LO Test Guide
Verze.0.9.3
19/5/2014

Kolektivní práce skupiny VŠB-TUO FEI Bc. 2013-2016

2014

Užitečné odkazy:

- http://www.converter.cz/baster/baster.php
  - (Český převodník soustav, neumí však desetiny)
- http://www.easysurf.cc/cnver17.htm
  - (Převody soustav včetně desetin!)
- http://www.miniwebtool.com/binary-calculator/
  - (Binární kalkulačka)
- http://www.miniwebtool.com/decimal-to-bcd-converter/
  - (BDC Converter)
- http://www.ee.calpoly.edu/media/uploads/resources/KarnaughExplorer_1.html
  - (Minimalizace kanaugh mapy)

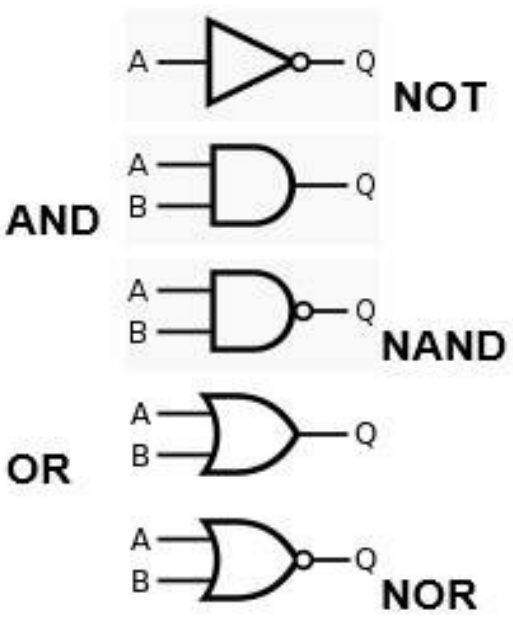
Index Testů 2014:

Logické/Bitové Operátory + Hradla (AND,OR,XOR,NOT)

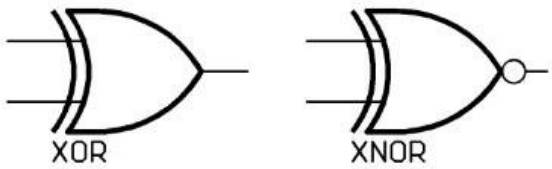
Bitové operátory

Celá nezáporná čísla mají ve světě počítačů díky dvojkové soustavě (téměř) jednotnou implementaci, jazyk C proto nabízí operátory, které usnadňují přístup k číslu jako k poli bitů.

Operace	A		NEBO		XOR		NE	
Operátor	Logický	Bitový	Logický	Bitový	Logický	Bitový	Logický	Bitový
	&&	&	||	|	C nemá	^	!	~
Příklad desítkově	6 && 3	6 & 3	6 || 3	6 | 3	6 XOR 3	6 ^ 3	!6	~6
Příklad binárně	110 && 011	110 & 011	110 || 011	110 | 011	110 XOR 011	110 ^ 011	!110	~110
Výsledek desítkově	1	2	1	7	0	5	0	4294967289
Výsledek binárně	1	10	1	111	0	101	0	$1^{29}001$

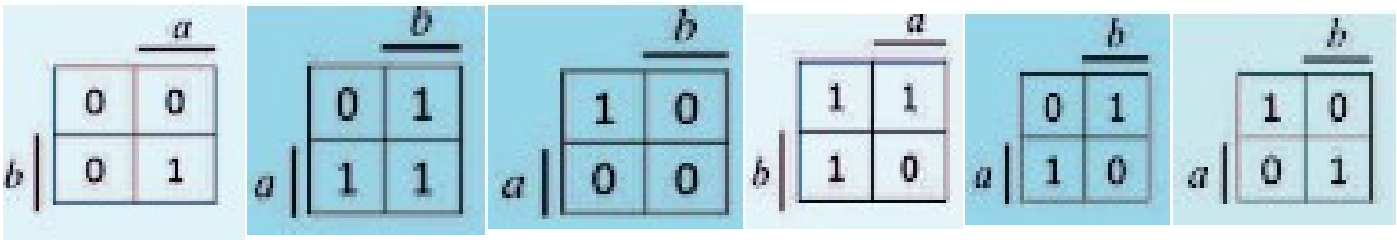


A	B	A alebo B	A a B	XOR
0	0	0	0	0
0	1	1	0	1
1	0	1	0	1
1	1	1	1	0



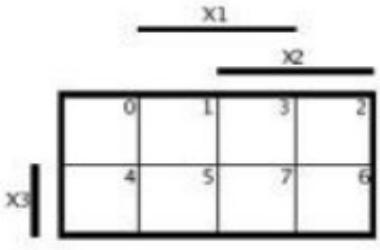
Karnaugh mapy hradel:

AND **OR** **NOR** **NAND** **XOR** **XNOR**



Karnaugh Mapy

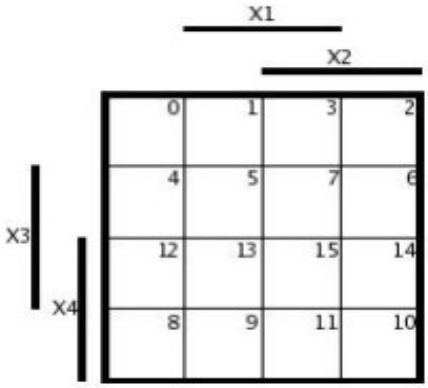
Součtová forma = smyčky z 1
Součinová forma = smyčky z 0



Jestliže buňky náležející některé proměnné obsahují celou smyčku, zapíšeme tuto
proměnnou do výrazu.

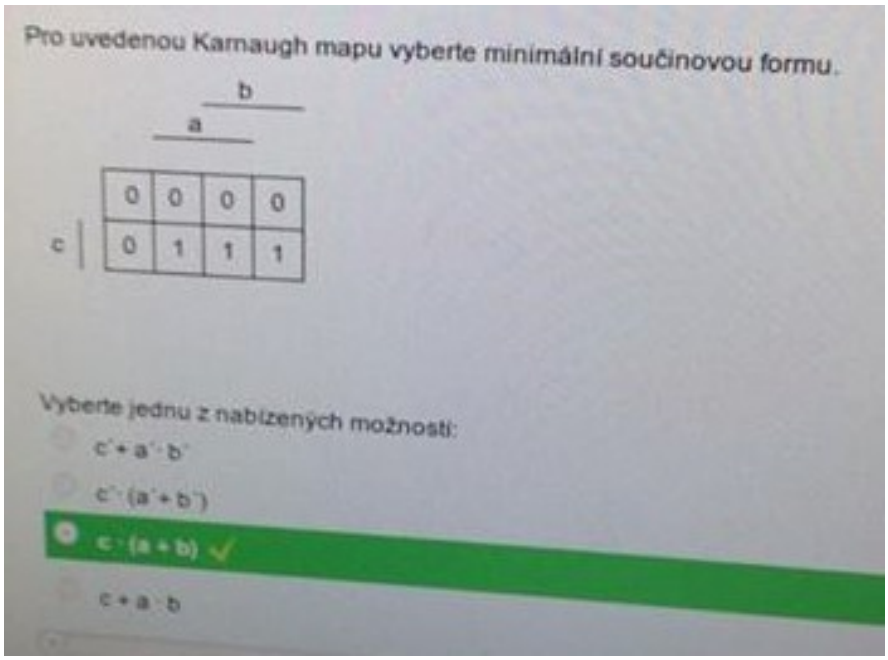
Jestliže buňky náležející některé proměnné neobsahují žádnou část smyčky, zapíšeme do výrazu tuto proměnnou v negaci (logická funkce NOT).

Jestliže buňky náležející některé proměnné obsahují jen část smyčky, tuto proměnnou ignorujeme.



Součtová (1):
Jednotlivé proměnné zapsané do výrazu mezi sebou logicky násobíme. Jednotlivé smyčky sčítáme.

Součinová (0):
Jednotlivé proměnne zapsané do výrazu mezi sebou sčítáme. Jednotlivé smyčky logicky násobíme.



Mocniny bi(y)tu/mi(y)bitu

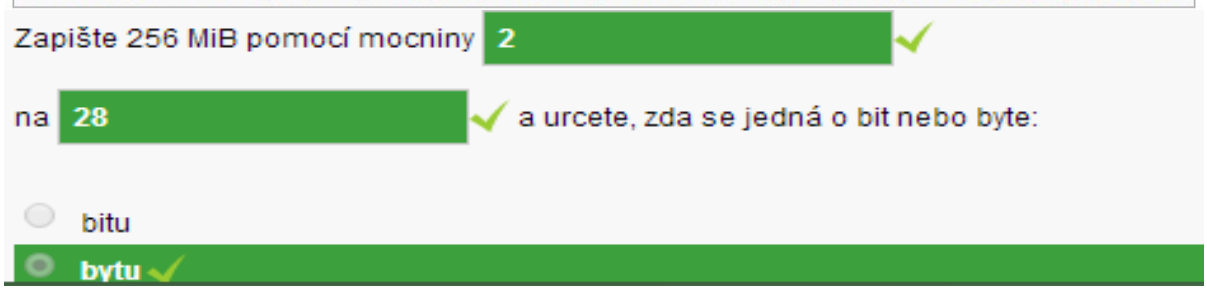
B = byte /// b = bit

k je kilo, a je to 10^3, neboli tisíc, M je mega, a je to 10^6, neboli milion, G je giga, a je to 10^9, neboli miliarda, T je tera, a je to 10^12, neboli bilion.....

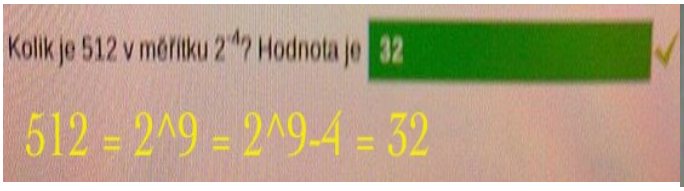
Předpona násobku/dílu	Značka násobku/dílu	Exponent násobku/dílu	Číselné vyjádření násobku/dílu
Tera	T	10^{12}	1 000 000 000 000
Giga	G	10^{9}	1 000 000 000
Mega	M	10^{6}	1 000 000
Kilo	k	10^{3}	1 000
Mili	m	10^{-3}	0, 001
Mikro	μ	10^{-6}	0, 000 001
Nano	n	10^{-9}	0, 000 000 001
Piko	p	10^{-12}	0, 000 000 000 001

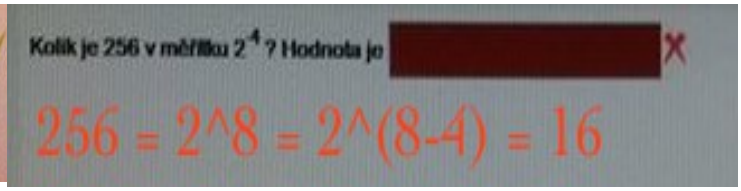
Převody (binární) - nible:

Binární předpony [e]				
Dvojkový řád *n*: 2^n	Nejbližší desítkový řád *k*: 10^k	Značka	Název	Hodnota
2^{10}	10^3	Ki	*kibi*	1 024
2^{20}	10^6	Mi	*mebi*	1 048 576
2^{30}	10^9	Gi	*gibi*	1 073 741 824
2^{40}	10^{12}	Ti	*tebi*	1 099 511 627 776
2^{50}	10^{15}	Pi	*pebi*	1 125 899 906 842 624
2^{60}	10^{18}	Ei	*exbi*	1 152 921 504 606 846 976
2^{70}	10^{21}	Zi	*zibi*	1 180 591 620 717 411 303 424
2^{80}	10^{24}	Yi	*yobi*	1 208 925 819 614 629 174 706 176
Poznámka: 10^k není rovno 2^n, je to jen nejblíže odpovídající mocnina; jsou z ní však odvozeny názvy i značky obdobně desítkovým předponám.				



Posun v měřítku:





Věty o boolean funkcích + Realizace boolean funkce:

- Úplná Boolean funkce je zobrazení $\{0,1\}^n$ do $\{0,1\}$
- Zápis Boolean funkce $f(x_{n-1}....x0)$ = IIM(1, 2, 9...) jednoznačně definuje:
  - úplnou Boolean funkci.
- Zápis Boolean funkce $f(x_{n-1}...x0)$ = II M(1,2,9...) + II D(4,5...) jednoznačně definuje
  - neúplnou Boolean funkci.

