

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 desetiny!)
- <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 Karnaughovy mapy)

Index Testů 2014:

Test 1

- Vybrat typ hradla dle C zápisu.....Strana 3
- Základy Booleovy algebry.....Strana 25
- Minterm + Maxterm.....Strana 26
- Pravdivostní tabulky hradel.....Strana 3
- Karnaughovy mapy hradel.....Strana 3
- Rozpoznat ikonu hradla.....Strana 3

Test 2

- Převody mezi soustavami
- Realizace Boolean funkce.....Strana 6
- MSB/LSB váhy.....Strana 23
- Minimalizace karnaugh mapy.....Strana 4
- Mocniny bitu.....Strana 5
- Věty o boolean zobrazení.....Strana 6

Test 3

- Typické posunutí.....Strana 7
- Operátory v jazyku C.....Strana 3
- Převody desetinných čísel mezi soustavama
- Věty o boolean zobrazení.....Strana 6
- Realizace Boolean funkce.....Strana 6
- AND, XOR, NOT ze dvou bin. Čísel.....Strana 3
- Mibity/byty a jejich mocniny.....Strana 5

Test 4

- Převod desetinných čísel mezi soustavami
- Nasobení/dělení a jeho zápis v C.....Strana 10
- Posun v měřítku.....Strana 5
- Polynom číselné soustavy.....Strana 18
- MSB/LSB a větvení.....Strana 13
- Little/big Endian a zápis čísel do registru.....Strana 9

Test 5

- Příznaky NZVC.....Strana 19
- Zákodování textu podle tabulky.....Strana 12
- AND, XOR, NOT operace s binárem (s převodem do 10 soustavy).....Strana 3
- polynom číselné soustavy.....Strana 18
- typické posunutí.....Strana 7
- gibit/byt a jeho mocniny.....Strana 5

Test 6

- BIAS.....Strana 13
- IBM packet Format.....Strana 14
- Standarty IEEE 754-2008.....Strana 24
- Pozice v registru (LSB/MSB).....Strana 23
- Zápis nasobení/dělení v C.....Strana 10
- Operace rozdílu.....Strana 17
- Zakodování pomocí UTF-8.....Strana 15
- Výměny binární formát čísel.....Strana 16

Test 7

- Pam. Elementy (flip-flop).....Strana 21
- Teorie o UTF-8Strana 27
- Časový průběh RS klop. Obvodu.....Strana 20
- Zaokrouhlování čísla.....Strana 22
- Rozpoznání mocniny.....Strana 5
- Qm.f formát.....Strana 8
- příznaky NZVC.....Strana 19
- operace součtu při zobrazení čísel posunutí.....Strana 17

Ostatní teoretické otázky.....Strana 27

Synchronní číslicové systémy.....Strana 28

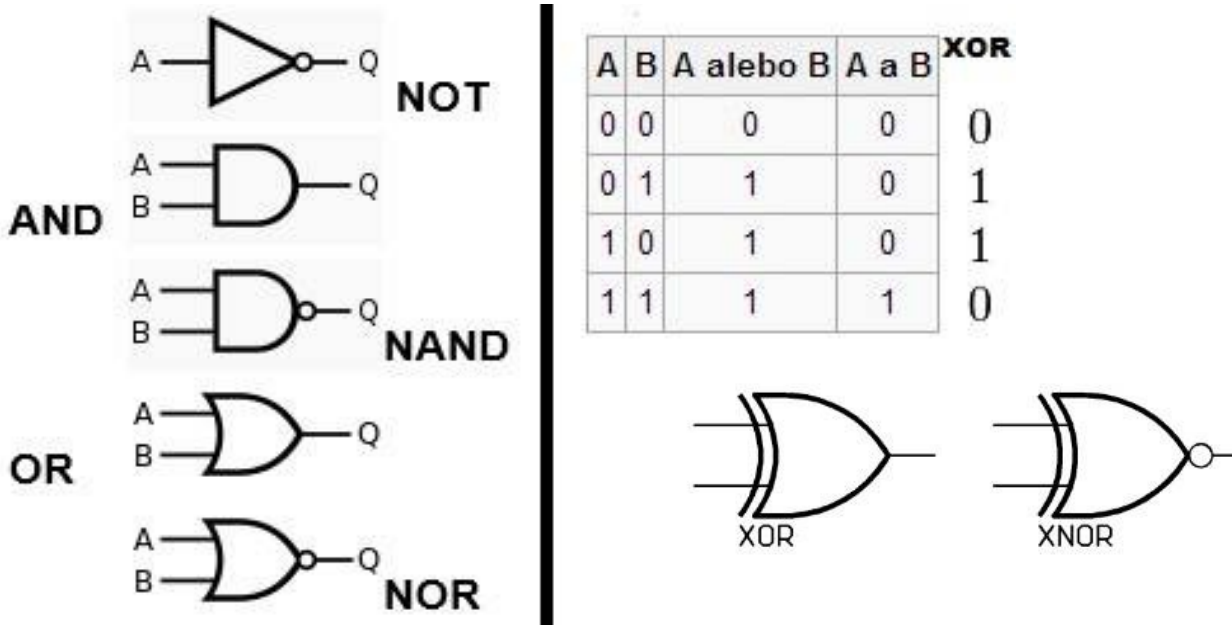
FSM (Automaty končených stavů).....Strana 29-30

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

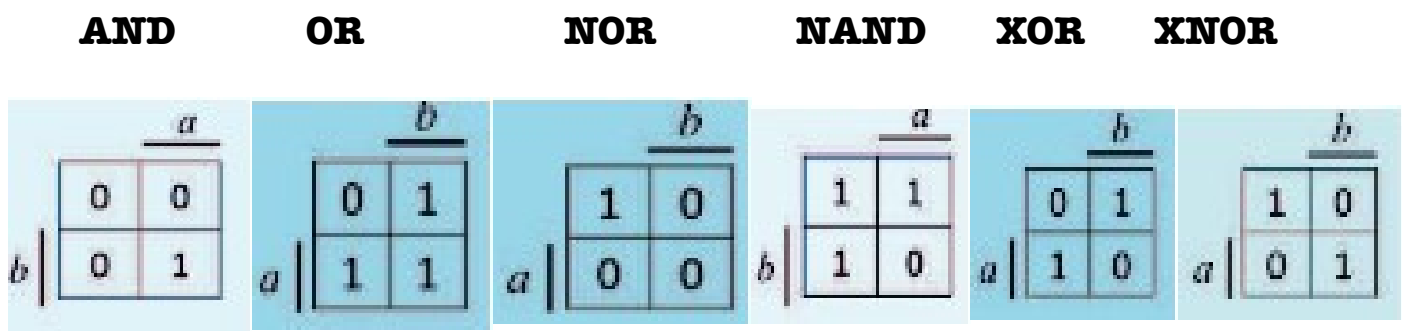
Bitové operátory

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 1		NEBO 0		XOR		NE	
	Logický	Bitový	Logický	Bitový	Logický	Bitový	Logický	Bitový
Operátor	&&	&			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



Karnaugh mapy hradel:



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).

