OBSAH

1. SEKVENČNÉ KLOPNÉ OBVODY 2

1.1 ROZDELENIE SEKVENČNÝCH OBVODOV 2
1.2 KLOPNÝ OBVOD RST 2
1.3 KLOPNÝ OBVOD JK 4
1.4 KLOPNÝ OBVOD D 5
1.5 POSUVNÉ REGISTRE 5
1.6 ČÍTAČE IMPULZOV 6

2 KÓDOVANIE INFORMÁCIÍ V POČÍTAČI 9

2.1 KÓDOVANIE ČÍSEL 9
2.2 INVERZNÝ KÓD 11
2.3 DOPLNKOVÝ KÓD 13

3 KÓDOVANIE ZNAKOV 15

4 ARCHITEKTÚRA VON NEUMANNOVSKÉHO POČÍTAČA 16

4.1 ARITMETICKÁ JEDNOTKA 19

1. sekvenčné klopné obvody

Sekvenčné obvody (nazývané aj **sekvenčné automaty**) sú digitálne elektronické obvody, u ktorých závisí stav výstupov okrem aktuálneho stavu vstupov aj od minulého stavu vstupov. Znamená to, že sekvenčné obvody majú pamäťové vlastnosti.

Najjednoduchšie základné sekvenčné obvody sa nazývajú **preklápacie obvody** (bežnejšie, i keď nespisovne, **klopné obvody**)

Časť sekvenčných obvodov je konštruovaná tak, že sa ich výstupy menia, len ak sa mení v niektorom smere jeden zo vstupov, tzv. **hodinový vstup** (angl. clock). Táto reakcia môže byť na nábežnú hranu - **čelo impulzu**(zmena z 0 na 1) alebo dobežnú hranu - **tylo impulzu** (zmena z 1 na 0) hodinového signálu, zriedkavo aj na obe hrany.

Sekvenčné obvody majú obvykle aj vstup pre **reset,** ktorým sa obvod dá uviesť do definovaného (počiatočného) stavu, napr. po pripojení napájacieho napätia.

1.1 Rozdelenie sekvenčných obvodov

Podľa stavov:

- **astabilné**: dva nestabilné stavy, žiaden stabilný. **Astabilný preklápací obvod** nemá žiaden stabilný stav a neustále sa preklápa medzi dvoma nestabilnými stavmi.
- **monostabilné**: jeden stabilný, jeden nestabilný stav. **Monostabilný preklápací obvod** má jeden stabilný stav, z ktorého je možné ho vstupom preklopiť do nestabilného stavu. Obvod sa sám po určitom čase preklopí naspäť do stabilného stavu.
- **bistabilné**: dva stabilné stavy, žiaden nestabilný. **Bistabilný preklápací obvod** (skr. **BKO**) sa môže nachádzať v jednom z dvoch stabilných stavov. Vstupmi obvodu je možné ho medzi týmito stavmi ľubovoľne preklápať.

Podľa existencie synchronizácie:

- ***Synchrónne*** - celý obvod je riadený z jedného zdroja hodinového signálu - jednoduchšie, najčastejšie používané
- ***Asynchrónne*** - nepoužívajú hodiny, reagujú rovno na zmenu vstupu, trochu zložitejšie na návrh ako synchrónne, musia sa brať do úvahy medzistavy pri prepnutí, tzv. *hazardy*.

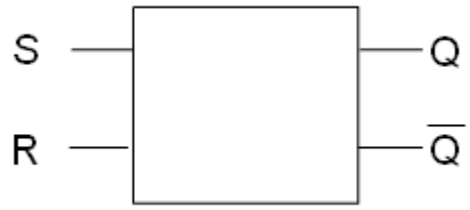
1.2 Klopný obvod RST

Vychádzame z klopného obvodu RS

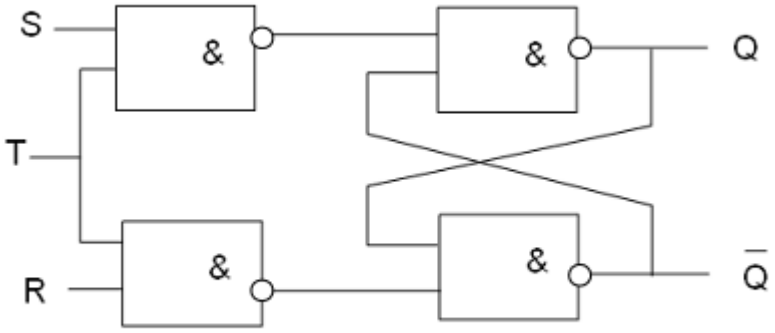
Preklápací obvod RS (angl. *SR latch*) je najjednoduchší asynchrónny bistabilný preklápací obvod. Má dva vstupy: **R** (z angl. *Reset* – nulovanie) a **S** (z angl. *Set* – nastavenie). Uložená hodnota je k dispozícii na výstupe **Q**. Obvykle je k dispozícii tiež negovaný výstup **Q**.

Základný stav oboch vstupov je log.0. V tomto režime si obvod pamätá naposledy nastavenú hodnotu. Privedením log.1 na vstup S sa obvod nastaví (Q = log.1) a vďaka vnútornej spätnej väzbe zostane nastavený aj po návrate vstupu S na log.0. Privedením log.1 na vstup R sa vynuluje (Q = log.0) a rovnako zostane vynulovaný aj po návrate R na log.0. Kombinácia R = S = log.1 sa nazýva *zakázaný* (alebo tiež nestabilný, hazardný) stav, pretože pri ňom nie je definované v akom stave zostane obvod po návrate R a S na log.0.