- Úplná Boolean funkce je zobrazení definičního oboru do oboru hodnot.
- Neúplná Boolean funkce je zobrazení f: definičního oboru do oboru hodnot.
- Boolean výraz jednoznačně definuje úplnou Boolean funkci.
- Neúplná (rozšířená) Boolean funkce je zobrazení f: $\{0,1\}^n$ do $\{0,1,X\}$
- úplná Boolean funkce je zobrazení $\{0,1\}^n$ do $\{0,1\}$
- Úplnou a neúplnou Boolean funkci jednoznačně definuje: pravdivostní tabulka.

Realizace Boolean Funkce:

Zapis do mapy, provedeme minimalizaci:

πM = 0, πD = X;

(Smyčky z 0)

Σm = 1, Σd = X;

(Smyčky z 1)

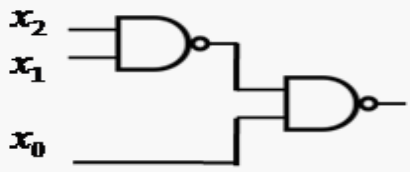
Boolean funkce je zadaná tvarem

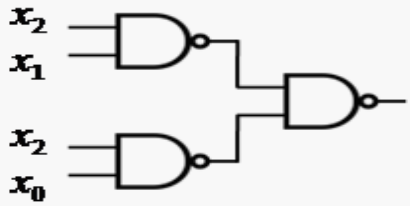
$f(x_2, x_1, x_0)=\Sigma m(1,3,6)+\Sigma d(5,7)$.

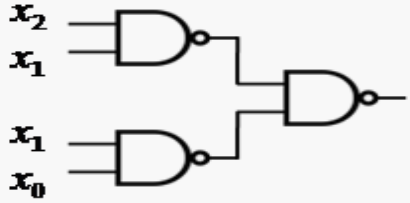
Vyberte odpovídající realizaci Boolean funkce.

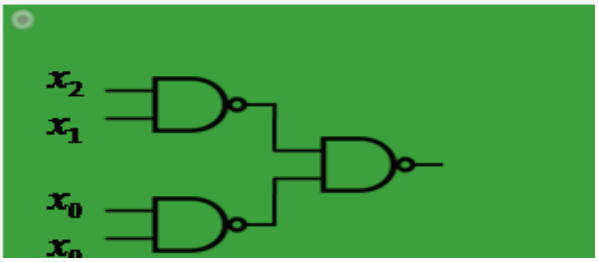
Záporné body.

Vyberte jednu z nabízených mo◆ností:

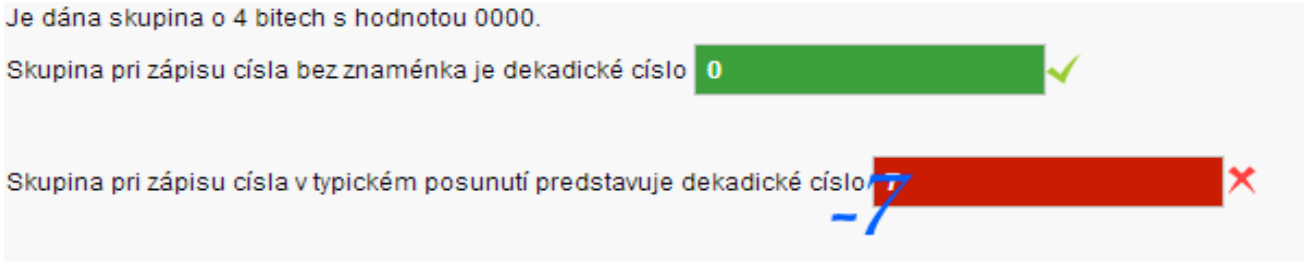








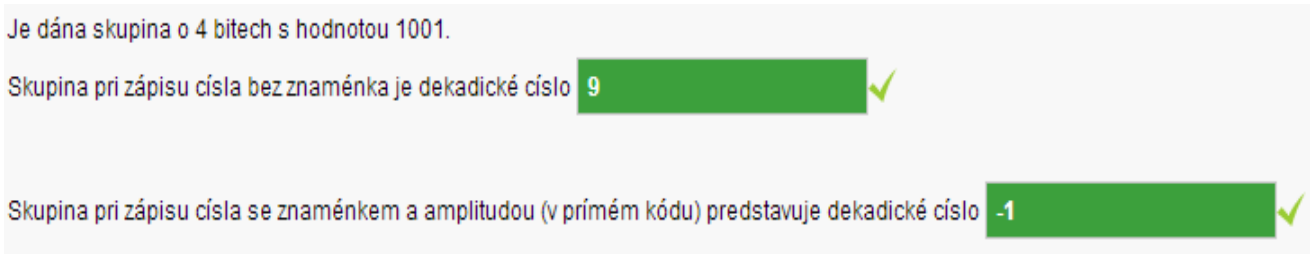
Posunutí / Přimý kod / dvojkový doplnek



Typické posunutí:

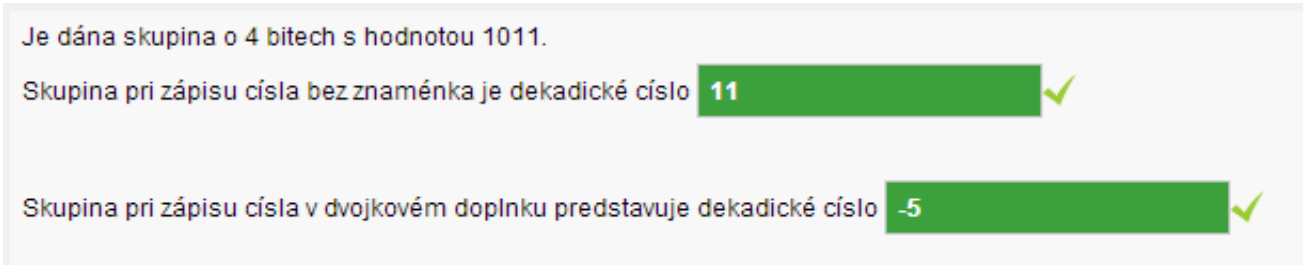
***podle vzorce 2^(n - 1) - 1 kde n je počet bitů**. Mělo by to být takhle, 2^(4 - 1) -1 ---> 2^3 -1 ---> 7.... Takže 13-7= 6*

Když máš zadání kde jsou pravdepodobne 4 bity, vždy odčitej 7 :)



Skupina při zápisu čísla se znaménkem a amplitudou

první bit je znaménkový (1 je minus, 0 je plus) a zbytek je to číslo.



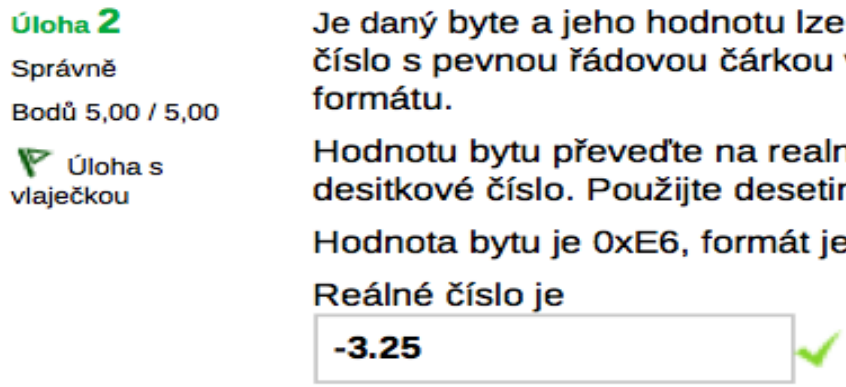
Skupina při zápisu čísla v dvojkovém doplňku představuje dekadické číslo:

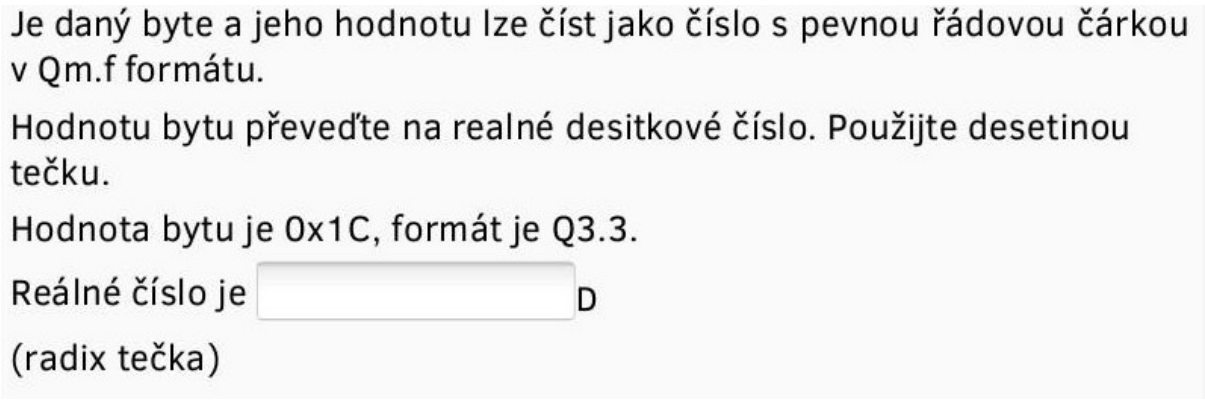
To by bylo -3, protože 1101 ---> (negace) 0010 + 1= 0011.

pokud to původní číslo je kladné, interpretace dvojkového doplňku je taky kladná, neměníme znaménko..

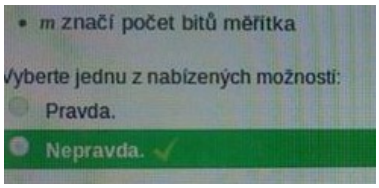
Převod na Qm.f formát

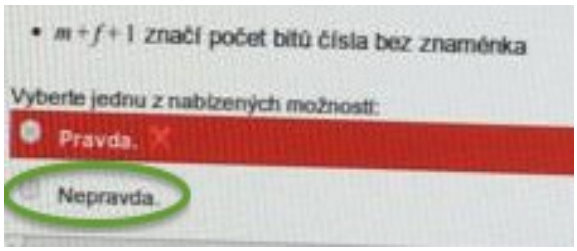
- *Převedeme zadané číslo na binár.*
- *MSB rozhoduje zda se jedná o +,- (+ = 0 , - = 1)*
- *Pokud máme záporné,provedeme dvojkový doplněk. (znegueš a přičteš 1)*
  - *pokud máme kladné , neprovádíme negaci ani přičitání!!*
- *Podle zadáného formátu upravíme výsledek Q3.3*

