	1	-	-	1	1
1	0	0	1	-	-	1	0
1	1	0	1	-	-	0	0
0	0	1	1	-	-	1	1
0	1	1	1	-	-	0	1
1	0	1	1	-	-	0	0
1	1	1	1	-	-	1	1

- A „Mode” bemenet függvényében az áramkör összead (Mode=0) vagy kivon (Mode=1)
- A XOR kapukat mint programozható invertereket alkalmazunk.
- Mode = 0 esetén az A és az A XOR B jelek változatlanul kapcsolódnak a további kapukhoz, a Mode = 1 esetén pedig azok negáltjai haladnak tovább.

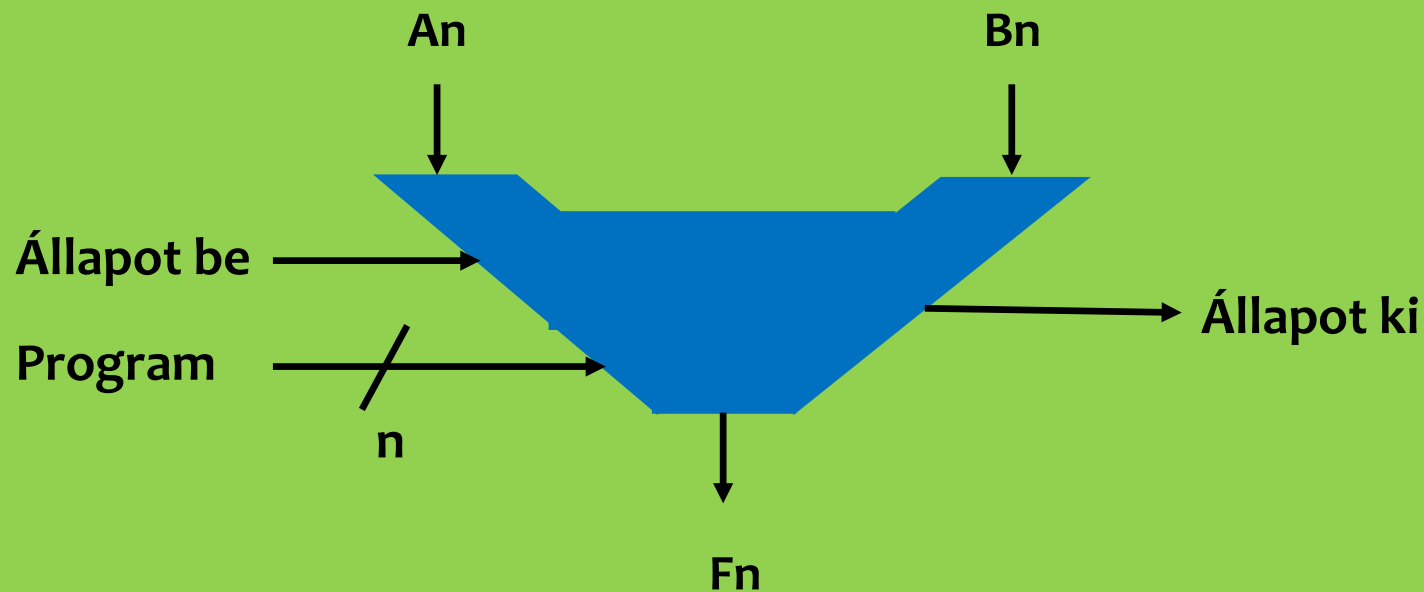
# ALU (1 bites összeadó/kivonó)



- Az összeadó/kivonó áramkör elhagyva a logikai kapuk jelölését a fentiek szerint is ábrázolható.