Preklápací obvod RST (angl. *Gated SR latch*) je *synchrónny* variant obvodu RS. Princíp zostáva zachovaný, avšak k preklopeniu obvodu dochádza len v konkrétnych prípadoch, v závislosti od hodnoty signálu na hodinovom vstupe **T** (time). Obvod RST je synchronizovaný úrovňou (hladinová synchronizácia) hodinového signálu – stav je možné meniť po celú dobu trvania hodinového impulzu.



schematická značka



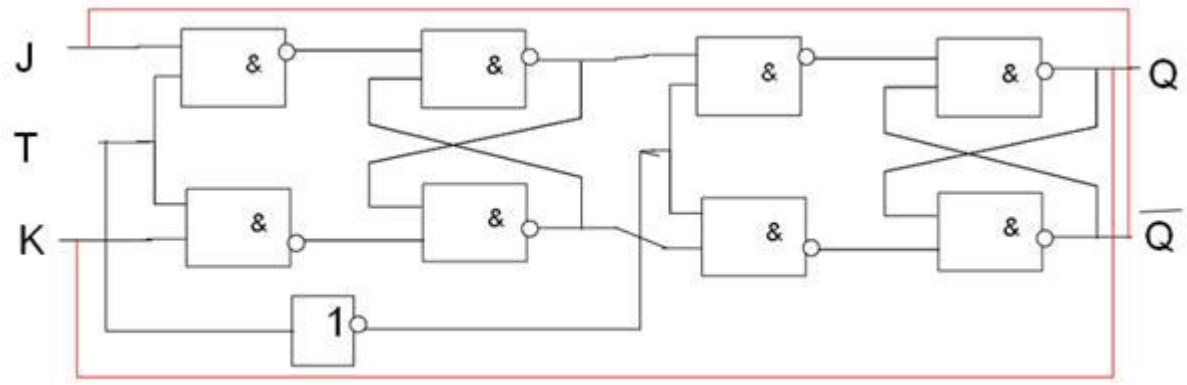
klopný obvod RS

R	S	Q	not Q
0	0	predchádzajúci stav	predchádzajúci stav
0	1	1	0
1	0	0	1
1	1	zakázaný stav	zakázaný stav

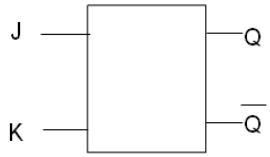
pravdivostná tabuľka

1.3 Klopný obvod JK

Za základ opäť zvolíme klopný obvod RS, pričom použijeme dva obvody prepojené nasledovným spôsobom:



- ***klopný obvod JK***



schematická značka

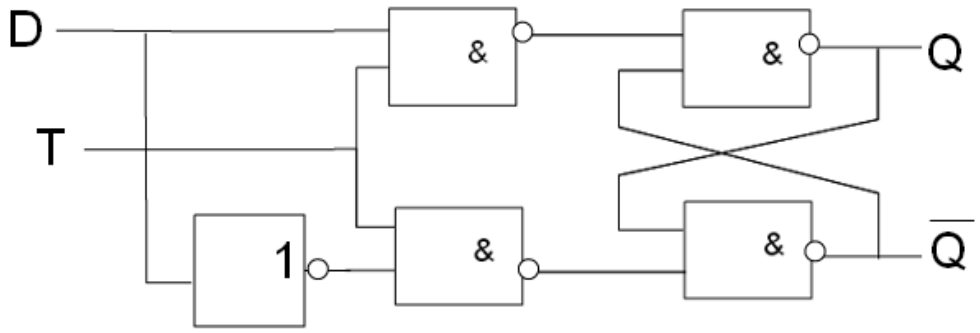
Význam KO JK spočíva v tom, že umožňuje definovať logickú hodnotu aj pre prípad, že obidva vstupy majú logickú hodnotu 1. Zapojením spätnej väzby (vyznačená je červenou farbou) sa dosiahne to, že vstupnej kombinácii 1,1 na vstupoch obvodu bude zodpovedá prevrátenie výstupného obvodu do opačného stavu, ako bol pred príchodom taktovacie impulzu.

J	K	Q
0	0	Predchádzajúci stav
0	1	0
1	0	1
1	1	Predchádzajúci stav

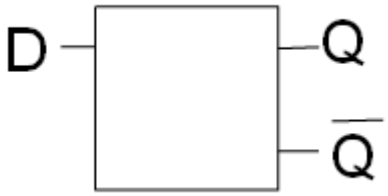
V tomto obvode teda nemôže vzniknúť zakázaný stav

1.4 Klopný obvod D

Za základ opäť zvolíme klopný obvod RS, pričom na vstup privedieme iba jeden z pôvodného obvodu :



- ***klopný obvod D***



schematická značka

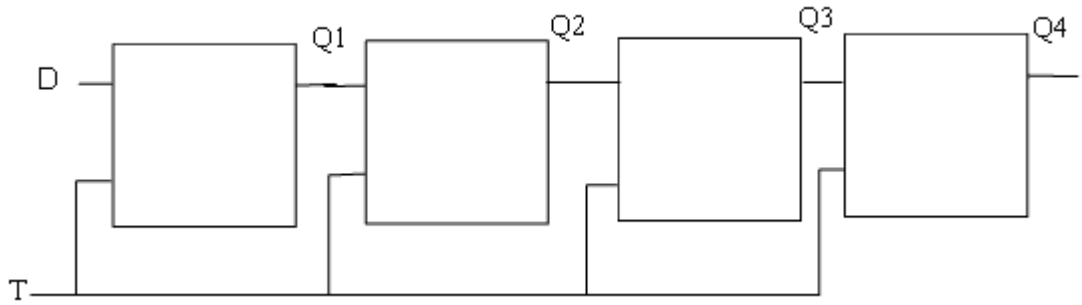
Nedefinovaný stav možno odstrániť v obvode RST zapojením invertora (člena logickej negácie) medzi vstupy RS. Dostaneme tak obvod D, ktorého stav sa prenáša na výstup počas trvania taktovacieho impulzu.

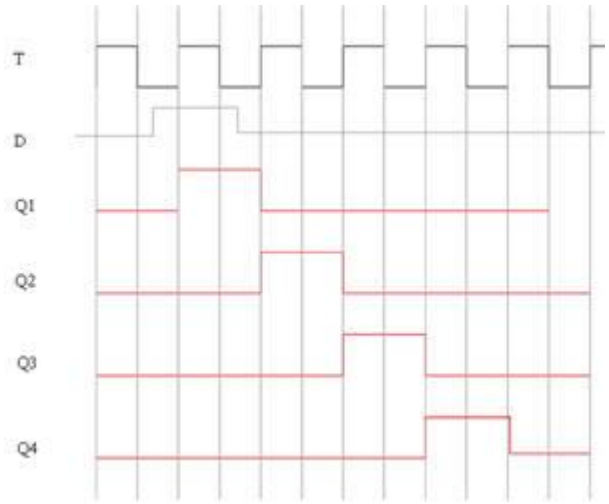
D	Q
1	1
0	O resp. predchádzajúci stav

V tomto obvode teda nemôže vzniknúť zakázaný stav.

