Gmail **Lukáš Hozda <luk.hozda@gmail.com>**

Programování - Třináctá a čtrnáctá hodina

1 message

Lukáš Hozda <luk.hozda@gmail.com> Sun, Mar 3, 2019 at 10:35 PM

Zdravím všechny,
posílám látku z posledních dvou hodin. Na třináctě hodině jsem vám nejdříve ukazoval hardware. Sice trošku starší, ale věřím, že i tak reprezentoval, co jak zhruba vypadá, popř. k čemu to je. Hardwaru jste se snad věnovali i trochu na normální informatice a věřím, že pokud vás budou jednotlivá zařízení zajímat, že si to vygooglíte.

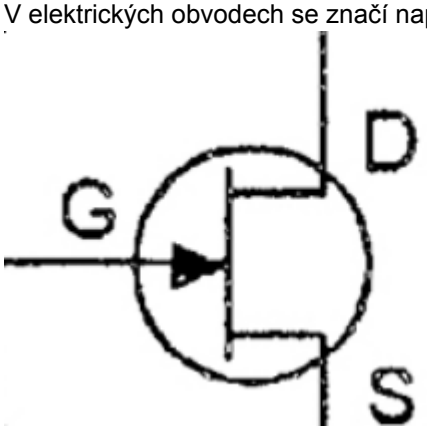
Primárně jsme se ale bavili o tranzistorech a o RAM na třináctě hodině, o logických hradlech a o ALU (a trošku procesorech obecně) na čtrnáctě hodině.

Tranzistor

Tranzistor je elektronická součástka, která obsahuje polovodič. V počítačích se používá typ tranzistoru zvaný **FET** (= Field Effect Transistor), což je tranzistor řízený elektrickým polem. Konkrétně se jedná o **MOSFET**y, u kterého je to elektrické pole řízeno strukturou kov (**M**etal) - **O**xid - polovodič (**S**emiconductor)

Pro naše účely stačí tranzistor zjednodušit jako součástku, která má tři drátky, kterým se v angličtině říká **Gate (G), Drain (D)** a **Source (S).**

V elektrických obvodech se značí například takto:



Na tomto obrázku je dokonce nakreslený v kroužku, ale to dělat nemusíte. Způsobů jak zakreslit tranzistor je více, ale pokud to tohle bude zhruba připomínat, tak je daná součástka v obvodu pravděpodobně tranzistor :)

V tranzistoru proud teče z **drain** do **source** (v závislosti na zapojení a pokud respektujeme to, že směr proudu vlastně kreslíme kvůli konvenci v elektrických okruzích špatně, může to být i opačně), ale jen pokud má **Gate** kýžený stav.

Podle toho rozlišujeme **NFET** a **PFET** tranzistory. V **NFET**u teče proud **D<->S**, jen pokud je **Gate** pod napětím, v **PFET**u naopak.

Zde jsou názorné pravdivostní tabulky (1 indikuje přítomnost napětí):

```
NFET
 D | G | S
-----------
 0 | 0 | 0
```

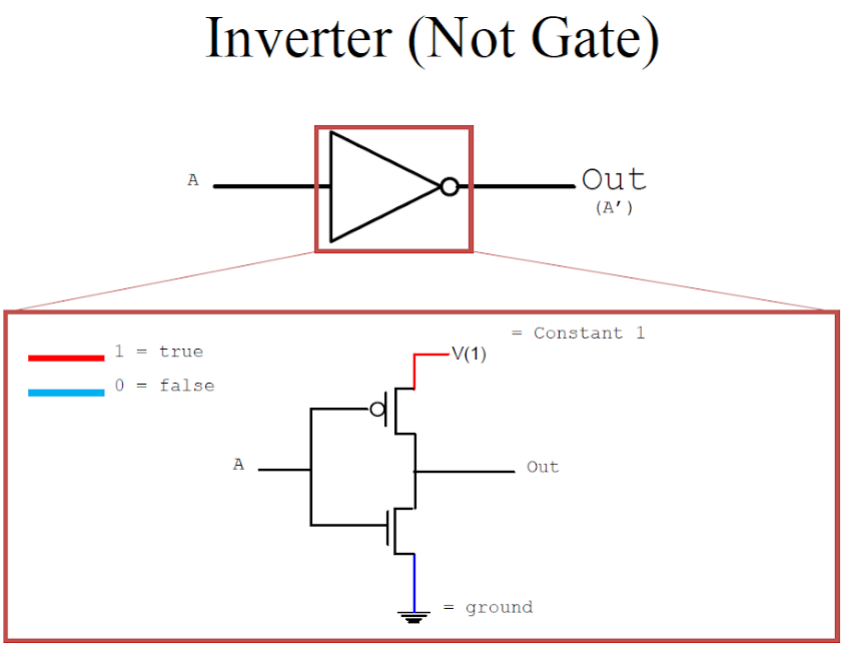
```
 0 | 1 | 0
 1 | 0 | 0
 1 | 1 | 1

PFET
 D | G | S
-----------
 0 | 0 | 0
 0 | 1 | 0
 1 | 0 | 1
 1 | 1 | 0
```