Bemenet				Kimenet			
A	B	Cin	Mode	A plus B (S)	Cout	A miús B (R)	Bout
0	0	0	0	0	0	-	-
1	0	1	0	1	0	-	-
0	0	0	0	1	0	-	-
1	0	1	0	0	1	-	-
0	1	0	0	1	0	-	-
1	1	1	0	0	1	-	-
0	1	0	0	0	1	-	-
1	1	1	0	1	1	-	-
0	0	0	1	-	-	0	0
0	1	0	1	-	-	1	1
1	0	0	1	-	-	1	0
1	1	0	1	-	-	0	0
0	0	1	1	-	-	1	1
0	1	1	1	-	-	0	1
1	0	1	1	-	-	0	0
1	1	1	1	-	-	1	1

# ALU

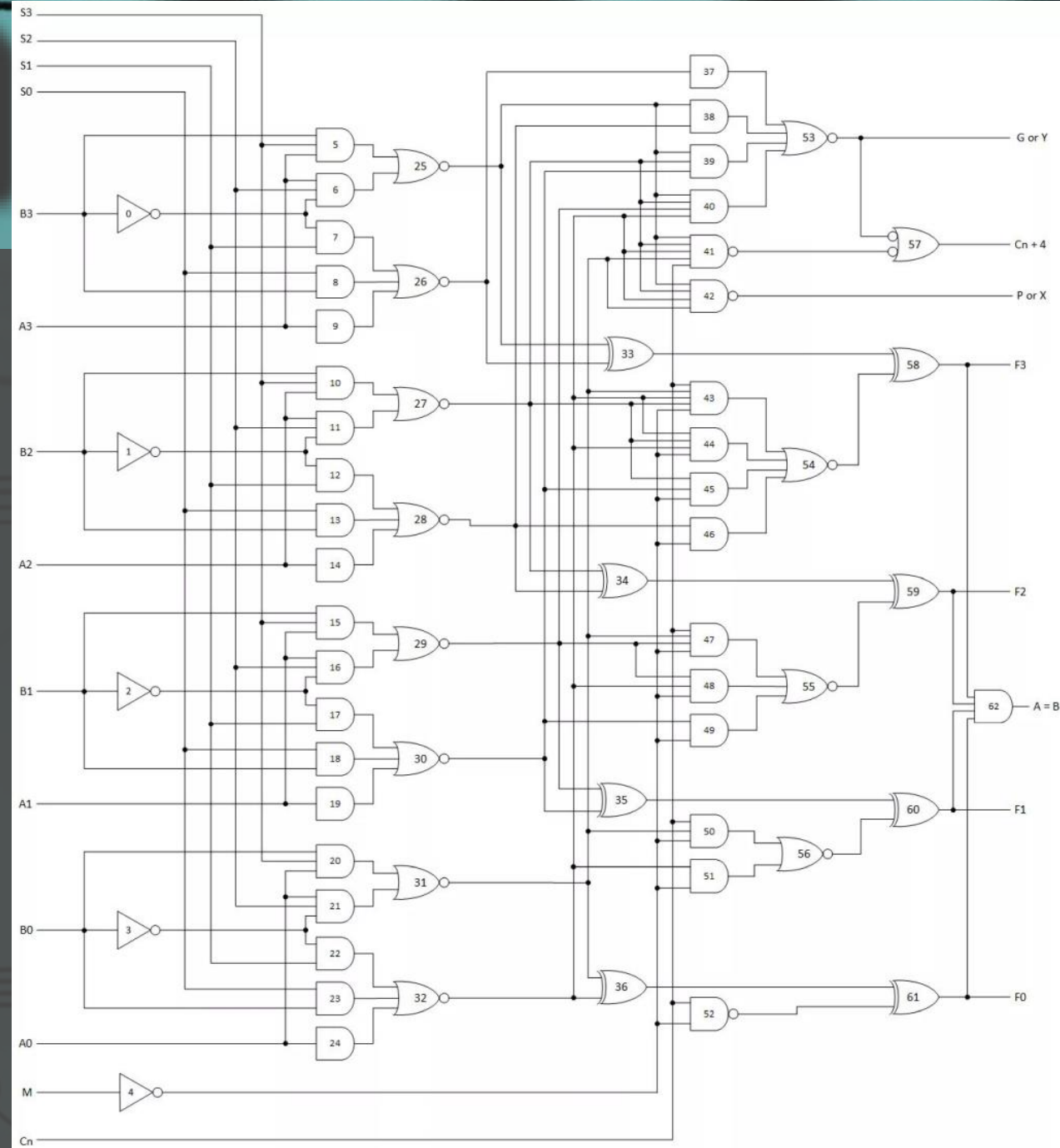


- A z  $A_n$  és  $B_n$  az  $n$  bites operandusokat jelöli amin az ALU elvégzi a kijelölt műveletet.
- Az állapot be- és kimenet jelenti a külső egységtől származó, illetve a művelet során keletkezett túlsordulást (Carry) vagy áthozatot (Borrow).
- Az  $F_n$  az  $n$  bites kimenet az elvégzett művelet eredményét jelöli.
- A Program egy több ( $n$ ) bites bemenet. Ezen keresztül lehet az ALU átviteli függvényét (például AND, OR, XOR, összeadás, stb.) meghatározni.