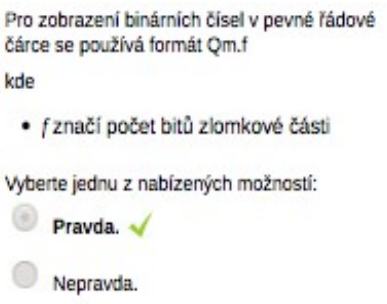


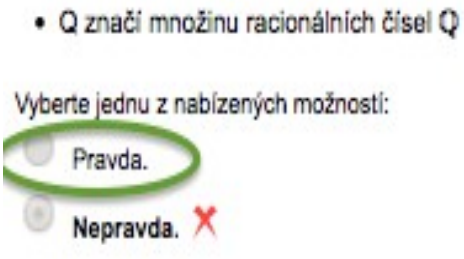


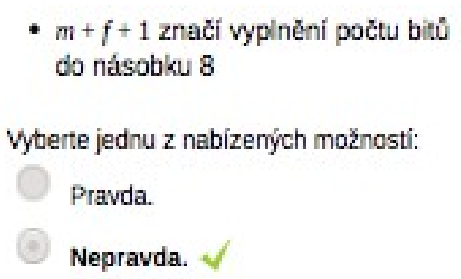
Teorie o Qm.f

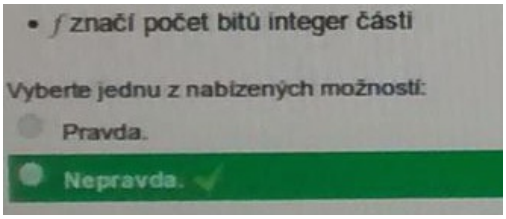






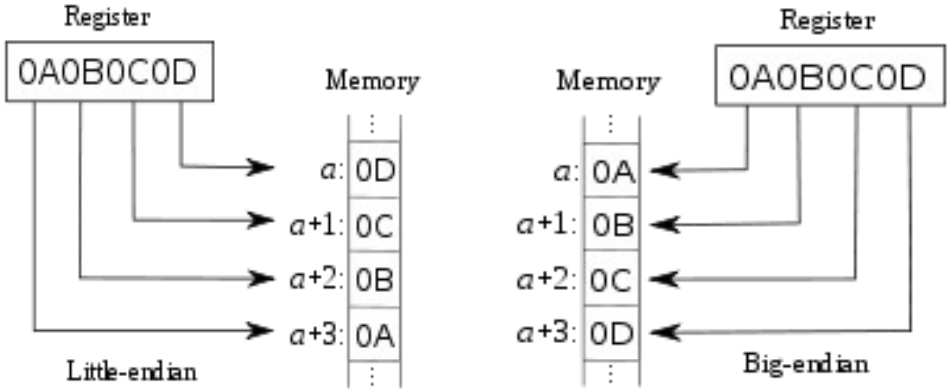


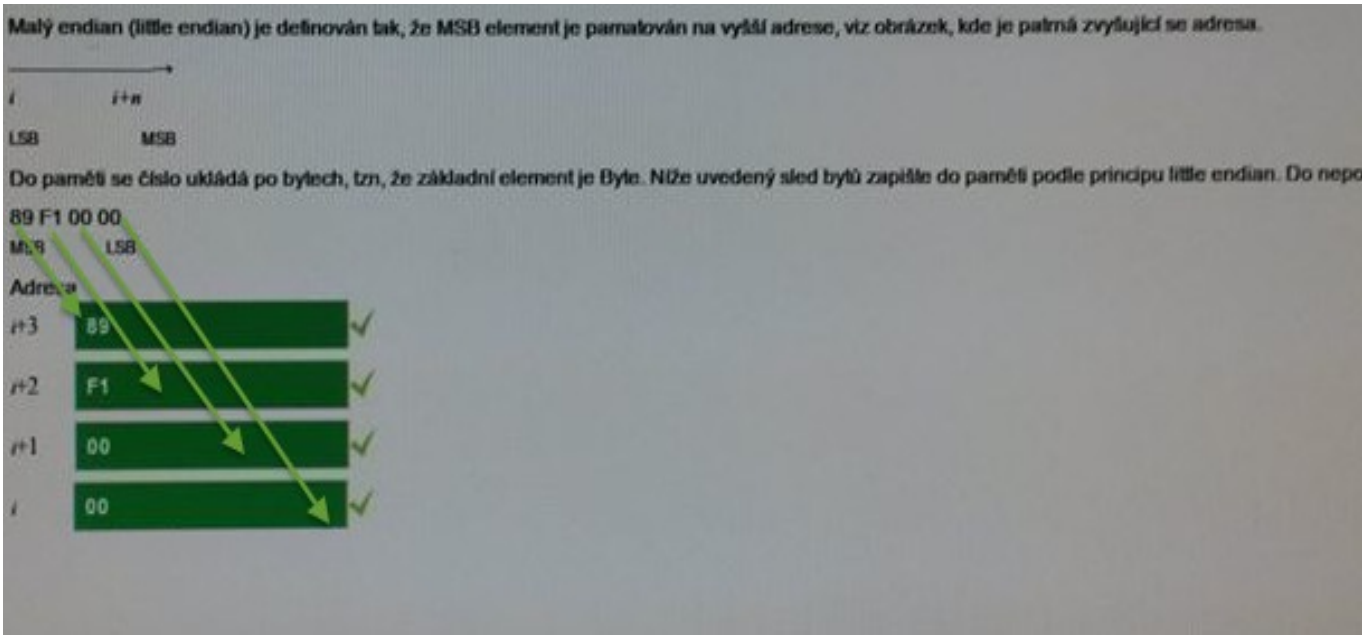




Little a Big Endian (Endianita)

Big = Řadíme za sebe / Little = Řadíme naopak
Word Memory : místo 2 bitu řadíme po 4 bitech





V případě že by jsme chtěli zapisovat na BIG endian bude pořadí v boxech opačně (00-00-F1-89)

!!Word Memory u endianu!!

Velký endian (big endian) je definován tak, že MSB element je pamatován na nižší adrese, viz obrázek, kde je patrná zvyšující se adresa.

i *i+n*

MSB LSB

Do paměti se číslo ukládá po slovech, tzn, že základní element je 16 bitové slovo. Níže uvedený sled bytů zapište do paměti podle principu big endian. Do nepoužitých adres použijte kombinaci 0000.

14 8C 2D 9A
MSB LSB

Pokud je v zadání ukládání po slovech používáme WORD memory !!!

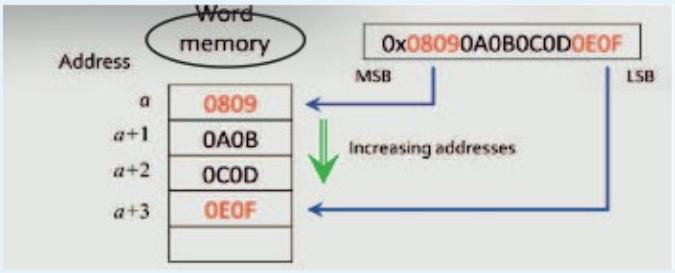
Adresa

i+3 ✗ 0000

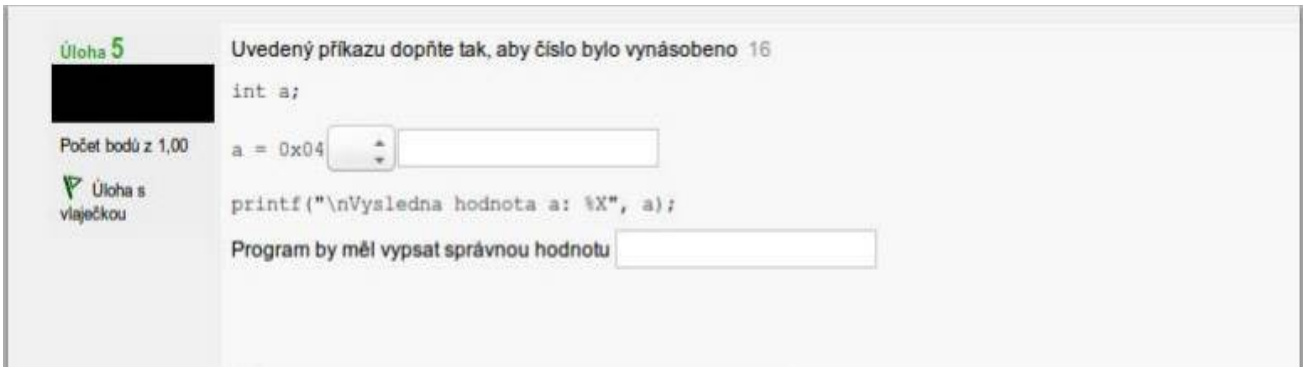
i+2 ✗ 0000

i+1 ✗ 2D9A

i ✗ 148C



Zápis násobení/dělení v Cčku

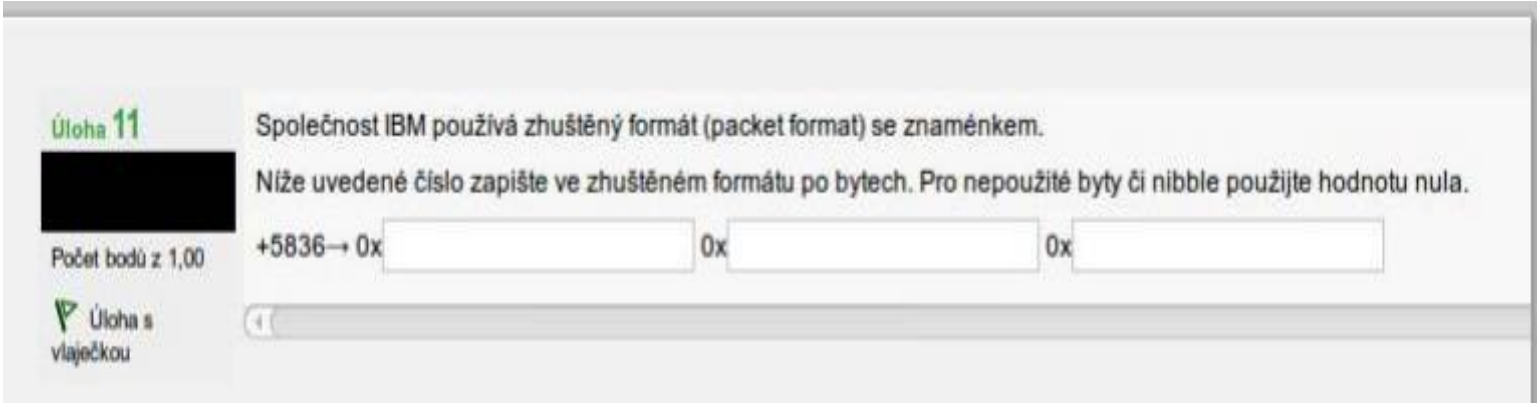


První ListBox:
Násobení: << (posun doleva)
Dělení: >> (posun doprava)

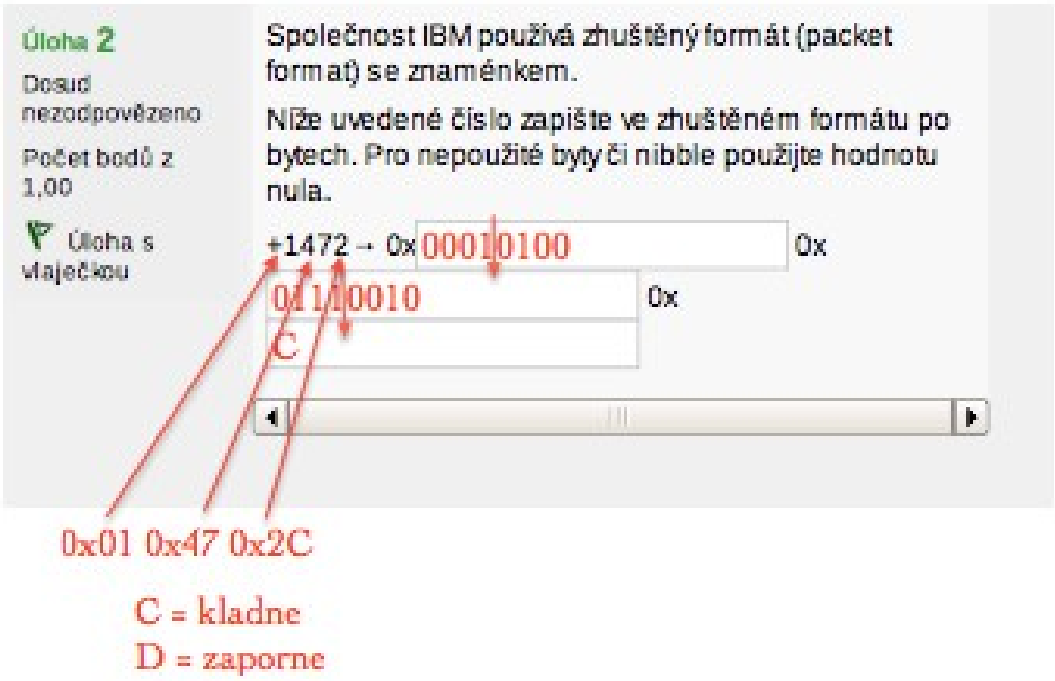
První Textbox:
Mocnina čísla (16) = 4

Druhý Textbox:
Hexa (0X04) převedeme a vynasobíme/vydělíme dle zadání pomocí hodnoty v prvním Textboxu.