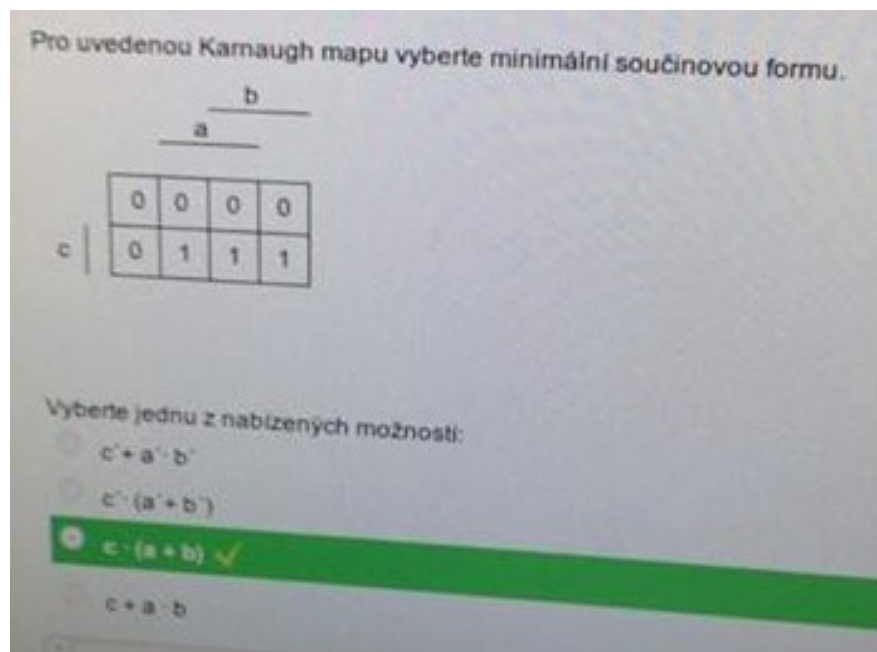
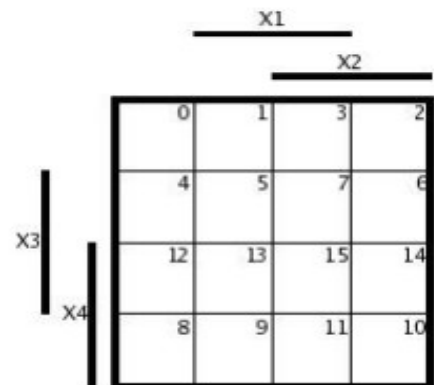
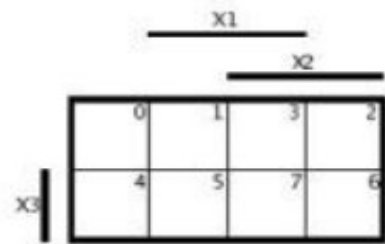
Jestliže buňky náležející některé proměnné obsahují jen část smyčky, tuto proměnnou ignorujeme.

Součtová (1):

Jednotlivé proměnné zapsané do výrazu mezi sebou logicky násobíme. Jednotlivé smyčky sčítáme.

Součinová (0):

Jednotlivé proměnné 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

Zapište 1 kb pomocí mocniny ✓ na ✓ a určete, zda se jedná o bit nebo byte. (Např. zápis mocniny 3^2 zapišete jako 3 na 2).

- bitů ✓
 bytů

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	<i>kibi</i>	1 024	
2^{20}	10^6	Mi	<i>mebi</i>	1 048 576	
2^{30}	10^9	Gi	<i>gibi</i>	1 073 741 824	
2^{40}	10^{12}	Ti	<i>tebi</i>	1 099 511 627 776	
2^{50}	10^{15}	Pi	<i>pebi</i>	1 125 899 906 842 624	
2^{60}	10^{18}	Ei	<i>exbi</i>	1 152 921 504 606 846 976	
2^{70}	10^{21}	Zi	<i>zibi</i>	1 180 591 620 717 411 303 424	
2^{80}	10^{24}	Yi	<i>yobi</i>	1 208 925 819 614 629 174 706 176	

Poznámka: 10^k není rovno 2^n , je to jen nejbližší odpovídající mocnina; jsou z ní však odvozeny názvy i značky obdobně desítkovým předponám.

Zapište 256 MiB pomocí mocniny ✓
na ✓ a určete, zda se jedná o bit nebo byte:

- bitu
 bytu ✓

Posun v měřítku:

Kolik je 512 v měřítku 2^4 ? Hodnota je ✓

$$512 = 2^9 = 2^{9-4} = 32$$

Kolik je 256 v měřítku 2^4 ? Hodnota je ✗

$$256 = 2^8 = 2^{(8-4)} = 16$$

Věty o boolean funkcích + Realizace boolean funkce:

- Úplná Boolean funkce je zobrazení $\{0,1\}^n$ do $\{0,1\}$
- Zápis Boolean funkce $f(x_{n-1}\dots x_0) = \text{IIM}(1, 2, 9\dots)$ jednoznačně definuje:
 - úplnou Boolean funkci.
- Zápis Boolean funkce $f(x_{n-1}\dots x_0) = \text{II M}(1,2,9\dots) + \text{II D}(4,5\dots)$ 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:

Zápis do mapy, provedeme minimalizaci:

$\pi M = 0, \pi D = X;$

(Smyčky z 0)

$\Sigma m = 1, \Sigma d = X;$

(Smyčky z 1)

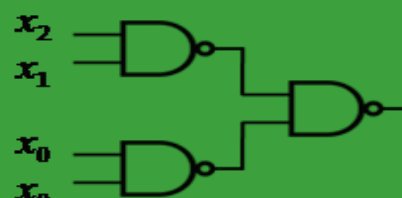
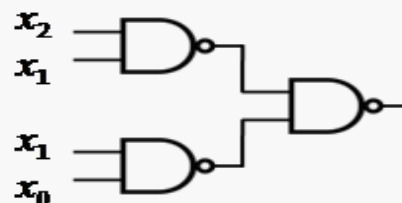
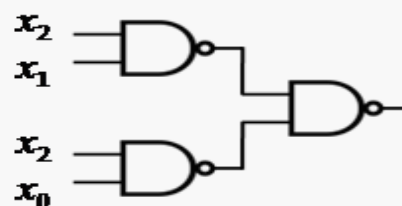
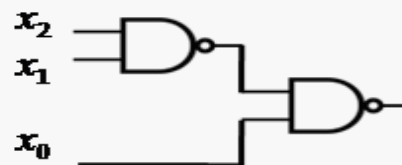
Boolean funkce je zadána tvarem

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

Vyberte odpovídající realizaci Boolean funkce.

Záporné body.

Vyberte jednu z nabízených možností:



Posunutí / Přímý kod / dvojkový doplněk

Je dána skupina o 4 bitech s hodnotou 0000.