# ALU (74181)

- A Texas Instruments cég által gyártott SN74181 egy egyetlen integrált áramkörben megvalósított 4 bites ALU.
- A B<sub>0</sub>-B<sub>3</sub> és A<sub>0</sub>-A<sub>3</sub> bemenetek az operandusok.
- Az S<sub>0</sub>-S<sub>3</sub> és M bemenetek a műveleteket meghatározó vezérlő jelek (részletes magyarázat a következő dián).
- C<sub>n</sub> a bemenő Carry.
- F<sub>0</sub>-F<sub>3</sub> az eredmény kimenetek.
- C<sub>n+4</sub> a túlcscordulás kimenet.
- G és P kimenetek a gyorsabb átvitelképzéshez használatos átvitel jelzők (Carry Look Ahead áramkör számára).



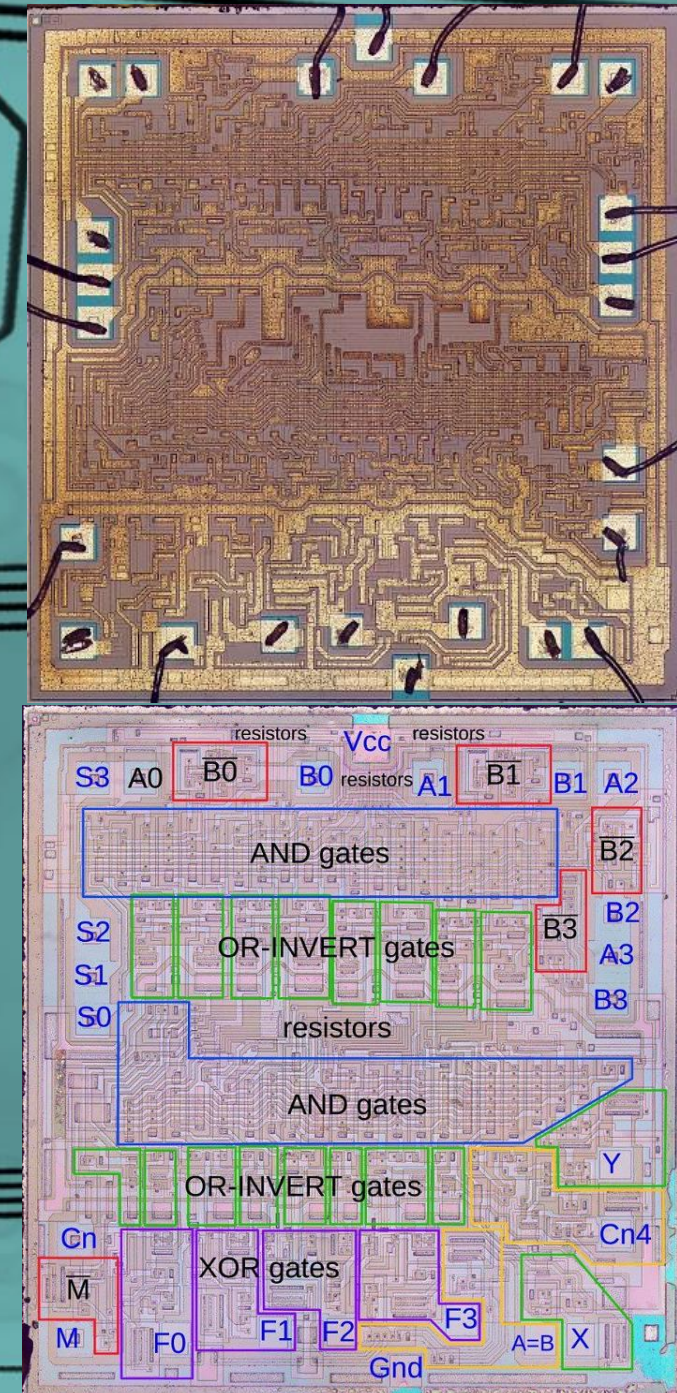


# ALU (74181)

- Az ALU által megvalósított átviteli függvények:

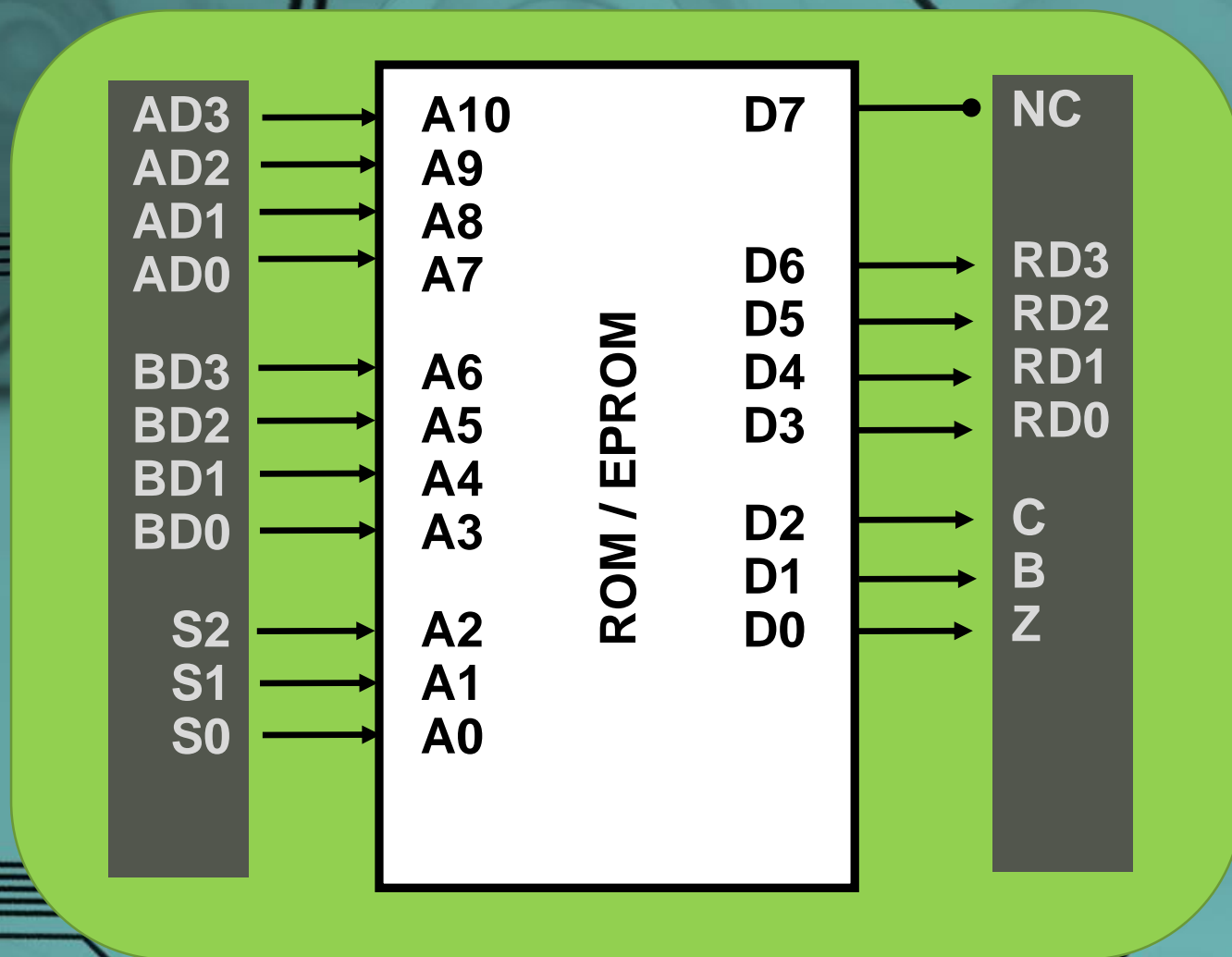


SELECTION				ACTIVE-HIGH DATA		
				M = H LOGIC FUNCTIONS	M = L; ARITHMETIC OPERATIONS	
S3	S2	S1	S0		$\overline{C_n} = H$ (no carry)	$\overline{C_n} = L$ (with carry)
L	L	L	L	$F = \overline{A}$	$F = A$	$F = A \text{ PLUS } 1$
L	L	L	H	$F = \overline{A + B}$	$F = A + B$	$F = (A + B) \text{ PLUS } 1$
L	L	H	L	$F = \overline{AB}$	$F = A + \overline{B}$	$F = (A + \overline{B}) \text{ PLUS } 1$
L	L	H	H	$F = 0$	$F = \text{MINUS } 1 \text{ (2's COMPL)}$	$F = \text{ZERO}$
L	H	L	L	$F = \overline{AB}$	$F = A \text{ PLUS } \overline{AB}$	$F = A \text{ PLUS } \overline{AB} \text{ PLUS } 1$
L	H	L	H	$F = \overline{B}$	$F = (A + B) \text{ PLUS } \overline{AB}$	$F = (A + B) \text{ PLUS } \overline{AB} \text{ PLUS } 1$
L	H	H	L	$F = A \oplus B$	$F = A \text{ MINUS } B \text{ MINUS } 1$	$F = A \text{ MINUS } B$
L	H	H	H	$F = \overline{AB}$	$F = \overline{AB} \text{ MINUS } 1$	$F = \overline{AB}$
H	L	L	L	$F = \overline{A} + B$	$F = A \text{ PLUS } AB$	$F = A \text{ PLUS } AB \text{ PLUS } 1$
H	L	L	H	$F = A \oplus B$	$F = A \text{ PLUS } B$	$F = A \text{ PLUS } B \text{ PLUS } 1$
H	L	H	L	$F = B$	$F = (A + \overline{B}) \text{ PLUS } AB$	$F = (A + \overline{B}) \text{ PLUS } AB \text{ PLUS } 1$