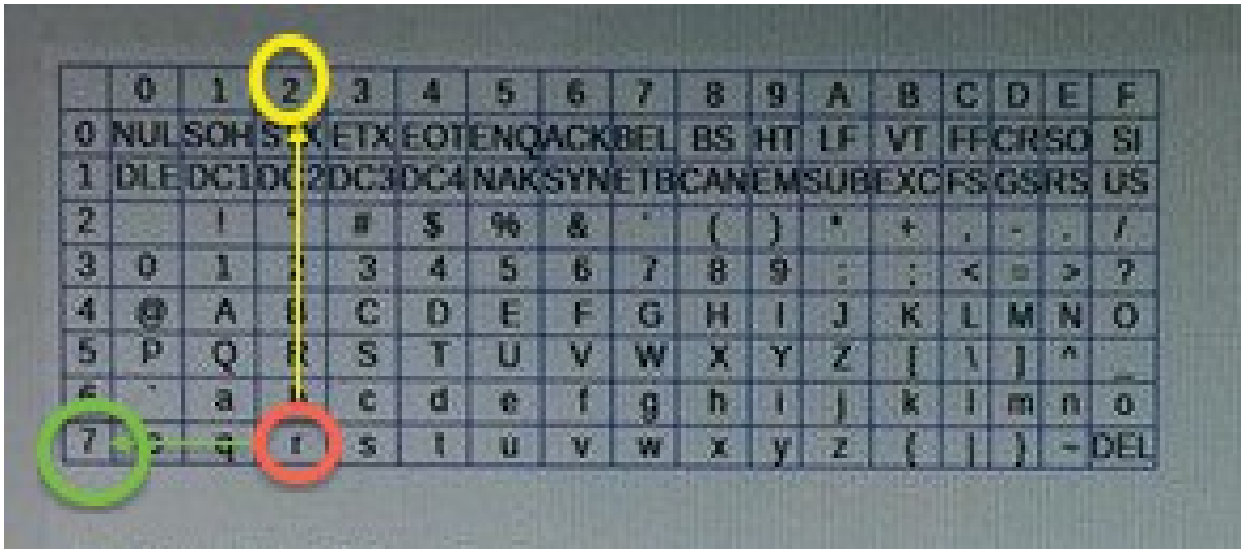
IBM Packet Format



1. Číslo převedeme do BCD kodu (binár po jednotlivých číslech) → 0000 0101 1000 0011 0110 (popřípadě vlevo to doplnit nulama)
2. Pokud je kladné bude poslední nible (na poslední pozici) písmeno **C** , pokud bude zaporné píšeme **D**
3. Převadíme jednotlíve čísla BCD a zapisujeme do jednotlivých boxu ve formatu 0x05 0x83 0x6**C**



UTF8 Kodování textu

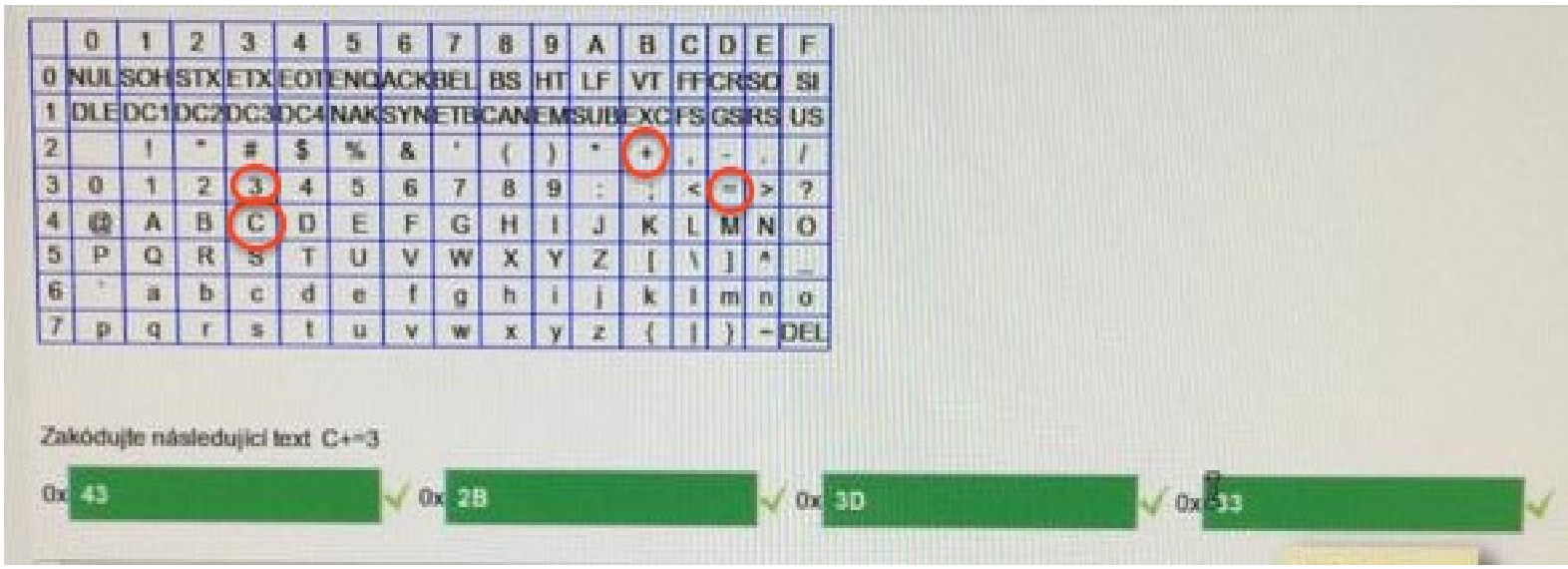


	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
0	NUL	SOH	STX	ETX	EOT	ENQ	ACK	BEL	BS	HT	LF	VT	FF	CR	SO	SI
1	DLE	DC1	DC2	DC3	DC4	NAK	SYN	ETB	CAN	EM	SUB	ESC	FS	GS	RS	US
2		!	"	#	$	%	&	'	(	)	*	+	,	-	.	/
3	0	1	2	3	4	5	6	7	8	9	:	;	<	=	>	?
4	@	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O
5	P	Q	R	S	T	U	V	W	X	Y	Z	[	\	]	^	_
6	`	a	b	c	d	e	f	g	h	i	j	k	l	m	n	o
7	p	q	r	s	t	u	v	w	x	y	z	{	|	}	~	DEL

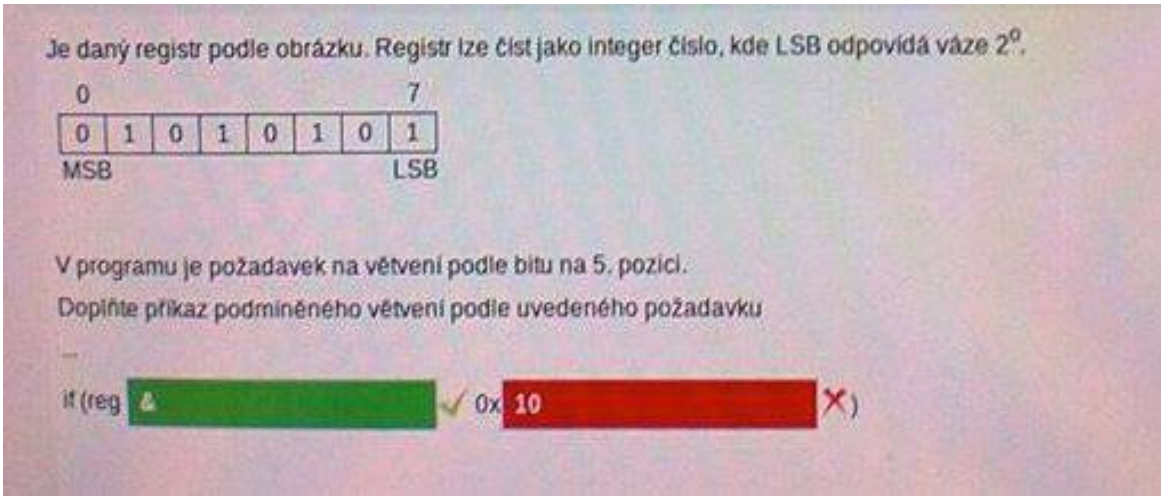
- Jednotlivé znaky ze zadání vyhledáváme v tabulce.
- Zapisujeme nejdříve index zleva (zelený na obrázku), poté horní index (žlutý na obrázku).

Viz obrázek: znak r kodujeme jako 0x72

Poté jednotlivé znaky zapisujeme do přislušných boxu



Větvení podle bitu/Znulování/Negace/Nastavení



- Vytvoříme si masku 0000 0000
- Podle zadané pozice doplníme do masky jedničku (Pozice jsou jasně vidět nad zadaným registrem).

(V případě na obrázku 5. pozice: 0000 0000 → 0000 0100)

- Výslednou masku převedeme na HEX .(00000100 → 0x04) – tento výsledek zapisujeme do druhého TextBoxu.

Podle operace:

Větvení podle bitu:

- DO prvního boxu zapisujeme bitový AND (&).

Znulování:

- Výslednou masku je ještě před zapisem třeba znegovat
- DO prvního boxu zapisujeme bitový AND (&).

Negace (flip):

- DO prvního boxu zapisujeme bitový XOR (^).

Nastavení:

- DO prvního boxu zapisujeme bitový OR

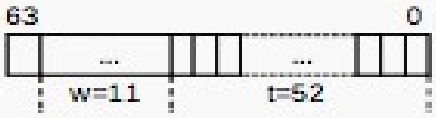
Hodnota posunutí (BIAS) – Float 754(2008)

Úloha 1

Dosud nezodpovězeno

Počet bodů z 1,00

Úloha s vlaječkou



Definice polí pro výměnný binární formát čísla v pohyblivé řádové čárce (binary floating-point number) binary64 podle IEEE 754-2008 je na obrázku.

Hodnota posunutí (bias) pro uvedený formát je .

mezi hranicemi 52(t) a 63 je 11 pozicí (w) : $2^{(11-1)} - 1 = 1023$

Definice polí pro výměnný binární formát čísla v pohyblivé řádové čárce (binary floating-point number) binary64 podle IEEE 754-2008 je na obrázku.

Hodnota posunutí (bias) pro uvedený formát je 1023

Sestavení UTF-8 formátu

Počet bitů	Poslední kódová pozice	Byty			
		Vedoucí (Leading)	Následný (Continuation)	Následný (Continuation)	Následný (Continuation)
7	U+007F	0xxx xxxx			
11	U+07FF	110x xxxx	10xx xxxx		
16	U+FFFF	1110 xxxx	10xx xxxx	10xx xxxx	
21	U+1FFFFF	1111 0xxx	10xx xxxx	10xx xxxx	10xx xxxx

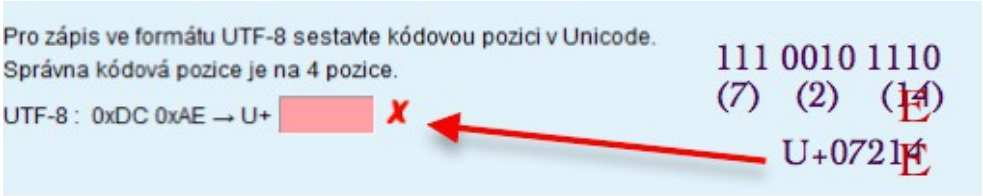
Pro kódovou pozici U+0741 sestavte UTF-8 formát

0x 0x

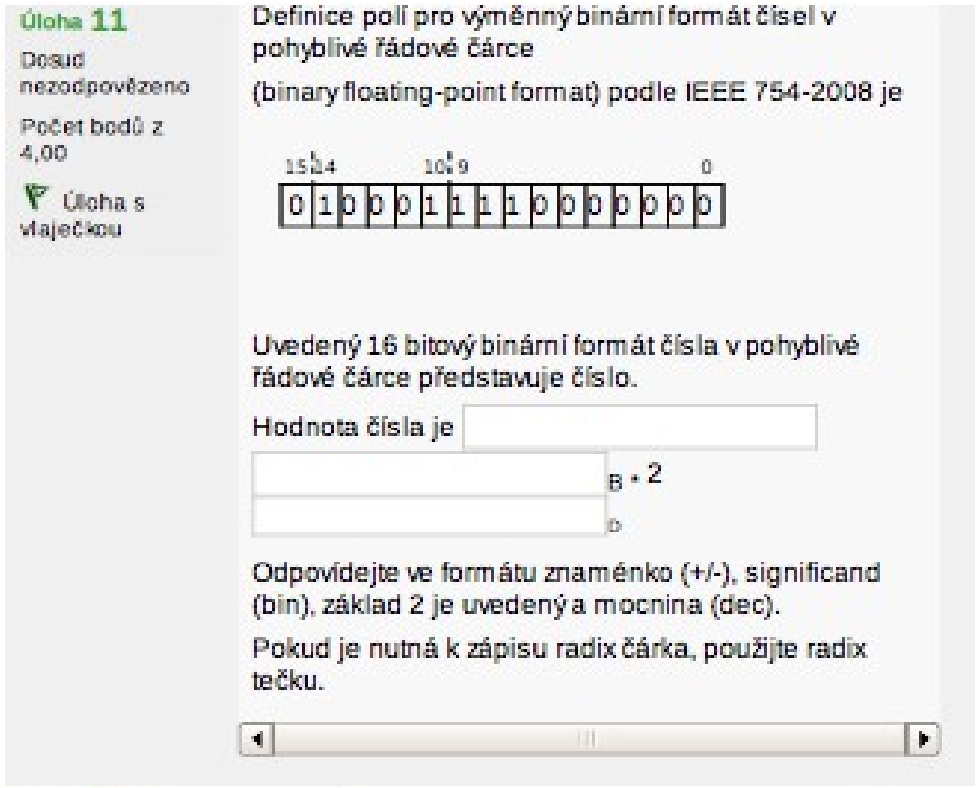
- Převedeme číslo ze zadání do bin. po bitech (7,4,1) → 0111 0100 0001.
- Následně podle zadáného formátu **(U+07FF)** vybíráme leading a continuation byte z tabulky. → (**110**x xxxx) a (**10**xx xxxx)
- Do continuation bytu (10xx xxxx) zapisujeme na pozice x prvních 6 bitu z prava (od zadu) → (**10**000 0001)
- Do leading bytu (110x xxxx) pokračujeme se zapisováním s dalšíma 5 bitama čísla → (**110**1 1101)
- Do prvního boxu zapisujeme leading bit v hexu, do dalších boxu zapisujeme continuation byty, také v hexu.