1.5 Posuvné registre

Posuvný register je konštruovaný z rady klopných obvodov spojených tak, že každý klopný obvod prenáša informáciu zo svojho vstupu na vstup nasledujúceho klopného obvodu. Samotný presun informácie nastáva vždy s príchodom "čela" impulzu:

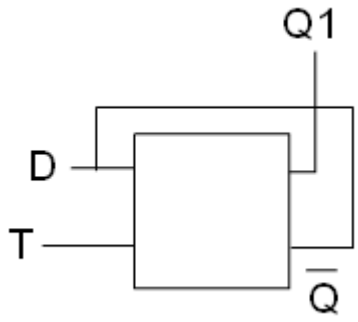


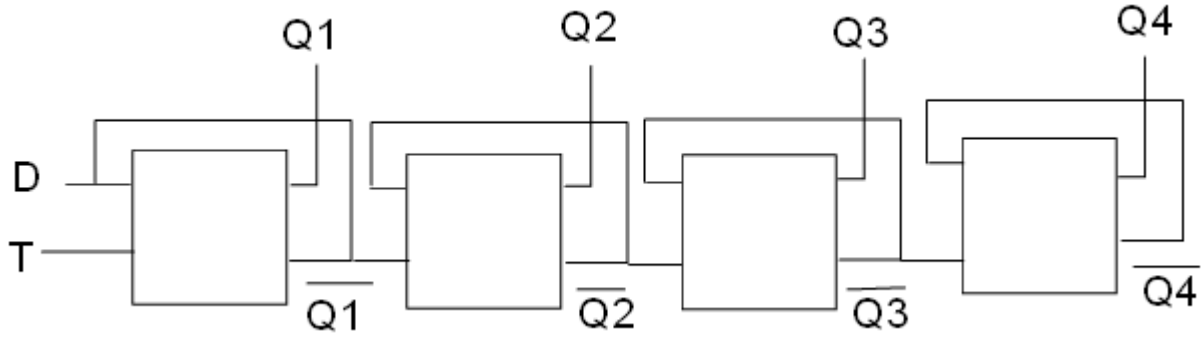


Posuvný register sa môže použiť tam, kde sa má zaznamenať informácia, ktorá prichádza bit po bite v časovom slede. S každým bitom prichádzajúceJ informácie sa musí priviesť na taktovací vstup jeden posúvací impulz. Tak je časový sled bitov zasunutý do registra. Táto sériová informácia sa môže prečítať pri zastavení taktovacích impulzov paralelne z výstupov klopných obvodov. Uplatňuje sa teda posuvný register ako prevodník sériovej informácie na paralelnú.

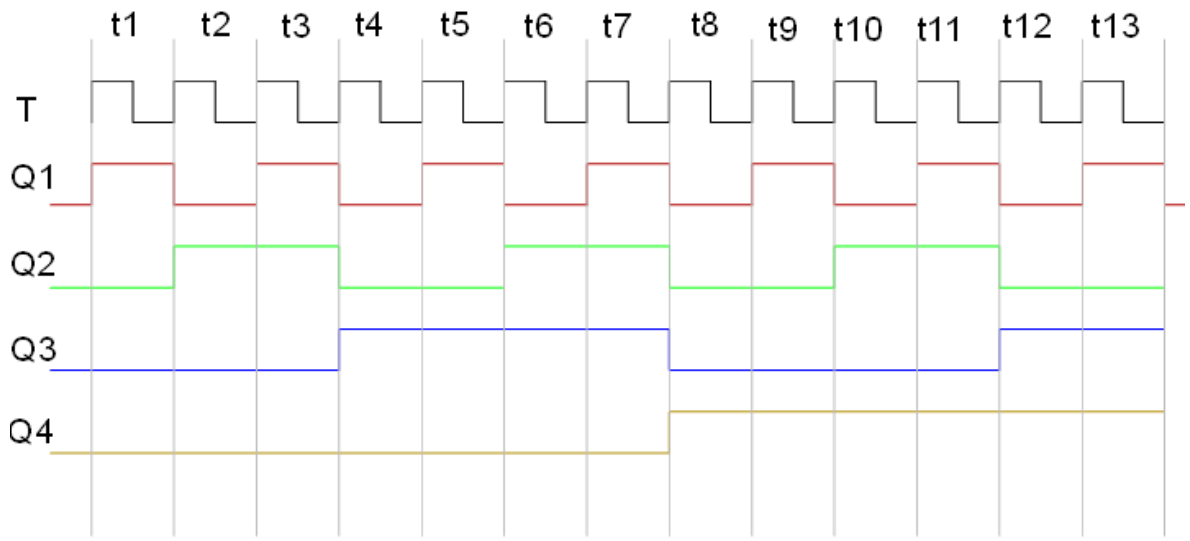
1.6 Čítače impulzov

Zapojíme klopný obvod typu D, kotrý sa spúšťa čelom taktovacieho impulzu. Spojenie výstupu not Q so vstupom D zabezpečí, že s každým taktovacím impulzu sa stav klopného obvodu zmení práve na opačný.





Názorne to ukazuje časový diagram. Vidíme, že frekvencia periodického pravouhlého časového priebehu je na výstupe klopného obvodu polovičná, ako je frekvencia vstupného signálu. Klopný obvod pracuje ako delič impulzovej frekvencie. Ak spojíme viac klopných obvodov tak, že výstup not Q je vždy privedený do nasledujúceho klopného obvodu na svorku T, bude frekvencia impulzov v každom nasledujúcom stupni polovičná oproti frekvencii predchádzajúceho stupňa. Možno povedať, že pôvodná frekvencia sa potom deí dvoma, štyrmi, ôsmimi, a všeobecne 2^n . Zapojenie 4 klopných obvodov a príslušný časový diagram je na obrázku. Ak zapíšeme do tabuľky logické stavy postupne za každým taktovacím impulzom, zistíme, že bity Q1 až Q4 reprezentujú postupnosť dvojkových čísel. Možno teda povedať, že obvod svojimi výstupmi odpočítava impulzy. Zariadenie preto budeme volať dvojkový čítač impulzov.