Skupina při zápisu čísla bez znaménka je dekadické číslo ✓

Skupina při zápisu čísla v typickém posunutí představuje dekadické číslo ✗

Typické posunutí:

podle vzorce $2^{(n-1)} - 1$ kde n je počet bitů. Mělo by to být takhle, $2^{(4-1)} - 1 \rightarrow 2^3 - 1 \rightarrow 7$ Takže $13-7=6$

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

Je dána skupina o 4 bitech s hodnotou 1001.

Skupina při zápisu čísla bez znaménka je dekadické číslo ✓

Skupina při zápisu čísla se znaménkem a amplitudou (v přímém kódu) představuje dekadické číslo ✓

Skupina při zápisu čísla se znaménkem a amplitudou

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

Je dána skupina o 4 bitech s hodnotou 1011.

Skupina při zápisu čísla bez znaménka je dekadické číslo ✓

Skupina při zápisu čísla v dvojkovém doplňku představuje dekadické číslo ✓

Skupina při zápisu čísla v dvojkovém doplňku představuje dekadické číslo:

To by bylo -3, protože $1101 \rightarrow$ (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. (zneguješ a přičteš 1)
 - pokud máme kladné, neprovádíme negaci ani přičítání!!
- Podle zadaného formátu upravíme výsledek Q3.3

Úloha 2

Správně

Bodů 5,00 / 5,00

Úloha s
vlaječkou

Je daný byte a jeho hodnotu lze číst jako číslo s pevnou řádovou čárkou v Qm.f formátu.

Hodnotu bytu převedte na reálné desítkové číslo. Použijte desetinou tečku.

Hodnota bytu je 0xE6, formát je Q3.3.

Reálné číslo je



Je daný byte a jeho hodnotu lze číst jako číslo s pevnou řádovou čárkou v Qm.f formátu.

Hodnotu bytu převedte na reálné desítkové číslo. Použijte desetinou tečku.

Hodnota bytu je 0x1C, formát je Q3.3.

Reálné číslo je D

(radix tečka)

Teorie o Qm.f

• m značí počet bitů měřítka

Vyberte jednu z nabízených možností:

Pravda.

Nepravda. ✓

• $m + f + 1$ značí počet bitů čísla bez znaménka

Vyberte jednu z nabízených možností:

Pravda. ✗

Nepravda. ✓

Pro zobrazení binárních čísel v pevné řádové čárce se používá formát Qm.f

kde

- f značí počet bitů zlomkové části

Vyberte jednu z nabízených možností:

Pravda. ✓

Nepravda.

- Q značí množinu racionálních čísel Q

- $m + f + 1$ značí vyplnění počtu bitů do násobku 8

Vyberte jednu z nabízených možností:

Pravda. ✓

Nepravda. ✗

Vyberte jednu z nabízených možností:

Pravda.

Nepravda. ✓

- f značí počet bitů integer části

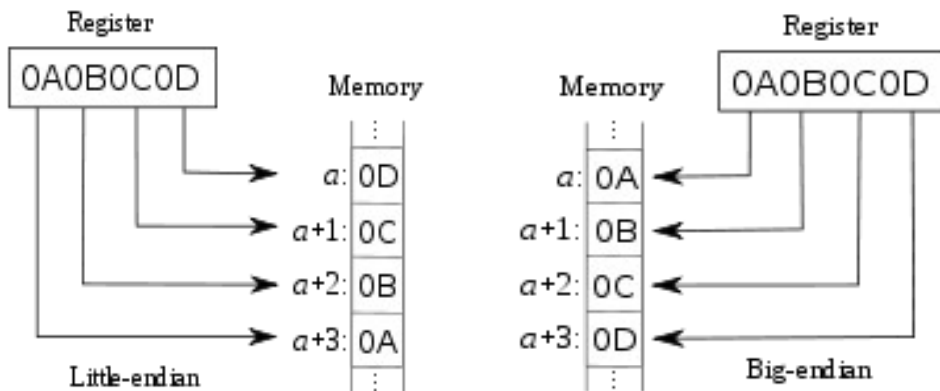
Vyberte jednu z nabízených možností:

Pravda.

Nepravda. ✓

Little a Big Endian (Endianita)

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



Malý endian (little endian) je definován tak, že MSB element je pamatován na vyšší adrese, viz obrázek, kde je patrná zvyšující se adresa.

Do paměti se číslo ukládá po bytech, tzn. že základní element je Byte. Niže uvedený sled bytů zapište do paměti podle principu little endian. Do nepoužitých adres použijte kombinaci 0000.

Adresa	89	F1	00	00
i+3				
i+2				
i+1				
i				

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.

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

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

Adresa	14	8C	2D	9A
i+3				
i+2				
i+1				
i				

Word memory

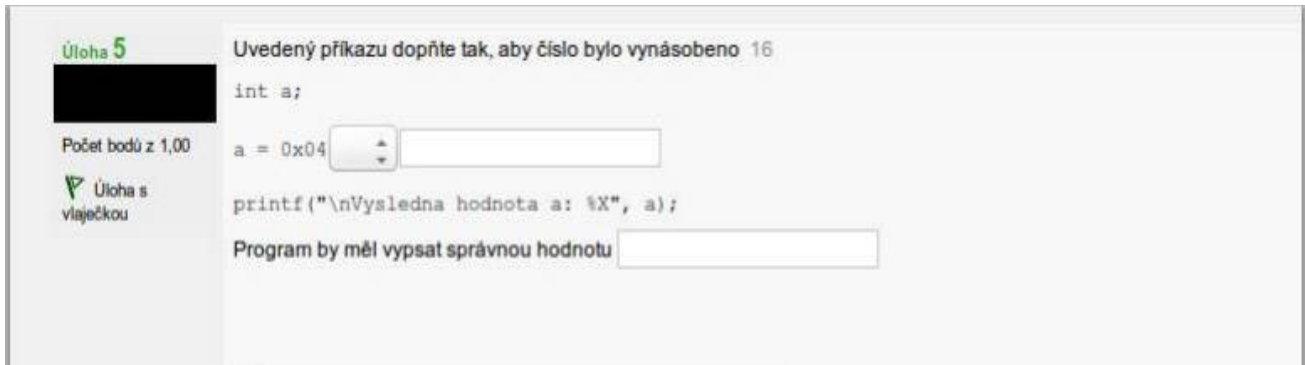
0x08090A0B0C0D0E0F

Address	a	0809
	a+1	0A0B
	a+2	0C0D
	a+3	0E0F

MSB ← → LSB

↓ Increasing addresses

Zápis násobení/dělení v Cčku



The screenshot shows a task titled "Úloha 5" with a point value of 1.00. The task description asks to complete a C++ program so that the number 16 is multiplied. The code provided is:

```
int a;  
a = 0x04  
printf("\nVysledna hodnota a: %X", a);
```

There is a numeric input field next to "a = 0x04" and a text input field for the expected output. The text below the code says "Program by měl vypsát správnou hodnotu".