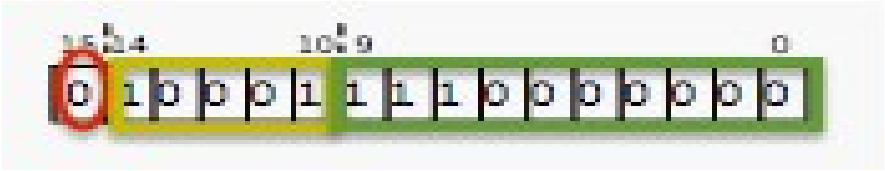
Opačný převod:

Převedeme z hexu na binára, odstraníme leading/continuation předpisy a jednotlive převadime zpatky na hex do určeného předpisu.



Definice polí pro výměnny binární formát čísel ve floating point



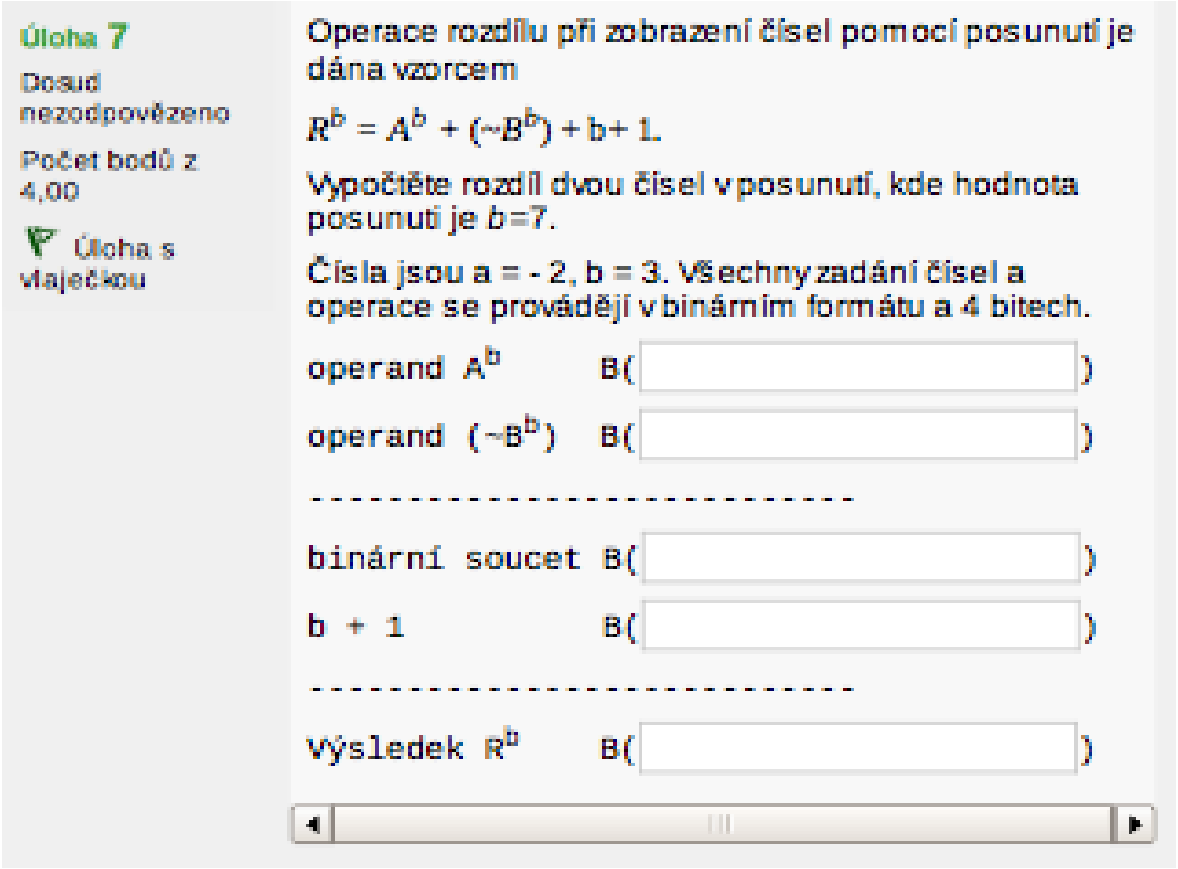


Hodnota čísla (První box): Zde zapisujeme pouze znaménko výsledného čísla. Bit na 15té (nejvyšší) pozici určuje (0 je +, 1 je -)

Druhý box: Napíšeme **1.** a opíšeme část oddělenou napravo (od 9 – 0) pozice, nuly již nezapisujeme → 1.111

Třetí box: Bity mezi vyznačenými čárkami (14-10) (10001) = 17. Vypočteme typické posunutí: 2^(w-1)-1, čili 2^(5-1)-1=15. poté posunutí odečteme od čísla (17-15 = 2)

Výpočet rozdilu, posunutí



Pozor: Zde nevíme jak to ministr Zdrálek vymyslel. Vypadá to , že malé b je hodnota posunutí a velké B je zadání čísla (b=3) :(

Operand A^b = (-2+7) → 5 → **0101**

Operand neg-B^b = Zde musíme provést ještě bitovou negaci. 3 → 0011->1100 + 0111 = 19 (10011) – Musíme však zapisovat ve 4 bitech! → **0011.**
(Zde se negace vyskytuje náhodně dle zadání. Nemusí být nutně u operandu B^b)

Binární součet = (Ab + Bb) → 5 + 19 → 11001 → zapisujeme na 4 bity → **1001**

b + 1 = (7 + 1) = **1000**

Výsledek (Rb) = (Dle rovnice na začátku zadání)

Polynom číselné soustavy (teoretické otázky):

pro i..

- je řád koeficientu v číselné soustavě
- *jiná možnost..*

pro b

- je základ číselné soustavy
- *jiná možnost*

pro b^n-1....

- nemá rozsah, jeho hodnota je b^n-1

pro b^-m-1....

- jiná možnost

pro b^-m....

- jiná možnost

pro b^i....

- je v rozsahu b^-m až b^n-1
- *jiná možnost*

pro a^i

- je v rozsahu 0 až b-1

pro n:

- udává počet číslic celočíselné části čísla nebo
- *jiná možnost*

pro Om:

- udává počet číslic zlomkové části čísla

pro On

- udává počet čislic celočíselné část čísla

Příznaky N/Z/V/C

Jsou dány čísla *a* a *b*. Vypočtěte jejich součet na 4 bity a nastavte příznaky NZVC.

Příznaky mají hodnotu 0 nebo 1.

0101

+0110

1011 ✓ B

Příznaky

N = 1 ✓

Z = 0 ✓

V = 1 ✓

C = 0 ✓

Příznak N značí:

Vyberte jednu z nabízených možností:

- Přetečení.
- Nulový výsledek
- Záporný výsledek ✓
- Přenos do vyšší váhy či vyššího řádu

N = Sign bit. První bit výsledku

Příznak Z značí:

Vyberte jednu z nabízených možností:

- Nulový výsledek ✓
- Přenos do vyšší váhy či vyššího řádu
- Přetečení.
- Záporný výsledek

Z = Zero bit. Je 1 pouze pokud jsou všechny bity výsledku 0.

V = Overflow bit Výsledek nelze správně zobrazit, tzn.: výsledek je špatně, když sčítáme záporné a kladné číslo nemůže to nastat, nebo když odečítáme čísla se stejným znaménkem, dá se to jednodušše zjistit jako XOR carry Z a do MSB.

Přetečení pro čísla ve dvojkovém doplňku.

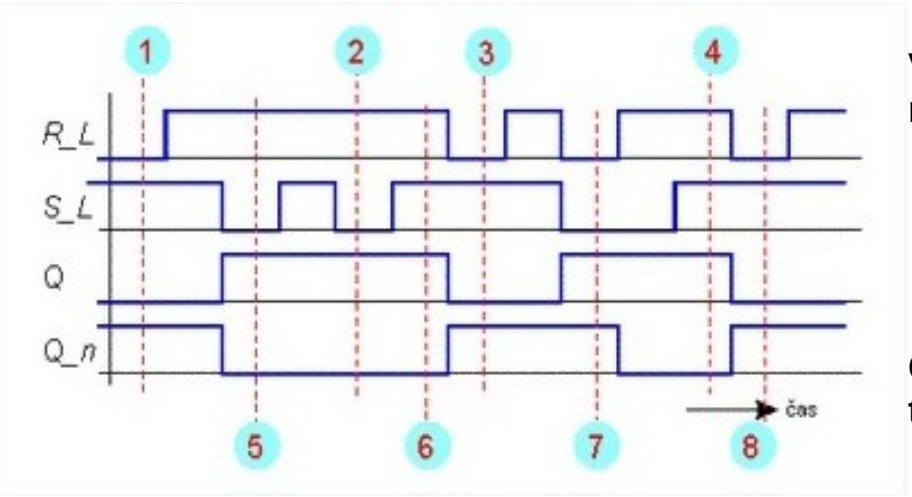
Příznak C značí:

Vyberte jednu z nabízených možností:

- Přetečení.
- Nulový výsledek
- Přenos do vyšší váhy či vyššího řádu ✓
- Záporný výsledek

C = Carry bit Zbude nam zbytek po operaci. (1 kterou už nelze zapsat).

Časový průběh RS klopných obvodů



Všímáme si výstupu na R_L,S_L(R,S)

1 = vyvýšená 

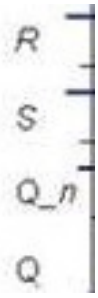
0 = na čáře 

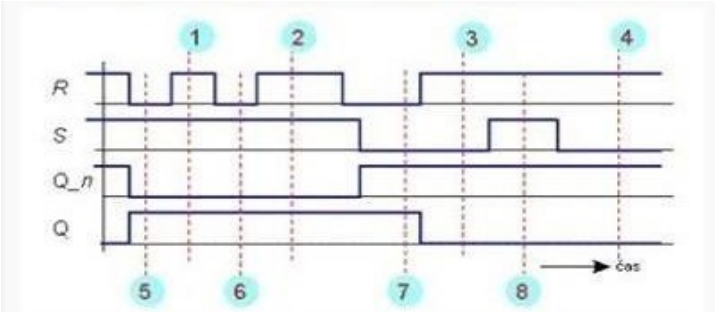
Čisla v modrem jsou ty časové body :)

NOR

(Rozhoduje logická 1)

R	S	
0	0	pamět
0	1	nastavení
1	0	nulování
1	1	Ilegální akce



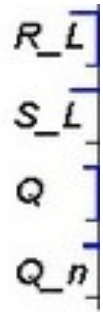


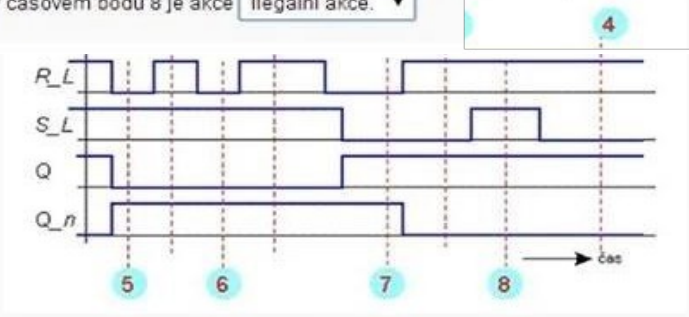
V časovém bodu 1 je akce ilegální akce.
V časovém bodu 2 je akce ilegální akce.
V časovém bodu 3 je akce nulování.
V časovém bodu 4 je akce nulování.
V časovém bodu 5 je akce nastavení.
V časovém bodu 6 je akce nastavení.
V časovém bodu 7 je akce paměť.
V časovém bodu 8 je akce ilegální akce.

NAND

(Rozhoduje logická 0)

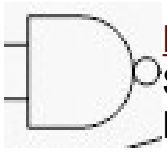
R_L	S_L	
0	0	Ilegální akce
0	1	nulování
1	0	nastavení
1	1	pamět





V časovém bodu 1 je akce paměť.
V časovém bodu 2 je akce paměť.
V časovém bodu 3 je akce nastavení.
V časovém bodu 4 je akce nastavení.
V časovém bodu 5 je akce nulování.
V časovém bodu 6 je akce nulování.
V časovém bodu 7 je akce ilegální akce.
V časovém bodu 8 je akce paměť.