Q4	Q3	Q2	Q1	N
0	0	0	0	0
0	0	0	1	1
0	0	1	0	2
0	0	1	1	3
0	1	0	0	4
0	1	0	1	5
0	1	1	0	6
0	1	1	1	7
1	0	0	0	8
1	0	0	1	9
1	0	1	0	10
1	0	1	1	11
1	1	0	0	12
1	1	0	1	13
.				
.				
.				

2 Kódovanie informácií v počítači

Kódovanie je proces, pri ktorom sa každému znaku alebo postupnosti znakov daného súboru znakov (vzorov) jednoznačne priradí znak alebo postupnosť znakov (obrazov) z iného súboru znakov.

Kódovanie je teda transformácia určitej informácie z jednej formy na druhú pomocou určitého postupu - algoritmu, ktorý je väčšinou verejne známy. Vo väčšine prípadov teda účelom kódovania nie je utajenie informácie (na rozdiel od **šifrovania**) ale len jej iná forma zápisu vybrané tak, aby sa informácia dala čo najlepšie alebo najúspornejšie uchovať alebo preniesť.

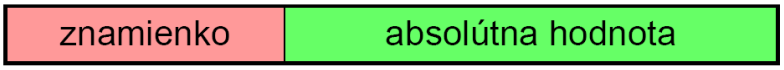
2.1 Kódovanie čísel

Všetky údaje v počítači sú kódované pomocou rôznej kombinácie hodnôt **bitov - najmenšej jednotky informácie**. Každý z bitov môže nadobúdať iba dve rôzne hodnoty 0 a 1. Tieto bity sú však do pamäťových buniek počítača ukladané po osmiciach, preto je výhodné na zakódovanie údajov použiť vždy taký počet bitov, ktorý je deliteľný ôsmimi (8,16,24,32 ....). Čím väčší počet bitov použijeme, tým väčší rozsah čísel môžeme použiť. Napríklad pomocou 8 bitov dostaneme 2^8 = 256 rôznych kombinácií núl a jednotiek. Pomocou 8 bitov teda môžeme zakódovať napríklad čísla od 0 do 255 alebo čísla od -128 do 127 v prípade, že potrebujeme i záporné čísla.

Čísla môžeme v počítači zakódovať 4 spôsobmi:

- **priamy kód**
- **inverzný kód**
- **doplnkový kód**
- **BCD kód**

Priamy kód



znamienko

+ = **0**

– = **1**

$+36_{10} = 0100100_{2PK}$

$-36_{10} = 1100100_{2PK}$

Inverzný kód

rieši problém odčítania spôsobom: 7-3 = 7 + (-3)

tvorí sa z priameho kódu spôsobom:

číslo ≥ 0 inverzný kód = priamy kód

číslo < 0 znamienko ostáva, ostatné bity sa invertujú

$+36_{10} = 0100100_{2PK} = 0100100_{2IK}$

$-36_{10} = 1100100_{2PK} = 1011011_{2IK}$

nevýhoda: problém prenosu pri operácii +

Doplnkový kód

rieši problém odčítania spôsobom: 7-3 = 7 + (-3)
odstraňuje problém prenosu pri inverznom kóde (zanedbáva ho)

tvorí sa z priameho kódu spôsobom:

číslo ≥ 0 doplnkový kód = priamy kód

číslo < 0 znamienko ostáva, po prvú jednotku sprava vrátane sa opisuje, ostatné bity sa invertujú

$+36_{10} = 0100100_{2PK} = 0100100_{2DK}$

$-36_{10} = 1100100_{2PK} = 1011100_{2DK}$

BCD kód

určený na rýchly prevod medzi 2-sústavou a 10-sústavou

tvorí sa priamym prepisom každej číslice zápisu čísla v 10-sústave do 2-sústavy použitím 4 bitov/číslicu

$36_{10} = 00110110_{BCD}$

BCD kód

0 = 0000	4 = 0100	8 = 1000	12 (C) = 1100
1 = 0001	5 = 0101	9 = 1001	13 (D) = 1101
2 = 0010	6 = 0110	10 (A) = 1010	14 (E) = 1110
3 = 0011	7 = 0111	11 (B) = 1011	15 (F) = 1111

tabuľka prevodu použiteľná pre BCD

2.2 Inverzný kód

rieši problém odčítania spôsobom: 7-3 = 7 + (-3)

tvorí sa z priameho kódu spôsobom:

číslo ≥ 0 inverzný kód = priamy kód

číslo < 0 znamienko ostáva, ostatné bity sa invertujú

$+36_{10} = 0100100_{2PK} = 0100100_{2IK}$

$-36_{10} = 1100100_{2PK} = 1011011_{2IK}$

Príklad č.1. **Preveďte číslo -37 do inverzného kódu**

:2	
37	1
18	0
9	1
4	0
2	0
1	1

Číslo prevedieme do dvojkovej sústavy

$(37)_{10}=(100101)_2$ (011010) - toto je inverzný kód

Odčítanie v inverznom kóde:

1. Obe čísla si upravíme na rovnaký počet bitov (pripísaním núl zľava)
2. Číslo, so záporným znamienkom prevedieme do inverzného kódu
3. Spočítame obe čísla
4. Ak po spočítaní vznikne prenos tak ho pripočítame k nultému rádu
5. Ak je výsledok kladný (teda kladné číslo bolo väčšie ako záporné) tak je výsledok v priamom kóde
6. Ak je výsledok záporný (teda kladné číslo bolo menšie ako záporné) tak je výsledok v inverznom kóde

Príklad č. 2 **Odčítajte 7 - 3 pomocou inverzného kódu**

Príklad prevedieme sa súčet: 7 + (-3), a obidve čísla upravíme na rovnaký počet bitov (pripísaním núl zľava)

7 - je to kladné číslo, necháme ho v priamom kóde, len prevedieme do dvojkovej sústavy = 0111

-3 je záporné číslo, takže ho prevedieme do inverzného kódu

$(3)_{10}=(0011)_2$ (1100) - toto je inverzný kód

a ideme urobiť súčet:

0111

\+ 1100

10011 - vznikol prenos (1 na začiatku) a preto k výsledku ju musíme pričítať:

0011

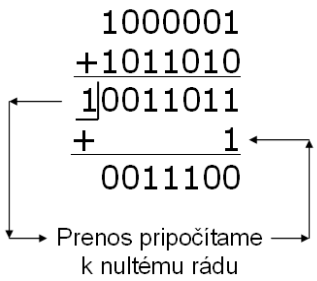
\+ 1

0100 - toto je výsledok po prevode čísla do dvojkovej sústavy bude výsledok 4.