První ListBox:

Násobení: << (posun doleva)

Dělení: >> (posun doprava)

První Textbox:

Mocnina čísla (16) = 4

Druhý Textbox:

Hexa (0x04) převedeme a vynásobíme/vydělíme dle zadání pomocí hodnoty v prvním Textboxu.

IBM Packet Format

Úloha 11

Počet bodů z 1,00

Úloha s vlajčkou

Společnost IBM používá zhuštěný formát (packet format) se znaménkem.

Níže uvedené číslo zapište ve zhuštěném formátu po bytech. Pro nepoužité byty či nibble použijte hodnotu nula.

+5836 → 0x 0x 0x

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 nibble (na poslední pozici) písmeno **C**, pokud bude zaporné píšeme **D**
3. Převádíme jednotlivé čísla BCD a zapisujeme do jednotlivých boxu ve formátu 0x05 0x83 0x6C

Úloha 2
Dotud nezodpovězeno
Počet bodů z 1,00
Úloha s vlajčkou

Společnost IBM používá zhuštěný formát (packet format) se znaménkem.
Níže uvedené číslo zapište ve zhuštěném formátu po bytech. Pro nepoužité byty či nibble použijte hodnotu nula.

+1472 → 0x 0x 0x

01100010 0x

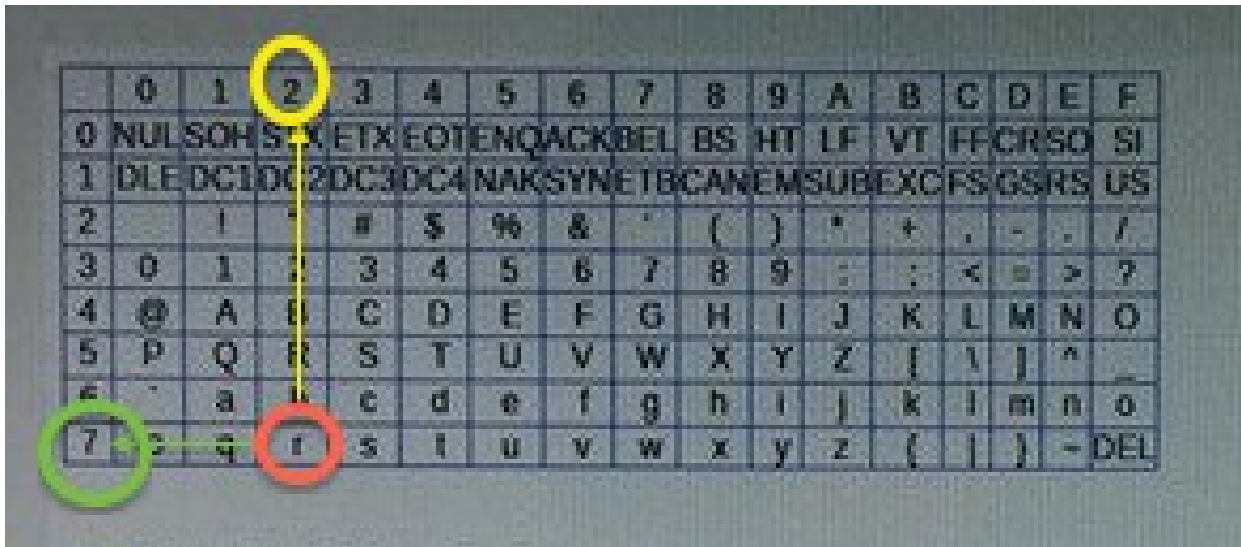
C

0x01 0x47 0x2C

C = kladne

D = zaporne

UTF8 Kodování textu



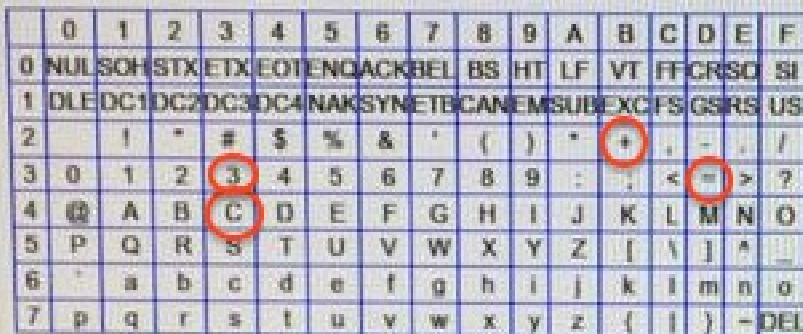
The image shows an ASCII table with several characters and indices highlighted. A yellow circle highlights the index '2' in the top row. A green circle highlights the index '7' in the bottom row. A red circle highlights the character 'r' in the bottom row, column '3'. A yellow arrow points from the '2' index down to the 'r' character.

	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 příslušných boxů



The image shows an ASCII table with several characters and indices highlighted in red circles. The index '3' is highlighted in the top row, column '3'. The character 'C' is highlighted in the top row, column '3'. The character 'r' is highlighted in the bottom row, column '3'. The character '+' is highlighted in the top row, column '10'. The character '-' is highlighted in the top row, column '14'.

	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

Zakódujte následující text: C+=3

0x 43

0x 2B

0x 3D

0x 33

Větvení podle bitu/Znulování/Negace/Nastavení

Je daný registr podle obrázku. Registr lze číst jako integer číslo, kde LSB odpovídá váze 2^0 .

0	1	0	1	0	1	0	1
MSB							LSB

V programu je požadavek na větvení podle bitu na 5. pozici.
Doplníte příkaz podmíněného větvení podle uvedeného požadavku

if (reg &)

- 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 zápisem třeba znegovat
- DO prvního boxu zapisujeme bitový AND (&).

Negace (flip):

- DO prvního boxu zapisujeme bitový XOR (^).

Nastavení:

- DO prvního boxu zapisujeme bitový OR

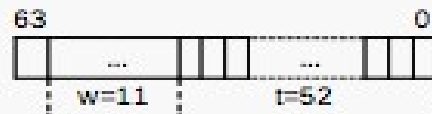
Hodnota posunutí (BIAS) - Float 754(2008)

Úloha 1

Dotaz
nezodpovězeno

Počet bodů z
1,00

Úloha s
vláječkou



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

Sestavení UTF-8 formátu

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

Pro kódovou pozici U+0741 sestavte UTF-8 formát

0x 0x

- Převédeme 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. → (**110x xxxx**) a (**10xx xxxx**)
- Do continuation bytu (10xx xxxx) zapisujeme na pozice x prvních 6 bitů z prava (od zadu) → (**1000 0001**)
- Do leading bytu (110x xxxx) pokračujeme se zapisováním s dalšíma 5 bitama čísla → (**1101 1101**)
- Do prvního boxu zapisujeme leading bit v hexu, do dalších boxů zapisujeme continuation byty, také v hexu.