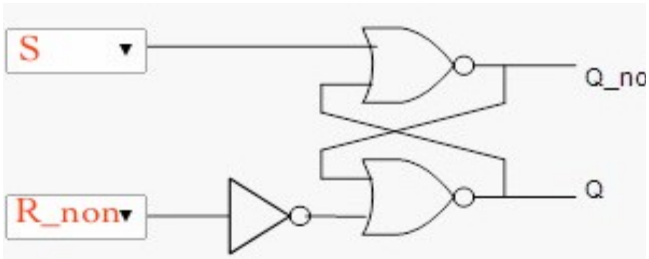
Paměťové elementy



RS NAND:

S_L → Q

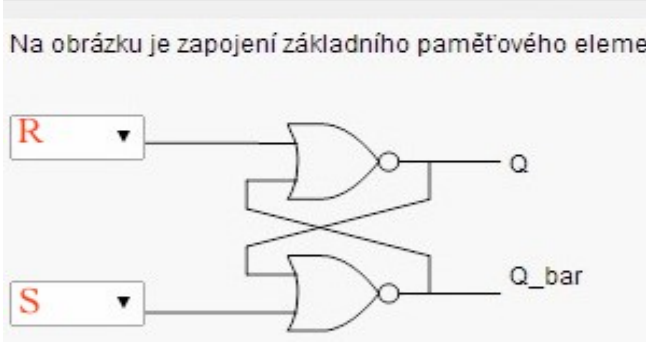
R_L → Q_L (Q_non, Q_Bar)

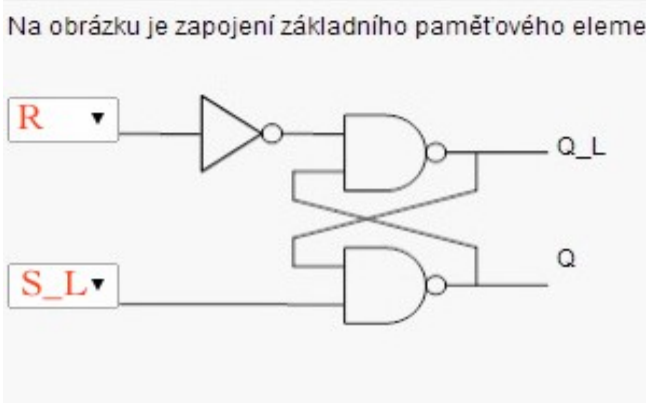


RS NOR:

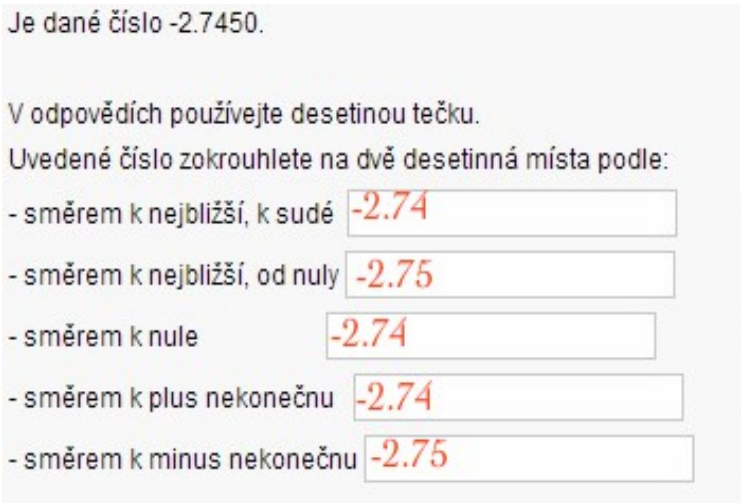
R → Q

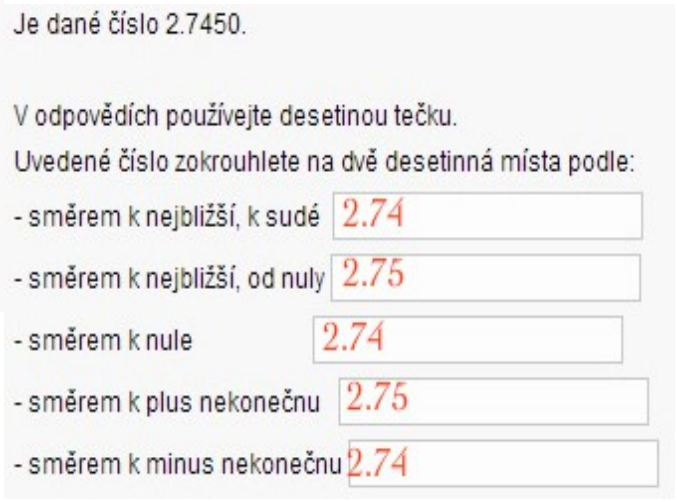
S → Q_L (Q_non, Q_Bar)





Zaokrouhlování čísel





Na 2.des. Místa !!:

Zaokrouhlování se „láme“ klasicky v 0.5
2.745 (dolu) → 2.74
2.745 (nahoru) → 2.75

2.333 (dolu) → 2.33 *(2.330)*
2.333 (nahodu) → 2.33 *(2.334)*

---

směrem k nejbližší, k sudé
Zaokrouhlíme na sudé číslo. :)
směrem k nejbližší, od nuly
Zaokrouhlení na číslo, které je dále od 0.

Následující 3 zaokrouhlení ignorují část za zaokrouhlením (0.01 - 0.99 se zaokrouhluje stejně):
(2.333 (nahodu) → 2.34*)*
směrem k nule
Zaokrouhlení na číslo, které je bližší 0
směrem k plus nekonečnu
Zaokrouhlíme na číslo, které je blíže +nek. :)
směrem k minus nekonečnu
Zaokrouhlíme na číslo, které je blíže -nek. :)

a) 1.465
k nej., k sudé: 1.46
k nej., od nuly: 1.47
k nule: 1.46
k +inf: 1.47
k -inf: 1.46

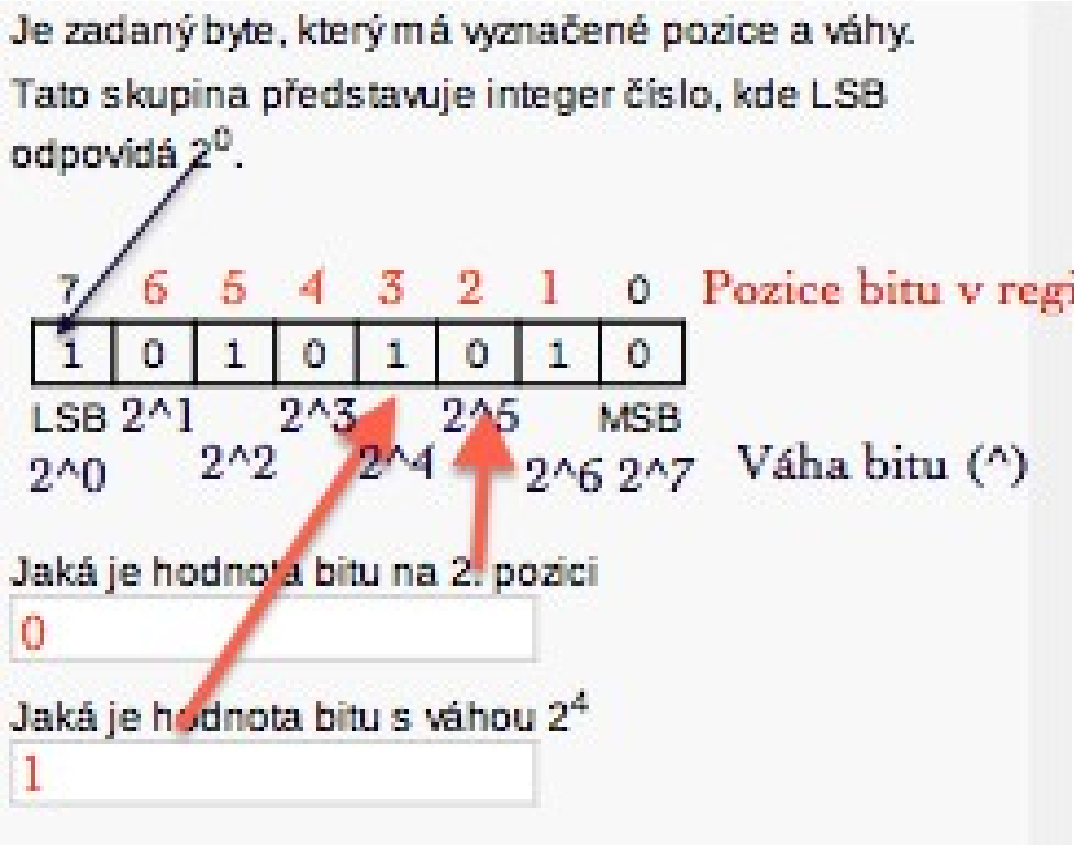
b) 2.333
k nej., k sudé: 2.33
k nej., od nuly: 2.33
k nule: 2.33
k +inf: 2.34
k -inf: 2.33

c) - 1.465
k nej., k sudé: - 1.46
k nej., od nuly: - 1.47
k nule: - 1.46
k +inf: - 1.46
k -inf: - 1.47

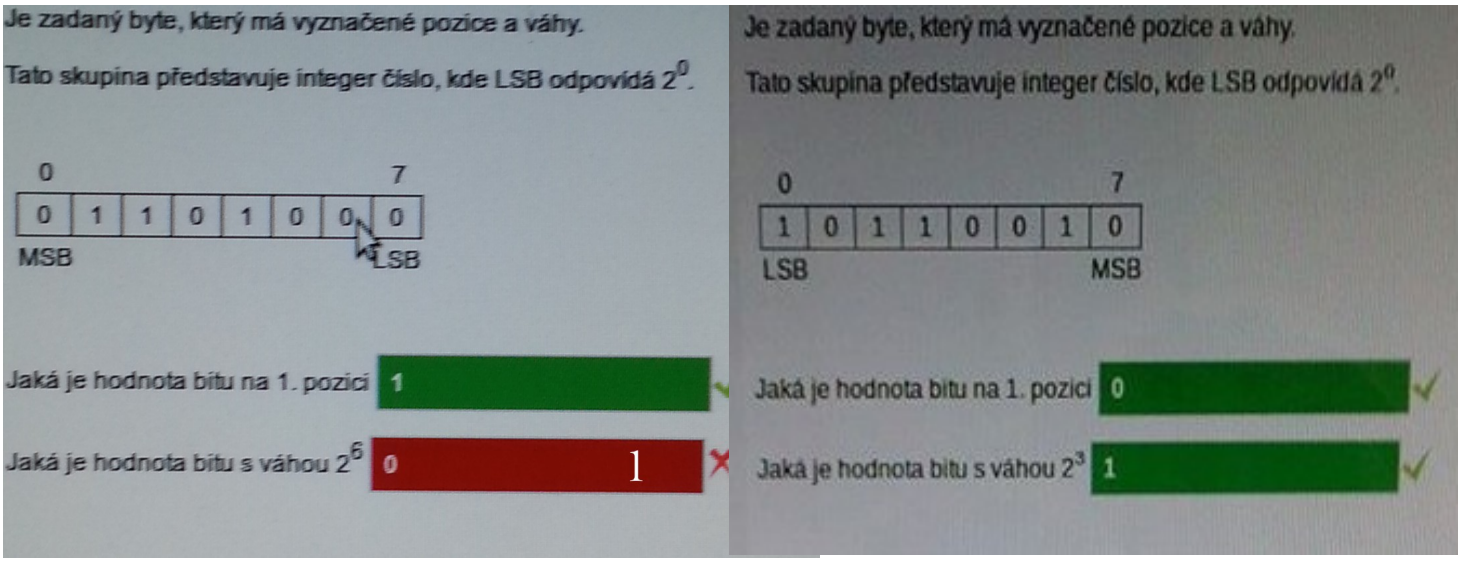
d) - 2.348
k nej., k sudé: - 2.35
k nej., od nuly: - 2.35
k nule: - 2.34
k +inf: - 2.34
k -inf: - 2.35

e) - 5.560
vše: - 5.56

MSB/LSB



- **Dávat pozor čemu odpovidá 2^0 !**
- **Dávat pozor na pozice v registru.**



IEEE 754-2008

IEEE 754-2008 x IEEE 754-1985

Standard IEEE 754-2008 definuje, krom jiného, formáty čísel s pohyblivou řadovou čárkou (FP čísel).
V praxi se však setkat i s jinými názvy.
Potom název "single precision" odpovídá názvu binary32 podle IEEE 754-2008.

Vyberte jednu z nabízených možností:
- Pravda.
- Nepravda.

Standard IEEE 754-2008 definuje, krom jiného, formáty čísel s pohyblivou řadovou čárkou (FP čísel).
V praxi se však setkat i s jinými názvy.
Potom název "double precision" odpovídá názvu binary64 podle IEEE 754-2008.