Príklad č. 3 **Odčítajte 65 - 37 pomocou inverzného kódu**

65-37=65+(-37)=???

$(65)_D=(1000001)_B$
$(-37)_D=(-0100101)_B$
$(-0100101)_B=(1011010)_{IK}$



Príklad č. 3 **Odčítajte 37 - 65 pomocou inverzného kódu**

37-65=37+(-65)=???

$(37)_D=(0100101)_B$
$(-65)_D=(-1000001)_B$
$(-1000001)_B=(0111110)_{IK}$

0100101
+0111110
1100011

Výsledok je v inverznom kóde!

2.3 Doplnkový kód

rieši problém odčítania spôsobom: 7-3 = 7 + (-3)

odstraňuje problém prenosu pri inverznom kóde (zanedbáva ho)

tvorí sa z priameho kódu spôsobom:

číslo ≥ 0	doplnkový kód = priamy kód
číslo < 0	znamienko ostáva, po prvú jednotku sprava vrátane sa opisuje, ostatné bity sa invertujú

$+36_{10} = 0100100_{2PK} = 0100100_{2DK}$
$-36_{10} = 1100100_{2PK} = 1011100_{2DK}$

Príklad č.1. **Preveďte číslo -37 do doplnkového kódu**

:2	
37	1
18	0
9	1
4	0
2	0
1	1

Číslo prevedieme do dvojkovej sústavy

$(37)_{10}=(100101)_2$ (011011) - toto je doplnkový kód - vznikne aj tak, ak k inverznému kódu pričítame 1

Odčítanie v doplnkovom kóde

1. Obe čísla si upravíme na rovnaký počet bitov (pripísaním núl zľava)
2. Číslo, so záporným znamienkom prevedieme do doplnkového kódu
3. Spočítame obe čísla
4. Ak po spočítaní vznikne prenos tak ho zanedbáme
5. Ak je výsledok kladný (teda kladné číslo bolo väčšie ako záporné) tak je výsledok v priamom kóde
6. Ak je výsledok záporný (teda kladné číslo bolo menšie ako záporné) tak je výsledok v doplnkovom kóde

Príklad č. 2 **Odčítajte 7 - 3 pomocou inverzného kódu**

Príklad prevedieme sa súčet: 7 + (-3), a obidve čísla upravíme na rovnaký počet bitov (pripísaním núl zľava)

7 - je to kladné číslo, necháme ho v priamom kóde, len prevedieme do dvojkovej sústavy = 0111

-3 je záporné číslo, takže ho prevedieme do doplnkového kódu

$(3)_{10}=(0011)_2$ (1101) - toto je doplnkový kód

a ideme urobiť súčet:

0111

\+ 1101

10100 - vznikol prenos (1 na začiatku) ale ten zanedbáme:

0100 - toto je výsledok po prevode čísla do dvojkovej sústavy bude výsledok 4.

Príklad č. 3 **Odčítajte 65 - 37 pomocou doplnkového kódu**

65-37=65+(-37)=???

$(65)_D=(1000001)_B$
$(-37)_D=(-0100101)_B$
$(-0100101)_B=(1011011)_{DK}$

```
 1000001
+1011011
 0011100

 0011100
```

Prenos zanedbáme

Príklad č. 4 **Odčítajte 37 - 65 pomocou doplnkového kódu**

37-65=37+(-65)=???

$(37)_D=(0100101)_B$
$(-65)_D=(-1000001)_B$
$(-1000001)_B=(0111111)_{DK}$

```
 0100101
+0111111
 1100100
```

Výsledok je v doplnkovom kóde!

3 Kódovanie znakov

Na rozdiel od čísel, znaky textu nevieme previesť do dvojkovej sústavy, preto bolo potrebné vymyslieť iný spôsob ako jednoznačne priradiť určitému znaku práve jednu kombináciu núl a jednotiek, ktorá tento znak v počítači bude reprezentovať. Keďže neexistuje žiadny univerzálny spôsob ako to urobiť, každý výrobca počítačov tento problém riešil iným spôsobom, preto existuje viacero znakových kódov. Poriadok do tohto chaosu sa snažil zaviesť americký úrad pre normalizáciu, ktorý vyhlásil jeden spôsob, ktorý by mali všetci používať. Tento spôsob kódovania sa volá **ASCII - American Standard Code for Information Interchange** (Americký štandardný kód pre výmenu informácií).

Tento štandard hovorí, že na zakódovanie každého znaku sa použije 8 bitov. Čo umožňuje definovať kód pre 256 znakov. Pričom prvá polovica znakov bude pre všetky krajiny rovnaká a zvyšných 128 znakov sa pre každú krajinu stanovil podľa ich potrieb. Tento spôsob vniesol do kódovania znakov neuveriteľný chaos. Preto sa vymyslel nový spôsob kódovania **UNICODE**.

Toto kódovanie používa 16 bitov na zakódovanie jedného znaku, čo umožňuje zakódovať 65536 možných znakov. Tento počet znakov umožňuje zakódovať znaky všetkých abecied pomocou jednej medzinárodnej tabuľky. Tento spôsob kódovania používa i kancelársky balík MS Office. Toto kódovanie zabezpečuje, že ten istý znak má rovnaký kód v každej krajine i na každom type počítača.

Nevýhodou tohto kódovania je, že znaky, ktoré sme predtým vedeli zakódovať iba ôsmymi bitmi v kódovaní Unicode, sú kódované 16 bitmi, a teda zaberajú viac pamäte oko kód ASCII. Istým vylepšením tohto kódovania je kódovanie **UTF-8.** V tomto kódovaní je prvých 128 znakov

tabuľky ASCII (tieto sú pre všetky krajiny rovnaké), zakódovaných pomocou 8 bitov a zvyšné znaky sú zakódované 16, 24, 32, 40 až 48 bitmi. Toto kódovanie je výhodné pre americky hovoriace krajiny a krajiny, v ktorých väčšinu znakov textu tvorí prvých 128 znakov tabuľky ASCII.