Opačný převod:

Převédeme z hexu na bináru, odstraníme leading/continuation předpisy a jednotlivě převadíme zpátky na hex do určeného předpisu.

Pro zápis ve formátu UTF-8 sestavte kódovou pozici v Unicode.
Správná kódová pozice je na 4. pozici.

UTF-8 : 0xDC 0xAE → U+ X

111 0010 1110
(7) (2) (1)
U+0721



Definice polí pro výměnný binární formát čísel ve floating point

Úloha 11
Dosud nezodpovězeno
Počet bodů z 4,00
Úloha s vlaječkou

Definice polí pro výměnný binární formát čísel v pohyblivé řádové čárce (binary floating-point format) podle IEEE 754-2008 je

15b4 10b9 0

0	1	0	0	0	1	1	1	1	0	0	0	0	0	0	0
---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---

Uvedený 16 bitový binární formát čísla v pohyblivé řádové čárce představuje číslo.

Hodnota čísla je

$\times 2^{\text{B}}$

$\times 2^{\text{b}}$

Odpovídejte ve formátu znaménko (+/-), significant (bin), základ 2 je uvedený a mocnina (dec).
Pokud je nutná k zápisu radix čárka, použijte radix tečku.

◀ || ▶

15b4 10b9 0

0	1	0	0	0	1	1	1	1	0	0	0	0	0	0	0
---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---

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 opišeme **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 rozdílu, posunutí

Úloha 7
Dosud nezodpovězeno
Počet bodů z 4,00
Úloha s vlaječkou

Operace rozdílu při zobrazení čísel pomocí posunutí je dána vzorcem

$$R^b = A^b + (\sim B^b) + b + 1$$

Vypočítejte rozdíl dvou čísel v posunutí, kde hodnota posunutí je $b=7$.

Čísla jsou $a = -2$, $b = 3$. Všechny zadání čísel a operace se provádějí v binárním formátu a 4 bitech.


operand A^b B()

operand $(\sim B^b)$ B()

binární součet B()

$b + 1$ B()

Výsledek R^b B()



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 $A^b = (-2+7) \rightarrow 5 \rightarrow 0101$

Operand $\text{neg-}B^b =$ Zde musíme provést ještě bitovou negaci. $3 \rightarrow 0011 \rightarrow 1100 + 0111 = 19 (10011)$ – Musíme však zapisovat ve 4 bitech!
 $\rightarrow 0011$.

(Zde se negace vyskytuje náhodně dle zadání. Nemusí být nutně u operandu B^b)

Binární součet = $(A^b + B^b) \rightarrow 5 + 19 \rightarrow 11001 \rightarrow$ zapisujeme na 4 bity
 $\rightarrow 1001$

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

Výsledek $(R^b) =$ (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 O_m :

- udává počet číslic zlomkové části čísla

pro O_n

- udává počet číslic celočíselné část čísla

Příznaky N/Z/V/C

Jsou dány čísla a a b . Vypočítejte 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 jednoduše zjistit jako XOR carry Z a do MSB.

Přetečení pro čísla ve dvojkovém doplňku.

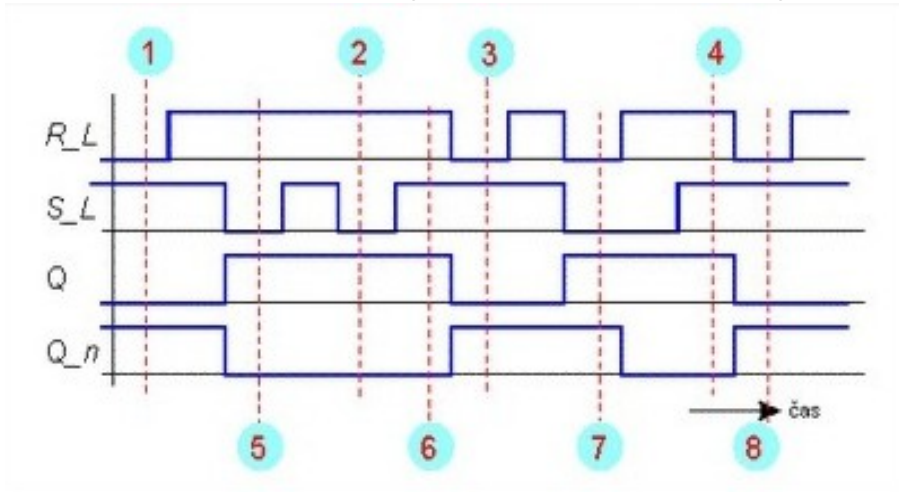
Příznak C značí:

Vyberte jednu z nabízených možností:

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

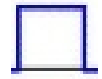
C = Carry bit Zbude nam zbytek po operaci. (1 kterou už nelze zapsat).

Časový průběh RS klopných obvodů

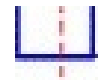


Všimáme si výstupu na R_L, S_L (R,S)

1 = vyvýšená



0 = na čáře

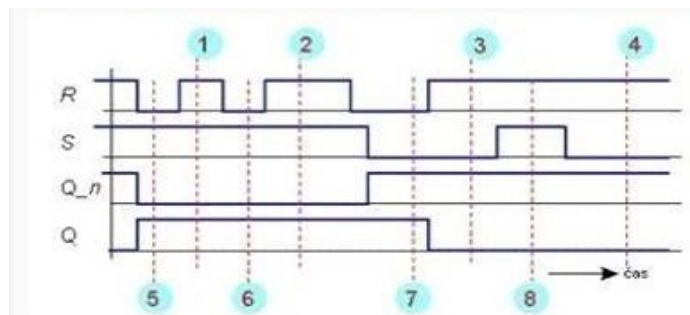
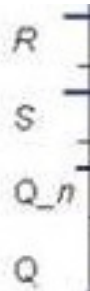


Čísla v modrem jsou ty časové body :)

NOR

(Rozhoduje logická 1)

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

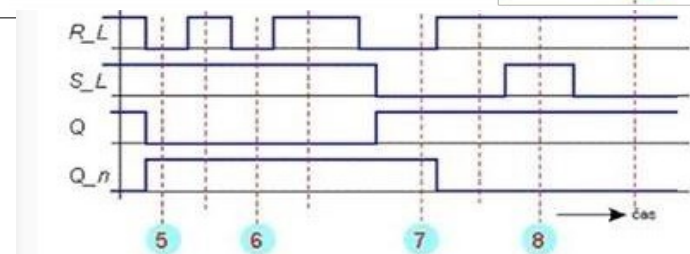
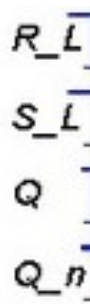


- V časovém bodu 1 je akce
- V časovém bodu 2 je akce
- V časovém bodu 3 je akce
- V časovém bodu 4 je akce
- V časovém bodu 5 je akce
- V časovém bodu 6 je akce
- V časovém bodu 7 je akce
- V časovém bodu 8 je akce