Vyberte jednu z nabízených možností:
- Pravda.
- Nepravda.

single (binary32) = Single Precision
double (binary64) = Double/Extended Precision
extended (binary128) = Quadruple Precision

Declet:

Definice pro výměnný dekadický formát čísel v pohyblivé řádové čárce
(decimal floating-point number) podle IEEE 754-2008
používá pojem declet.
Declet je:

Vyberte jednu z nabízených možností:
- dvě dekadické číslice v BCD kódu
- koeficient v binárním tvaru
- tři dekadické číslice zakódované do 10 bitů
- koeficient v dekadickém tvaru

DPD Encoding:

Definice pro výměnný dekadický formát čísel v pohyblivé řádové čárce
(decimal floating-point number) podle IEEE 754-2008
používá pojem - nahuštěný formát, nahuštěné decimální kódování
(densely packed format, densely packed decimal encoding).
Pojem je:

Vyberte jednu z nabízených možností:
- kódovací tabulka, jak zakódovat zhuštěný BCD kód a zpět.
- kódovací tabulka, jak kódovat binární čísla v BCD kódu a zpět.
- kódovací tabulka, jak zakódovat tři decimální číslice do 10 bitů a zpět.
- kódovací tabulka pro kódování exponentu floating point čísla.

Significant m:

Standard IEEE 754-2008 uvádí formulu

$(-1)^S * m * b^e$

Uvedená formula používá pro significand m jako integer číslo s příslušnou hodnotou exponentu.

Vyberte jednu z nabízených možností:
- Pravda.
- Nepravda.

pojem posunutý exponent E.

Pojem je: Součet exponentu e a posunutí (bias)

Booleaova Algebra – Pravidla

Summary of Boolean algebra

- 1 + 1 = 1
- 1 + 0 = 1
- !1 = 0

- 1 . 1 = 1
- 1 . 0 = 0
- !0 = 1

- !(!a) = a
- a + a + ... = a
- 1 + a = 1

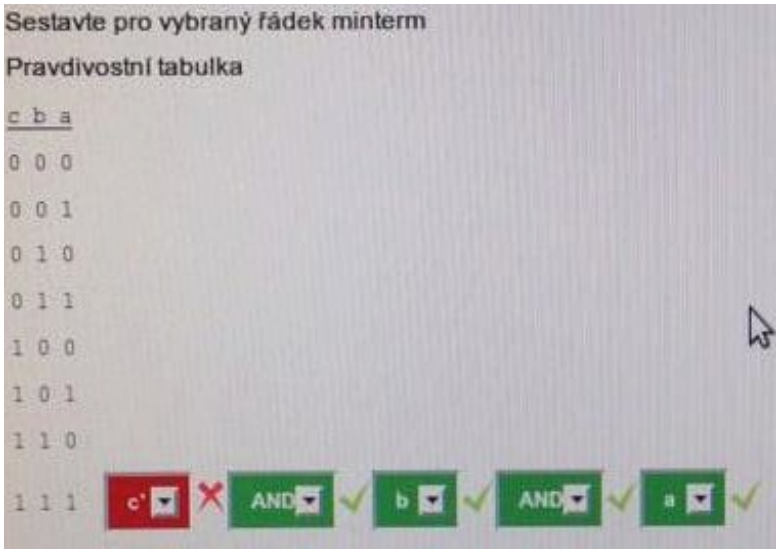
- a . a . ... = a
- 0 . a = 0

- a + !a = 1
- 1 + anything = 1
- anything + !(anything) = 1

- a . !a = 0
- 0 . anything = 0
- anything . !(anything) = 0

	Zákony součtu	Zákony součinu
Zákon idempotence	$a+a=a$	$a.a=a$
Zákon absorbce	$a+ab=a$	$a.(a+b)=a$
Zákon absorbce negace	$a+\bar{a}b=a+b$	$a.(\bar{a}+b)=ab$
Zákon komutativní	$a+b=b+a$	$ab=ba$
Zákon asociativní	$a+(b+c)=(a+b)+c$	$a.(bc)=(ab)c$
Zákon distributivní	$a+(bc)=(a+b).(a+c)$	$a.(b+c)=ab+ac$
Neutrálnost nuly a jedničky	$a+0=a$	$a.1=a$
Agresívnost nuly a jedničky	$a+1=1$	$a.0=0$
Zákon vyloučeného třetího	$a+\bar{a}=1$	$a\bar{a}=0$
Zákon negace	$\bar{0}=1$	$\bar{1}=0$
Zákon dvojité negace	$\bar{\bar{a}}=a$	
De Morganovy zákony	$\overline{a+b}=\bar{a}\bar{b}$	$\overline{ab}=\bar{a}+\bar{b}$

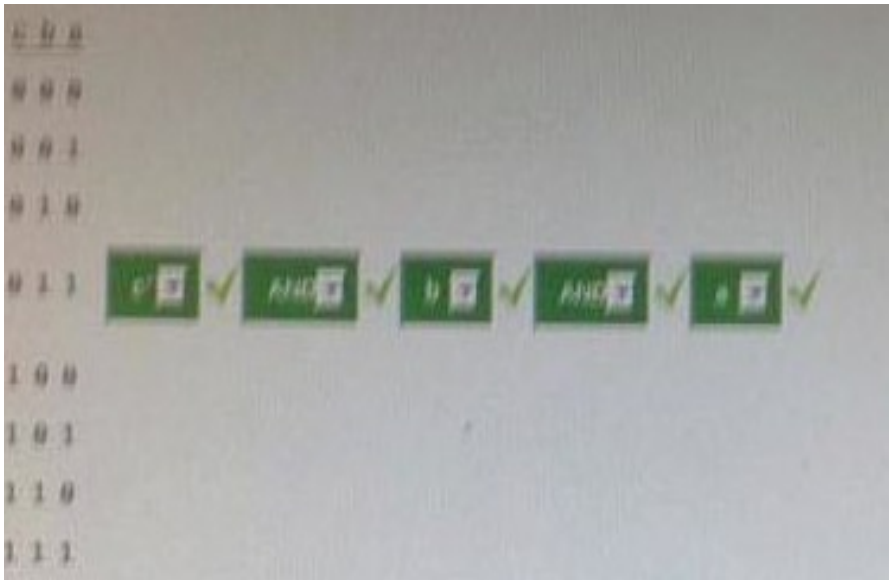
Minterm / Maxterm



minterm/maxterm se provadí pro určitý řádek tabulky

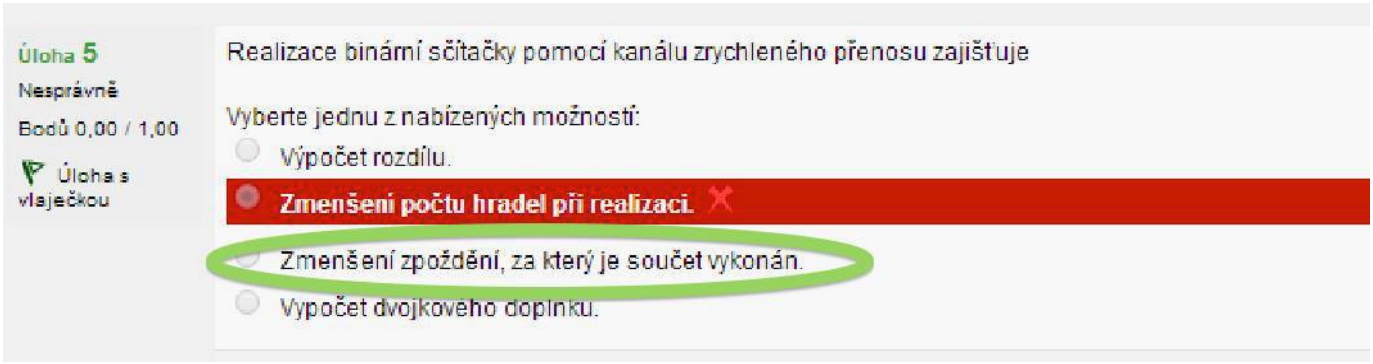
minterm -
kde je 1, píšem proměnnou
kde je 0, píšem její negaci
mezi nima AND

maxterm
kde je 0, píšem proměnnou
kde je 1, píšem její negaci
mezi nima OR



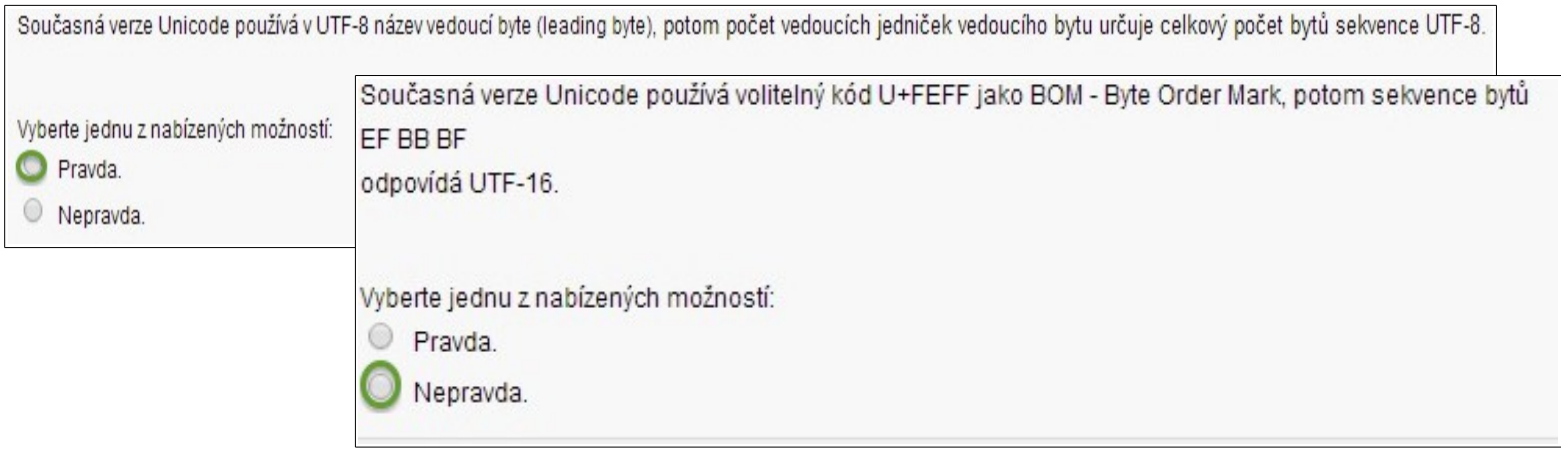
Ostatní (teoretické) otázky

Binární sčítačka:

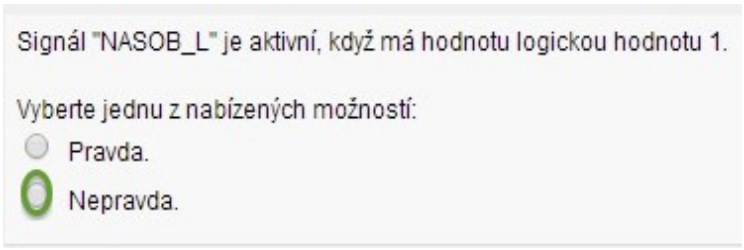


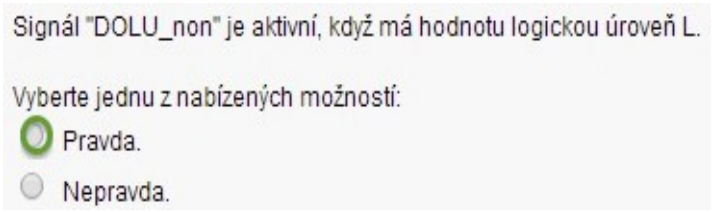
Hradla XOR (První Test)

UTF Teorie (Test 7)

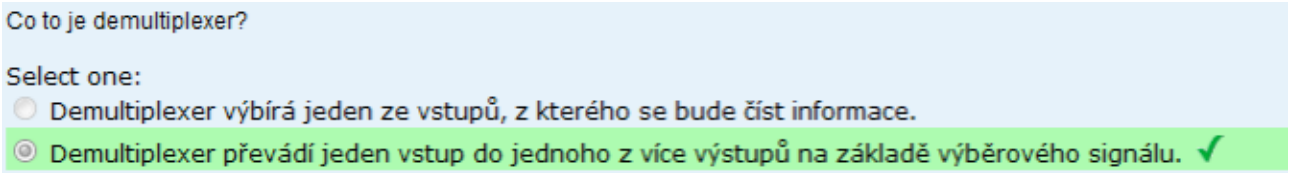


Signály ? (Test7) – Nepotvrzené výsledky

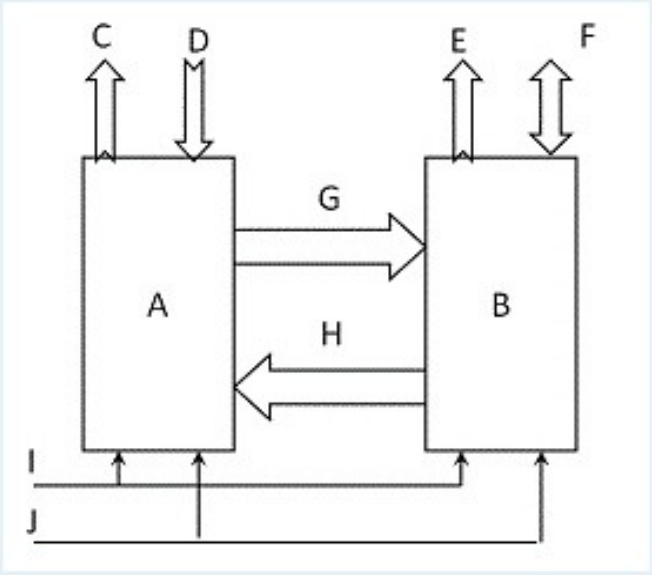




Demultiplexer



Synchronní číslicové systémy



Na výše uvedeném obrázku je blokové schéma synchronního číslicového systému.