Štandardizovaný kód ASCII hovorí, že na zakódovanie každého znaku sa použije 8 bitov, čo umožňuje definovať kód pre 256 znakov. Pre Slovensko sa by sa mala na kódovanie znakov textu používať medzinárodná norma **ISO 8859-2**, ktorá sa tiež nazýva Latin 2. Firma Microsoft však tento štandard nepoužíva, používa štandard schválený americkým úradom ANSI (American National Standards Institute), označovaný tiež **windows-1250.**

Napríklad veta "Ema má mamu." je v počítači uložená nasledovne: 69 109 97 32 109 225 32 109 97 109 117

E m a medzera m á medzera m a m u

Veta „Ema má mamu“ je pomocou kódovania UNICODE v počítači uložená nasledovne:

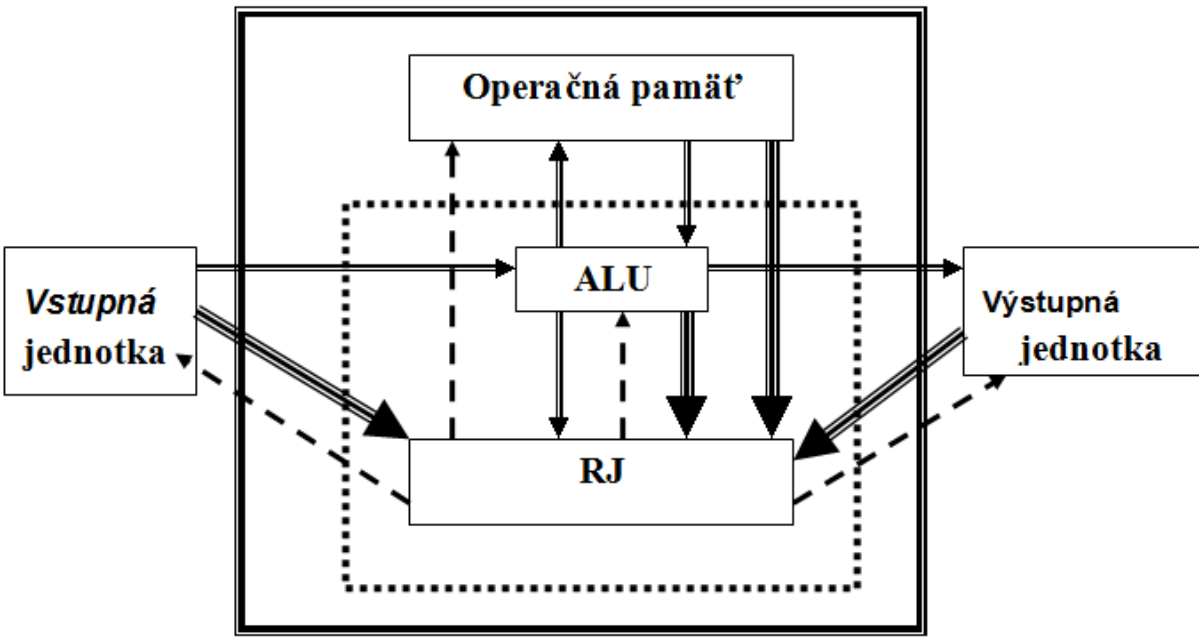
```
69 00 109 00 97 00 32 00 109 00 225 00 32 00 109 00 97 00 109 00 117 00
  E       m       a    medzera      m       á     medzera      m       a    m    u
```

4 Architektúra von Neumannovského počítača

Počítač je zariadenie, ktoré vie spracovávať vstupné informácie a na základe vstupu vykonávajú dopredu definovanú činnosť.
Ak sa pozrieme do „vnútra“ počítača nájdeme tam veľa mechanických, elektrických a elektronických súčiastok, ktoré sú navzájom pospájané a tvoria tak jeden pomerne komplikovaný spolupracujúci celok. V zjednodušenej forme je možné každý počítač chápať ako štruktúru uvedenú na nasledujúcom obrázku.

Základným princípom, na základe ktorého pracujú súčasné počítače je tzv. princíp von Neumanna. Je založený na koncepcii sekvenčného spracovania informácií definovaného existenciou tzv. počítadla (smerníka) vykonávaných inštrukcií a koncepciou operačnej a riadiacej jednotky. Pri riadení práce takto koncipovaného počítača sa využíva koncepcia mikroprogramového riadenia. Činnosť počítača je definovaná programom uloženým v operačnej pamäti. Počítač je vlastne interpretátorom (vykonávateľom) uvedeného programu. Charakteristické vlastnosti pre počítače s von Neumannovou architektúrou by sa dali zhrnúť do nasledovných bodov:

- počítač sa skladá z procesora, pamäte a vstupno – výstupných zariadení
- program je uložený v pamäti počítača
- procesor vykonáva inštrukcie programu postupne
- údaje sa spracúvajú v dvojkovej sústave



Koncepcia Von Neumannovho počítača

Centrálna riadiaca jednotka
Procesor
Riadiace príkazy
Stavová jednotka
Tok údajov

Vstupná jednotka je zariadenie, ktoré do „počítača posiela vstupné informácie, ktoré sa spracovávajú“. Vstupom môže byť klávesnica, myš, mikrofón, kamera, atď.
Centrálna riadiaca jednotka (CPU) je napojená na vstupné a výstupné zariadenia. Je to centrum spracovávania informácií. Skladá sa z troch častí:

- pamäť
- ALU (Aritmeticko – logická jednotka)
- RJ (Riadiaca jednotka)

Pamäť je priestor, kde sa ukladajú spracovávané programy a údaje. Možeme ho označiť aj ako internú pamäť a lebo dočasnú pamäť. Okrem internej pamäte existuje externá pamäť (pamäťové zariadenie), ktorá slúži najmä na ukladanie na trvalé uloženie dát.