NAND

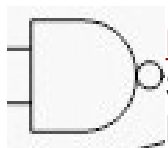
(Rozhoduje logická 0)

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



- V časovém bodu 1 je akce
- V časovém bodu 2 je akce
- V časovém bodu 3 je akce
- V časovém bodu 4 je akce
- V časovém bodu 5 je akce
- V časovém bodu 6 je akce
- V časovém bodu 7 je akce
- V časovém bodu 8 je akce

Paměťové elementy



RS NAND:

$S_L \rightarrow Q$

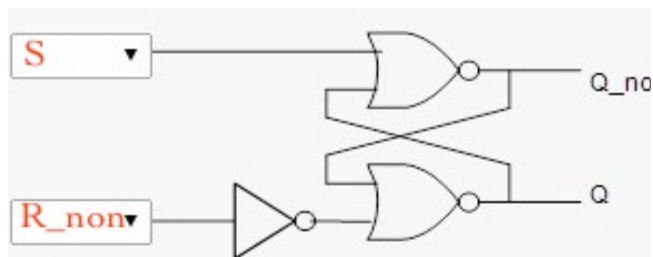
$R_L \rightarrow Q_L (Q_{non}, Q_{Bar})$



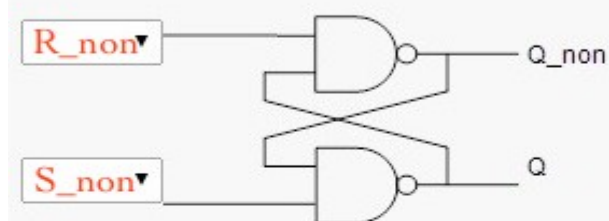
RS NOR:

$R \rightarrow Q$

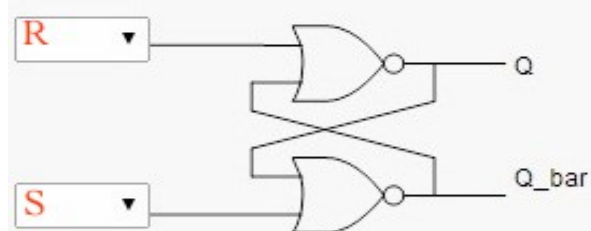
$S \rightarrow Q_L (Q_{non}, Q_{Bar})$



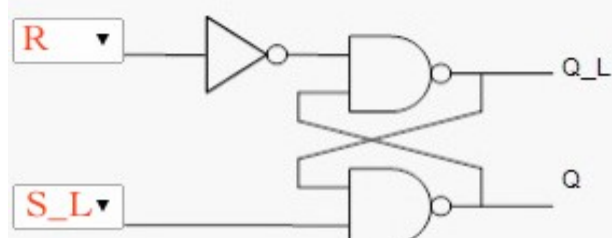
Na obrázku je zapojení základního paměťového elementu



Na obrázku je zapojení základního paměťového elementu



Na obrázku je zapojení základního paměťového elementu



Zaokrouhlování čísel

Je dané číslo -2.7450.

V odpovědích používejte desetinou tečku.

Uvedené číslo zokrouhlete na dvě desetinná místa podle:

- směrem k nejbližší, k sudé

- směrem k nejbližší, od nuly

- směrem k nule

- směrem k plus nekonečnu

- směrem k minus nekonečnu

Je dané číslo 2.7450.

V odpovědích používejte desetinou tečku.

Uvedené číslo zokrouhlete na dvě desetinná místa podle:

- směrem k nejbližší, k sudé

- směrem k nejbližší, od nuly

- směrem k nule

- směrem k plus nekonečnu

- směrem k minus nekonečnu

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 (nahoru) → 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 (nahoru) → 2.34)

směrem k nule

Zaokrouhlení na číslo, které je blíže 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

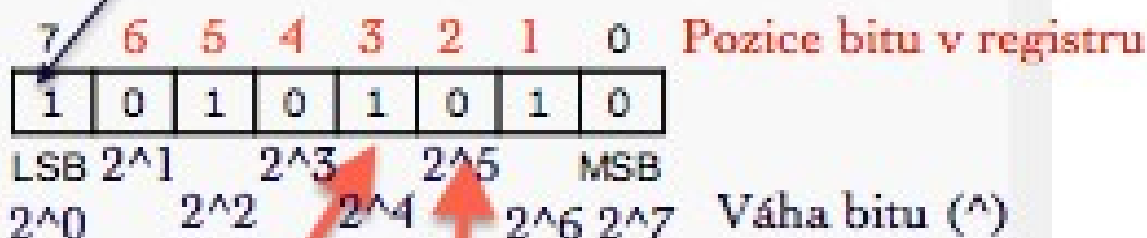
e) - 5.560

vše: - 5.56

MSB/LSB

Je zadaný byte, který má vyznačené pozice a váhy.

Tato skupina představuje integer číslo, kde LSB odpovídá 2^0 .



Jaká je hodnota bitu na 2. pozici

0

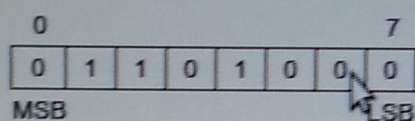
Jaká je hodnota bitu s váhou 2^4

1

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

Je zadaný byte, který má vyznačené pozice a váhy.

Tato skupina představuje integer číslo, kde LSB odpovídá 2^0 .



Jaká je hodnota bitu na 1. pozici

1

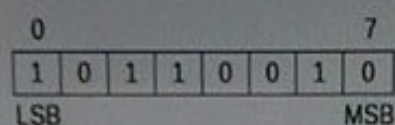
Jaká je hodnota bitu s váhou 2^6

0

1

Je zadaný byte, který má vyznačené pozice a váhy.

Tato skupina představuje integer číslo, kde LSB odpovídá 2^0 .



Jaká je hodnota bitu na 1. pozici

0

Jaká je hodnota bitu s váhou 2^3

1

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 significant 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)

Booleova Algebra - Pravidla

Summary of Boolean algebra

- $1 + 1 = 1$
- $1 + 0 = 1$
- $!1 = 0$
- $!(!a) = a$
- $a + a + \dots = a$
- $1 + a = 1$
- $a + !a = 1$
- $1 + \text{anything} = 1$
- $\text{anything} + !(anything) = 1$
- $1 \cdot 1 = 1$
- $1 \cdot 0 = 0$
- $!0 = 1$
- $a \cdot a \cdot \dots = a$
- $0 \cdot a = 0$
- $a \cdot !a = 0$
- $0 \cdot \text{anything} = 0$
- $\text{anything} \cdot !(anything) = 0$

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

Minterm / Maxterm

Sestavte pro vybraný řádek minterm

Pravdivostní tabulka

c	b	a
0	0	0
0	0	1
0	1	0
0	1	1
1	0	0
1	0	1
1	1	0
1	1	1

c' AND b AND a

minterm/maxterm se provádí pro určitý řádek tabulky

minterm -

kde je 1, píšem proměnnou

kde je 0, píšem její negaci

mezi nimi AND

maxterm

kde je 0, píšem proměnnou

kde je 1, píšem její negaci

mezi nimi OR

c	b	a
0	0	0
0	0	1
0	1	0
0	1	1
1	0	0
1	0	1
1	1	0
1	1	1

c AND b AND a

Ostatní (teoretické) otázky

Binární sčítačka:

Úloha 5
Nesprávně
Bodů 0,00 / 1,00
Úloha s vlajčkou

Realizace binární sčítačky pomocí kanálu zrychleného přenosu zajišťuje

Vyberte jednu z nabízených možností:

- Výpočet rozdílu.
- Zmenšení počtu hradel při realizaci. ✘
- Zmenšení zpoždění, za který je součet vykonán.
- Výpočet dvojkového doplnku.