H	L	H	H	$F = AB$	$F = AB \text{ MINUS } 1$	$F = AB$
H	H	L	L	$F = 1$	$F = A \text{ PLUS } A$	$F = A \text{ PLUS } A \text{ PLUS } 1$
H	H	L	H	$F = A + \overline{B}$	$F = (A + B) \text{ PLUS } A$	$F = (A + B) \text{ PLUS } A \text{ PLUS } 1$
H	H	H	L	$F = A + B$	$F = (A + \overline{B}) \text{ PLUS } A$	$F = (A + \overline{B}) \text{ PLUS } A \text{ PLUS } 1$
H	H	H	H	$F = A$	$F = A \text{ MINUS } 1$	$F = A$





# ALU (megvalósítás EPROM-mal)



- Az EPROM vagy ROM címvezetékei jelentik az ALU bemeneteit.
- Az EPROM vagy ROM adat kimenetei jelentik az ALU kimenetét.
- Minden címkombinációhoz tartozik egy adatkombináció.  
Például:  
(S<sub>2</sub>=0; S<sub>1</sub>=0; S<sub>0</sub>=0 a két operandus összeadását (A plusz B) jelöli)  
Cím: Adat:  
0011 0001 000 → x 0100 000  
  
(S<sub>2</sub>=0; S<sub>1</sub>=0; S<sub>0</sub>=0 a két operandus összeadását (A plusz B) jelöli)  
Cím: Adat:  
1111 0001 000 → x 0000 100  
  
(S<sub>2</sub>=0; S<sub>1</sub>=0; S<sub>0</sub>=1 a két operandus kivonását (A mínusz B) jelöli)  
Cím: Adat:  
0011 0001 000 → x 0010 000  
  
(S<sub>2</sub>=0; S<sub>1</sub>=0; S<sub>0</sub>=1 a két operandus kivonását (A mínusz B) jelöli)  
Cím: Adat:  
0011 0011 000 → x 0000 001