ALU je výkonná jednotka, ktorá robí výpočty (matematické alebo logické)

RJ je miesto, ktoré riadi celý proces toku údajov pomocou riadiacich signálov a zodpovedajúcich stavových funkcií. Na každý riadiaci signál odpovedá príslušná časť poslaním stavového signálu.

Spojenie ALU a RJ je označované ako procesor (mikroprocesor).

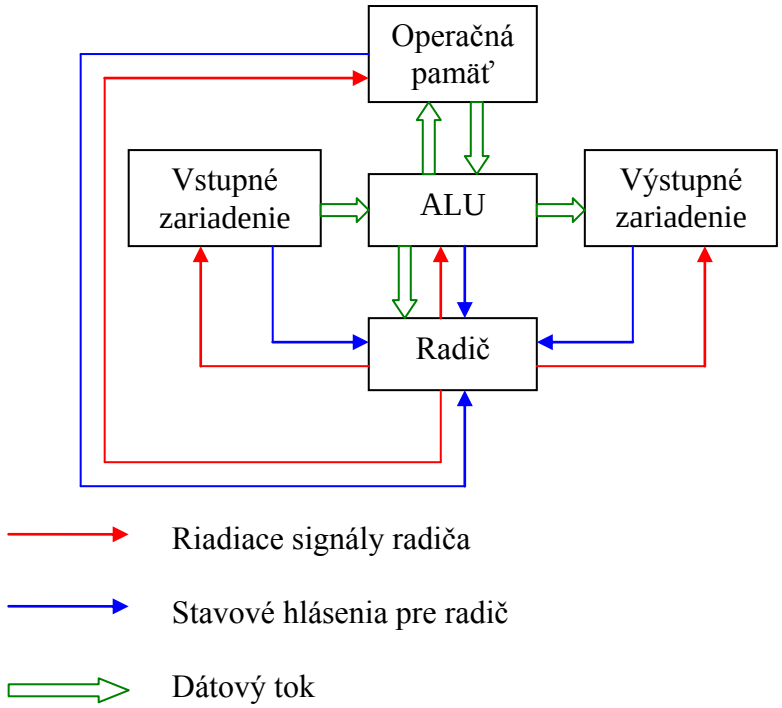
Prepojenie medzi jednotlivými časťami počítača je zabezpečená **zbernicami**. Tieto zbernice majú rôznu šírku a funkciu. Šírka zbernice sa udáva v počte bitov, ktoré zbernica dokáže naraz preniesť (8, 16, 32, 64, ...). Funkcia zbernice je daná miestom a typom informácie, ktorá je po nej prenášaná. Existuje dátová zbernica, adresná zbernica, lokálna (vnútorná zbernica CPU) zbernica, vonkajšia (medzi CPU a perifériami) zbernica.

Činnosť všetkých jednotiek počítača je synchronizovaná hodinami (definujú základný strojový takt procesora).

Princíp počítača

V dnešnom svete plnom techniky sa s počítačmi stretávame na takmer každom kroku. Väčšina z nich, či je to obyčajný stolový počítač, či server, palubný počítač v aute, mobilný telefón alebo PDA zariadenie, všetko sú to počítače, ktoré pracujú na rovnakom princípe, ktorý popísal už v roku 1945 americký matematik narodený v Maďarsku John von Neumann. Podľa tejto teórie, ktorá s menšími obmenami platí dodnes, sa bloková schéma počítača skladá zo piatich blokov:

1. **ALU – Aritmeticko-Logická jednotka** – jednotka vykonávajúca všetky aritmetické a logické operácie. Obsahuje bloky určené na aritmetické operácie ako sčítanie, odčítanie, násobenie a delenie a bloky na logické operácie ako porovnávanie a pod. Úlohou ALU je krok po kroku vykonávať program uložený v pamäti.



2. **Operačná pamäť** – slúži ako skladisko pre samotný program, dáta programu, dočasné skladisko pre medzivýpočty a samotné výsledky. V operačnej pamäti sa nachádzajú miesta na uloženie daných dát, ktoré je možné adresovať a tým čítať a zapisovať z a do ľubovoľného miesta v pamäti.

   Najzákladnejšou bunkou pamäte je jeden bit, ktorý reprezentuje logický stav 0 alebo 1. Množina ôsmich bitov sa nazýva jeden bajt (slabika) a je to najzákladnejšia a najmenšia adresovateľná jednotka v pamäti. Keďže jeden bajt je pomerne malá jednotka, pre potreby vyčíslenia väčších pamäťových kapacít sa používajú násobky kilo, mega, giga a najnovšie aj tera, pričom sa tieto násobky mierne odlišujú od bežných metrických násobkov v číselnom vyjadrení. Zatiaľ čo napr. kilogram má 1000 gramov, kilobajt obsahuje 1024 bajtov. Je to dané tým, že číslo 1000 nie je mocninou dvojky a najbližšie číslo, ktoré túto podmienku spĺňa je 2^{10}=1024. Teda:
   1 kilobajt = 2^{10} bajtov = 1024 bajtov
   1 megabajt = 1024 kilobajtov = 2^{20} bajtov = 1 048 576 bajtov
   1 gigabajt = 1024 megabajtov = 2^{30} bajtov = 1 073 741 824 bajtov
   1 terabajt = 1024 gigabajtov = 2^{40} bajtov = 1 099 511 627 776 bajtov
   Okrem bajtov sa v počítačovej terminológii používa aj pojem word (slovo). Veľkosť jedného slova je daná hardvérom a operačným systémom a pohybuje sa od 1 až do 4 bajtov.
3. **Radič** – riadiaca jednotka počítača, ktorá riadi jeho celú činnosť. Toto riadenie sa uskutočňuje pomocou riadiacich signálov, ktoré predáva každému zariadeniu. Reakciou na riadiace signály sú stavové hlásenia radiča, ktoré sú mu posielané na spracovanie a následné rozhodnutie nad ďalším krokom.
4. **Vstupné zariadenie** – zariadenie, ktoré slúži na vstup programu a dát

5. **Výstupné zariadenie** – zariadenie, ktoré slúži na výstup spracovaných dát, ktoré ALU spracovala pomocou programu.