Hradla XOR (První Test)

Zapište výsledek výrazu,

$0 \oplus 0 \oplus 1 \oplus 0 \oplus 1 \oplus 1 \oplus 0 \oplus 1 =$ ✘

UTF Teorie (Test 7)

Současná verze Unicode používá v UTF-8 název vedoucí byte (leading byte), potom počet vedoucích jedniček vedoucího bytu určuje celkový počet bytů sekvence UTF-8.

Vyberte jednu z nabízených možností:

- Pravda.
- Nepravda.

Současná verze Unicode používá volitelný kód U+FEFF jako BOM - Byte Order Mark, potom sekvence bytů EF BB BF odpovídá UTF-16.

Vyberte jednu z nabízených možností:

- Pravda.
- Nepravda.

Signály ? (Test7) - Nepotvrzené výsledky

Signál "NASOB_L" je aktivní, když má hodnotu logickou hodnotu 1.

Vyberte jednu z nabízených možností:

- Pravda.
- Nepravda.

Signál "DOLU_non" je aktivní, když má hodnotu logickou úroveň L.

Vyberte jednu z nabízených možností:

- Pravda.
- Nepravda.

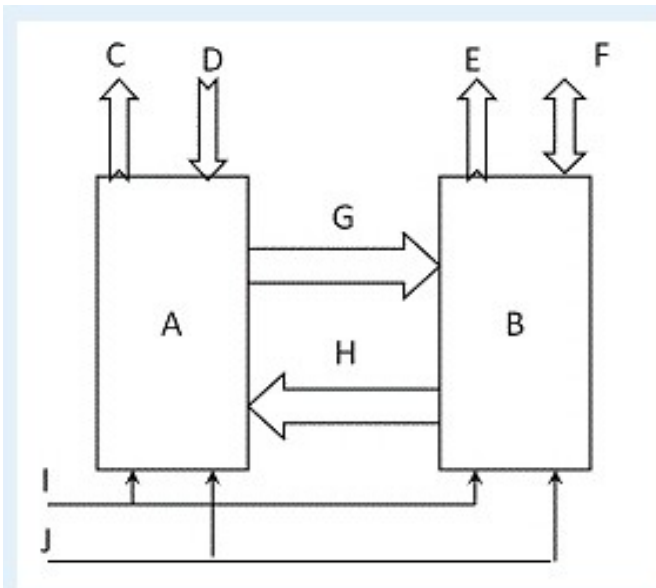
Demultiplexer

Co to je demultiplexer?

Select one:

- Demultiplexer vybírá jeden ze vstupů, z kterého se bude číst informace.
- Demultiplexer převádí jeden vstup do jednoho z více výstupů na základě výběrového signálu. ✔

Synchronní číslicové systémy



Na výše uvedeném obrázku je blokové schéma synchronního číslicového systému.

Písmenu A odpovídá **X**

Písmenu D odpovídá **✓**

Písmenu H odpovídá **X**

Řadič A

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

Datová Jednotka B

- Přijímá i odesílá data (datový vstup/výstup) (*F*)
- Odesílá výstup příznaku (*E*)
- Přijímá Signál řízení od „Řídící jednotky“ (*G*)
- Odesílá stavový výstup do „Řídící jednotky“ (*H*)

A: řadič (řídící jednotka)

B: Datová jednotka (řízený objekt)

C+E: příznaky

D: příkazy

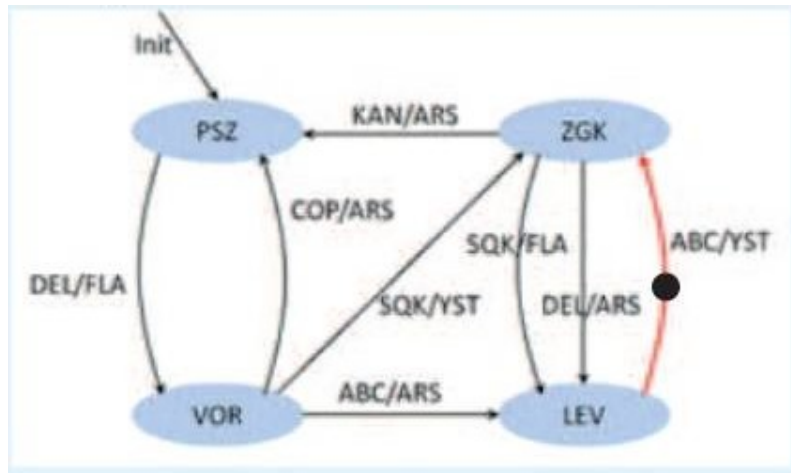
F: datové vstupy, výstupy

G: signály řízení (Řídící signály)

H: podmínky, stavy

Finite State Machine - Automaty Končených Stavů (FSM)

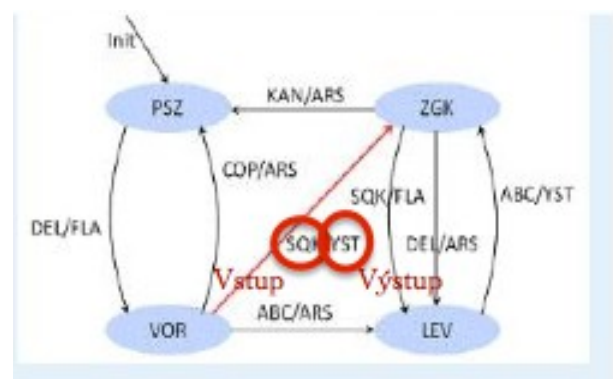
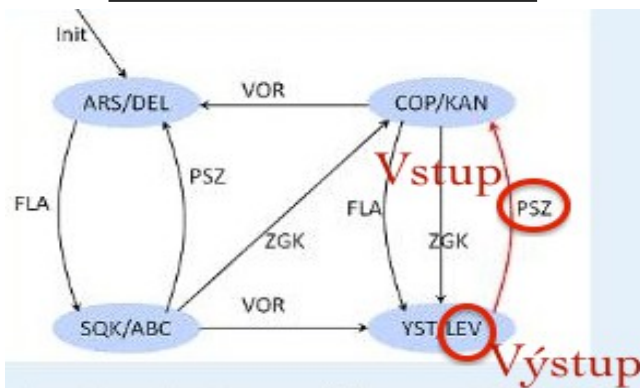
Ukazkový příklad:



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

FSM se nachází v **Současném** ✓ stavu **LEV**

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

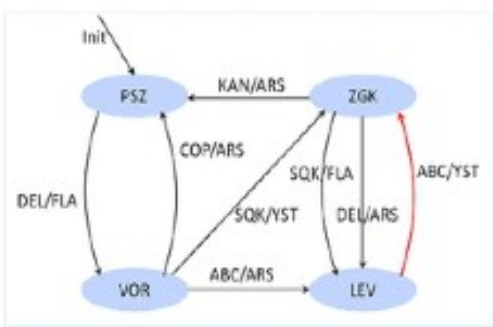


- V případě na ukázkovém příkladě je Současný vstup ABC a výstup YST.

a při **Současném** ✖ vstupu **ABC**
generuje **Současný** ✓ výstup **YST** ✓

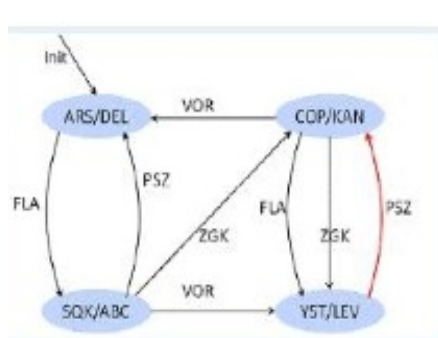
- Poslední úkol je zjistit **následující stav**, ten se nachází v políčku kde **červená šipka** končí.
- Pokud se na konečném políčku vyskytuje více identifikací (např. COP/KAN) zapisujeme ten první (levý).
 - V případě na ukázkovém příkladě končí šipka na poli ZGK

a přechází do **Následujícího** ✓ stavu **ZGK** ✓



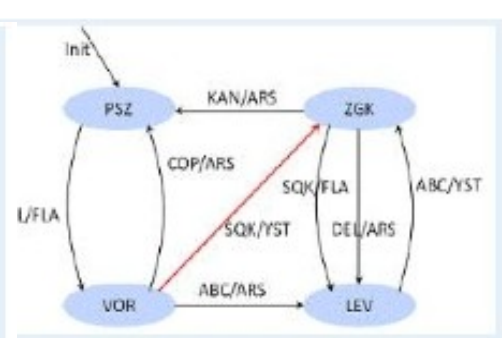
Pro červeně vyznačenou hranu na obrázku sestavte následující větu.

FSM se nachází v stavu **LEV**
 a při vstupu **ABC**
 generuje výstup **YST**
 a přechází do stavu **ZGK**



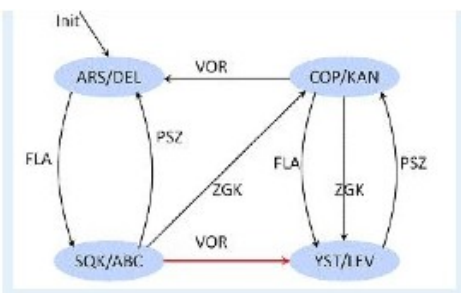
Pro červeně vyznačenou hranu na obrázku sestavte následující větu.

FSM se nachází v stavu **YST**
 a při vstupu **PSZ**
 generuje výstup **LEV**
 a přechází do stavu **COP**



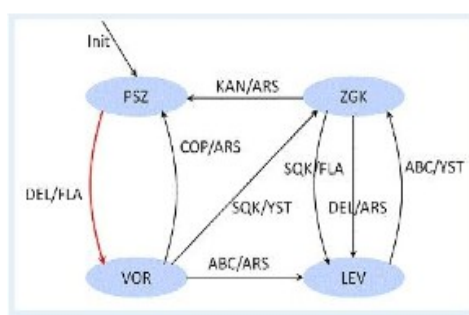
Pro červeně vyznačenou hranu na obrázku sestavte následující větu.

FSM se nachází v stavu **VOR**
 a při vstupu **SOK**
 generuje výstup **YST**
 a přechází do stavu **ZGK**



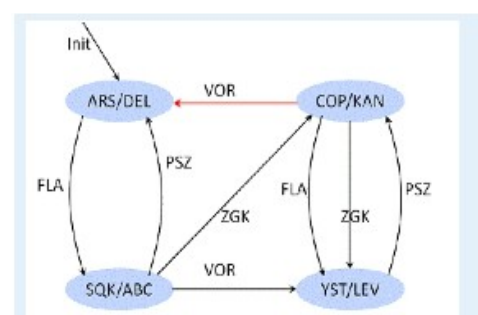
Pro červeně vyznačenou hranu na obrázku sestavte následující větu.

FSM se nachází v stavu **SOK**
 a generuje výstup **ABC**
 a při vstupu **VOR**
 přechází do stavu **YST**



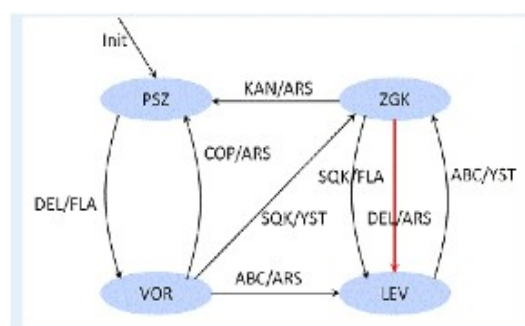
Pro červeně vyznačenou hranu na obrázku sestavte následující větu.

FSM se nachází v stavu **PSZ**
 a při vstupu **DEL**
 generuje výstup **FLA**
 a přechází do stavu **VOR**



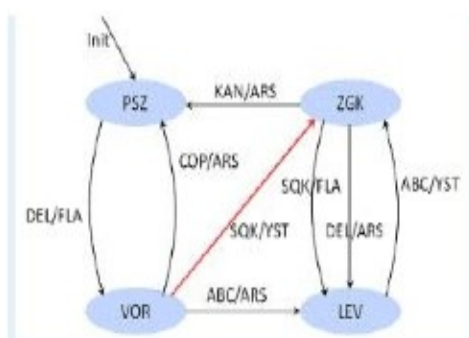
Pro červeně vyznačenou hranu na obrázku sestavte následující větu.

FSM se nachází v stavu **COP**
 a generuje výstup **KAN**
 a při vstupu **VOR**
 přechází do stavu **ARS**



Pro červeně vyznačenou hranu na obrázku sestavte následující větu.

FSM se nachází v stavu **ZGK**
 a při vstupu **DEL**
 generuje výstup **ARS**
 a přechází do stavu **LEV**



Pro červeně vyznačenou hranu na obrázku sestavte následující větu.

FSM se nachází v stavu **VOR**
 a při vstupu **SOK**
 generuje výstup **YST**
 a přechází do stavu **ZGK**