Písmenu A odpovídá Datová jednotka ✗

Písmenu D odpovídá Příkazy ✓

Písmenu H odpovídá Synchronizační nebo inicializační signál ✗

Řadič A

- Přijímá vstup – Příkaz *(D)*
- Odesilá výstup – Příznak *(C)*
- Odesilá Signál Řízení směrem k „Datové Jednotce“ *(G)*
- Přijímá Stavový signal od „Datové Jednotky“ *(H)*

Datová Jednotka B

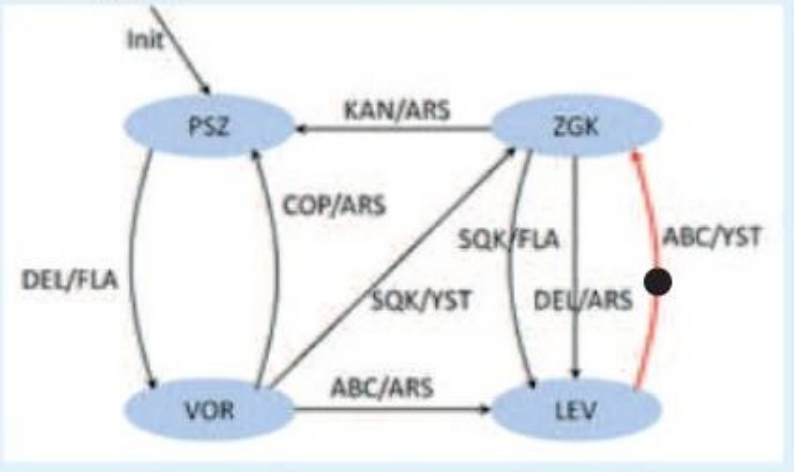
- Přijímá i odesilá data (datový vstup/výstup) *(F)*
- Odesilá výstup přiznaku *(E)*
- Přijímá Signál řízení od „Řídící jednotky“ *(G)*
- Odesilá stavový výstup do „Řidící jednotky“ *(H)*

A: řadič (řidící jednotka)
B: Datová jednotka (řízený objekt)

C+E: příznaky
D: příkazy
F: datové vstupy, výstupy
G: signály řízení (Řidící signály)
H: podmínky, stavy

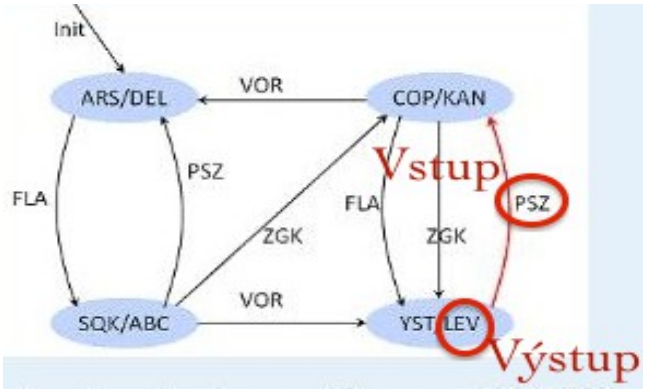
Finite State Machine – Automaty Končených Stavů (FSM)

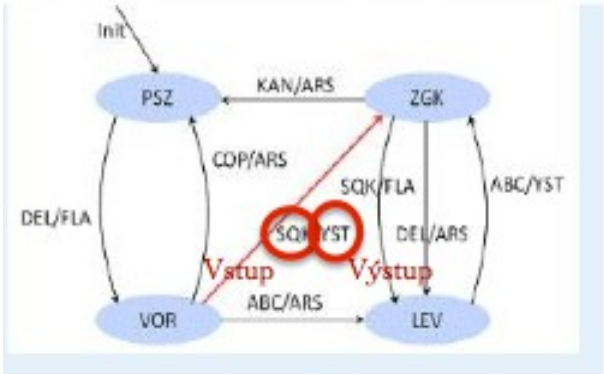
Ukazkový příklad:



- Všimáme si červené šipky odkud vychází, tím získáme **současný** stav .
- *Pokud se na počatečním políčku vyskytuje více identifikací (např. COP/KAN) zapisujeme ten první (levý). Druhý (pravý) nam označuje výstup!!*
    - *V případě na ukazkovém příkladě vychazí šipka z políčka (**LEV) – LEV je tedy současný stav***

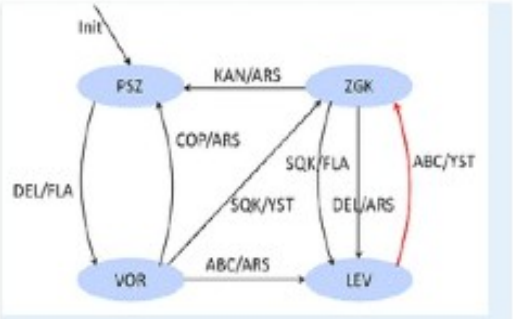
- Dále je třeba zjistit **Současný** vstup/výstup.
- **Můžou nastat dvě možnosti:**





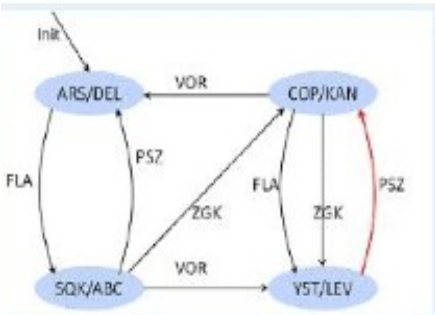
- *V případě na ukázkovém příkladě je Současný vstup ABC a výstup YST.*

- Poslední úkol je zjistit **následující stav,** ten se nachází v políčku kde červená šipka končí.
- *Pokud se na konečném políčku vyskytuje více identifikací (např. COP/KAN) zapisujeme ten první (levý).*
    - V případě na ukázkovém příkladě končí šipka na poli ZKG



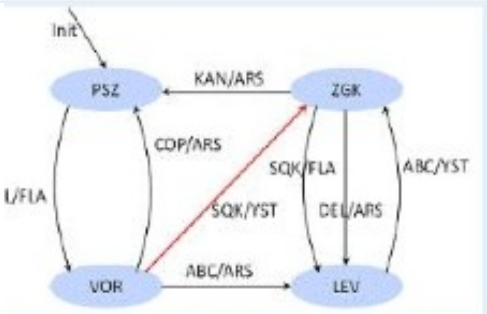
Pro červeně vyznačenou hranu na obrázku sestavte následující větu.

FSM se nachází v současném ✓ stavu ABC ✗ LEV
a při předcházejícím ✗ vstupu LEV ✗ ABC
generuje současný ✓ výstup YST ✓ YST
a přechází do následujícího ✓ stavu ZGK ✓ ZGK



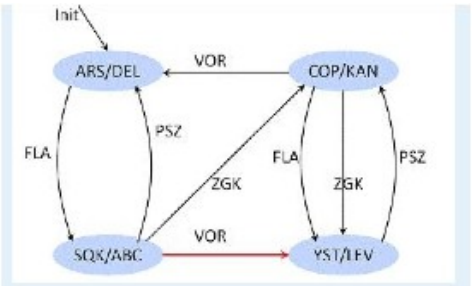
o červeně vyznačenou hranu na obrázku sestavte následující větu.

SM se nachází v současném ✓ stavu LEV ✗ YST
generuje současný ✓ výstup COP ✗ LEV
při následujícím ✗ vstupu ZGK ✗ PSZ
echází do následujícího ✓ stavu YST ✗ COP



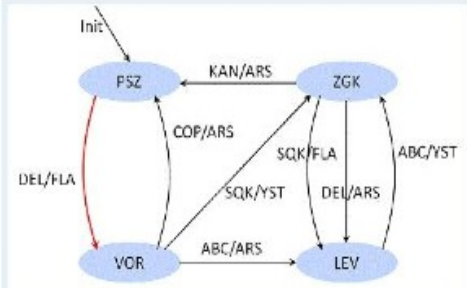
ervně vyznačenou hranu na obrázku sestavte následující větu.

se nachází v současném ✓ stavu ARS ✗ VOR
následujícím ✗ vstupu VOR ✗ SQK
uje současný ✓ výstup YST ✓ YST
ází do následujícího ✓ stavu ZGK ✓ ZGK



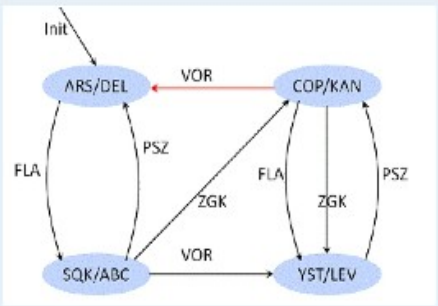
Pro červeně vyznačenou hranu na obrázku sestavte následující větu.

FSM se nachází v současném ✓ stavu ZGK ✗ SQK
a generuje následující ✗ výstup LEV ✗ ABC
a při následujícím ✗ vstupu SQK ✗ VOR
přechází do následujícího ✓ stavu LEV ✗ YST



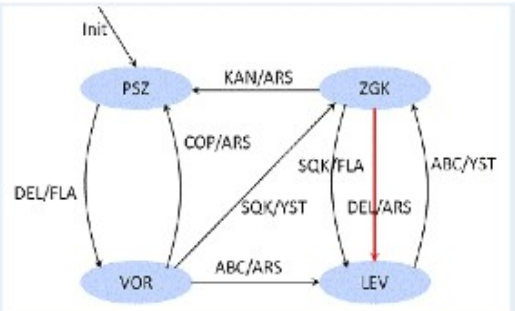
Pro červeně vyznačenou hranu na obrázku sestavte následující větu.

FSM se nachází v současném ✓ stavu FLA ✗ PSZ
a při předcházejícím ✗ vstupu PSZ ✗ DEL
generuje současný ✓ výstup PSZ ✗ FLA
a přechází do následujícího ✓ stavu PSZ ✗ VOR



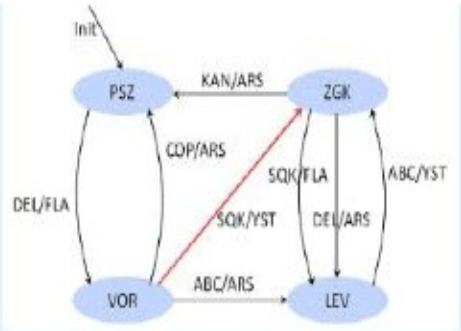
Pro červeně vyznačenou hranu na obrázku sestavte následující větu.

FSM se nachází v předcházejícím ✗ stavu ARS ✗ COP
a generuje současný ✓ výstup VOR ✗ KAN
a při předcházejícím ✗ vstupu PSZ ✗ VOR
přechází do následujícího ✓ stavu KAN ✗ ARS



Pro červeně vyznačenou hranu na obrázku sestavte následující větu.

FSM se nachází v současném ✓ stavu ZGK ✓ ZGK
a při následujícím ✗ vstupu SQK ✗ DEL
generuje současný ✓ výstup DEL ✗ ARS
a přechází do následujícího ✓ stavu FLA ✗ LEV



Pro červeně vyznačenou hranu na obrázku sestavte následující větu.

FSM se nachází v současném ✓ stavu ARS ✗ VOR
a při následujícím ✗ vstupu VOR ✗ SQK
generuje současný ✓ výstup YST ✓ YST
a přechází do následujícího ✓ stavu ZGK ✓ ZGK