Medzi týmito blokmi počítačovej schémy prebieha neustála komunikácia, ktorá sa dá rozdeliť na tri časti:

- **Riadiace signály radiča –** týmito signálmi predáva radič informácie ostatným zariadeniam svoje požiadavky.
- **Stavové hlásenia pre radič** – tieto signály sú v podstate odpoveďou na riadiace signály. Zariadenie nimi dáva informácie radiču o úspešnej/neúspešnej vykonaní požadovanej operácie, poprípade poskytne dodatočné informácie požadované radičom.
- **Dátový tok** – predstavuje samotné dáta prúdiace zo vstupných zariadení cez ALU do pamäte alebo výstupných zariadení.

Počítač, ktorého metódy spĺňajú metódy von Neumannovej schémy pracuje nasledovne:

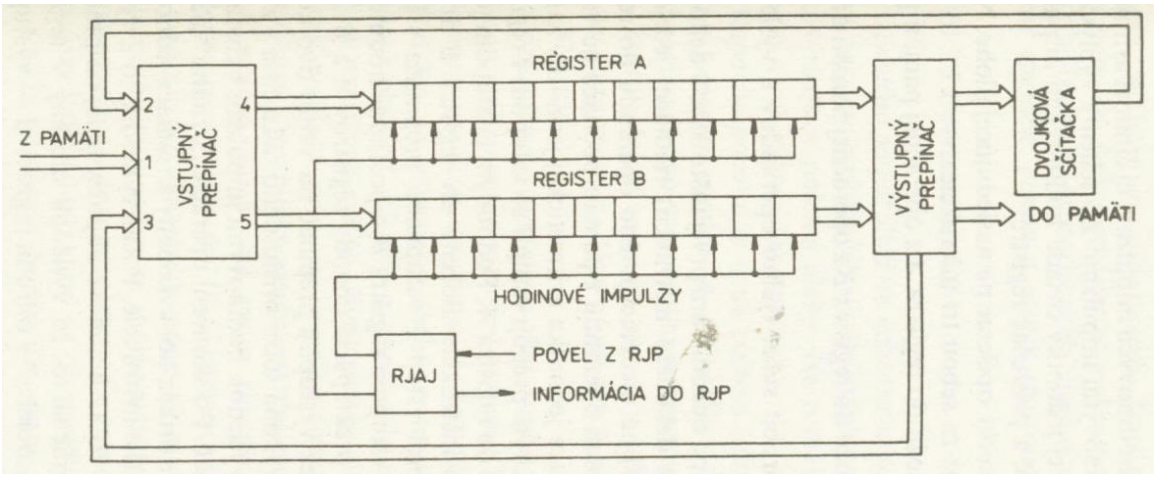
1. Do pamäte sa cez ALU zapíše program (postupnosť inštrukcií, ktoré sú postupne vykonávané ALU) zo vstupných zariadení. Takýmto spôsobom sa zapíšu do pamäte aj vstupné dáta, ktoré program požaduje.
2. Prebehne vlastný výpočet, ktorý postupne vykonáva ALU. Táto jednotka je riadená radičom, pričom si medzivýsledky ukladá do pamäte.
3. Po ukončení programu sú výsledky kontrolovane poslané na výstupné zariadenie.

Dnešné počítače sa v akomkoľvek prevedení a forme v teoretickej rovine podobajú tejto schéme. Je samozrejmé, že evolúcia počítačov sa mierne podpísala aj pod niektoré výnimky, ktoré nie sú obsiahnuté vo von Neumannovej schéme:

- Dnešné počítače dokážu spracovávať niekoľko úloh a teda aj programov naraz – multitasking.
- Počítač môže disponovať viacerými procesormi.
- Existujú aj takzvané vstupno/výstupné zariadenia, z a do ktorých môže byť program a jeho výsledky zapísané a načítané.
- Program sa nemusí zaviesť do pamäti celý, stačí len jeho najdôležitejšia časť, pričom ostatné časti sa zavedú vo chvíli keď sú potrebné
- Program sa nemusí nahrávať do pamäte cez procesor – DMA (Direct Memory Access – priamy prístup do pamäte), šetrí procesorový čas, ktorý sa vďaka tomu môže venovať vykonávaniu programu.

Von Neumannova schéma predstavuje len teoretickú schému a popis činnosti počítača. V praktickej rovine sa táto schéma pomerne verne zhoduje s hardvérovým vybavením dnešných počítačov.

4.1 Aritmetická jednotka



Bloková schéma aritmetickej jednotky počítača

Aritmetická jednotka začína svoju činnosť na povel z riadiacej jednotky počítača, presnejšie na povel z dekódera inštrukcií riadiacej jednotky**. Aritmetická jednotka obsahuje vstupný a výstupný prepínač, register A a B, dvojkovú sčítačku a vlastnú riadiacu jednotku (RJAJ). R**egistre, ktorých dĺžka je daná dĺžkou strojového slova, sa realizujú preklápacími obvodmi. Ak má strojové slovo dĺžku 16 bitov, potom register A a B má 16 preklápacích obvodov. Podobne, ako je činnosť celého pc riadená hodinovými impulzmi, aj činnosť aritmetickej jednotky je riadená hodinovými impulzmi.

Činnosť aritmetickej jednotky opíšeme na nasledujúcej úlohe. Nech sú v pamäti pc uložené za sebou tri inštrukcie:

1. inštrukcia prikazuje uložiť do registra A z operačnej pamäti prvý operand (číslo)

2. inštrukcia prikazuje uložiť do registra B z operačnej pamäti druhý operand (číslo)

3. inštrukcia prikazuje urobiť súčet týchto operandov a výsledok uchovať v registri B

Riadiaca jednotka vydá po dekódovaní 1. inštrukcie v dekóderi inštrukcií príkaz RJAJ na uloženie 1.operandu do registra A. Súčasne zabezpečí presun operandu z pamäti pc na vstup AJ. RJAJ zabezpečí cez vstupný prepínač vloženie 1. operandu do registra A. Podobným postupom je na základe dekódovanej 2. inštrukcie uložený do registra B 2. operand. Sčítačka vykoná súčet príslušného rádu čísla a z výstupu dvojkovej sčítačky sa výsledok hodinovými impulzmi vkladá do registra B cez vstupný prepínač. Po skončení operácie oznámi RJAJ riadiacej jednotke pc, že príklaz bol vykonaný a riadiaca jednotka pc začne vykonávať ďalšie inštrukcie.