Můžete si rovnou povšimnout, že pravdivostní tabulka NFETu připomíná jednu logickou operaci, ale to je jen drobná zajímavost.

Tranzistory jsou pro počítače klíčové, jsou takovou základní stavební jednotkou pro mnoho součástí, např. paměti RAM, disky (SSD, Flash) a hlavně procesory.

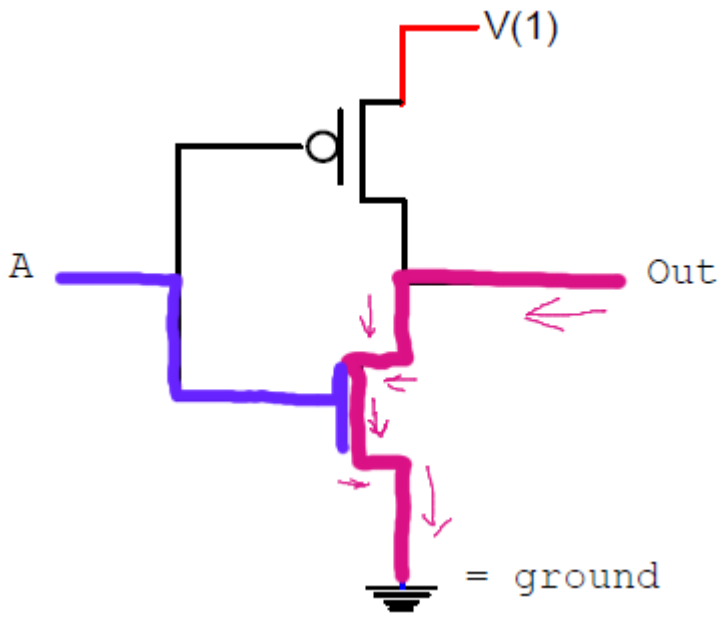
NOT gate
Když jsem vám ukazoval zapojení SRAM, ke kterému se za chvíli dostaneme, tak jsem vám také ukázal, jak je pomocí tranzistorů vytvořené logické hradlo NOT, které představuje logickou operaci negace.

Myslím si, že je výhodnější to ukázat už teď. Zde je schéma:



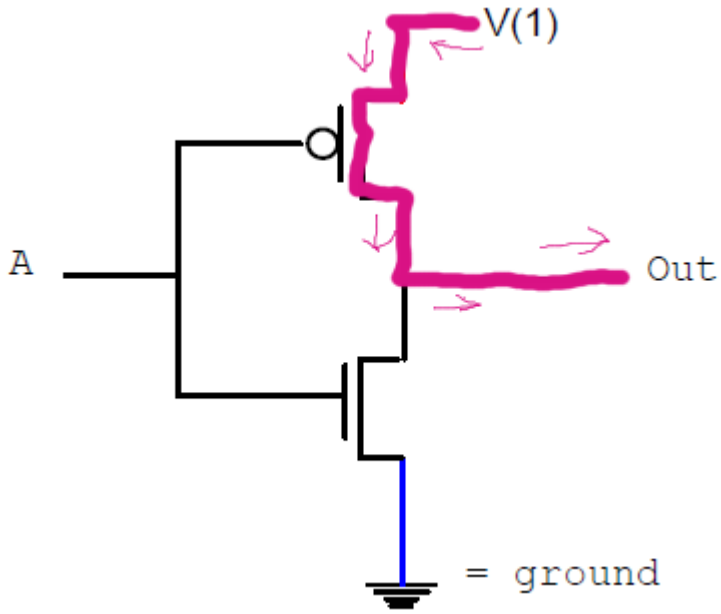
Vstup, který má být znegován (**A**) je zapojen ke dvoum tranzistorům jako gate, ten spodní je NFET a ten horní je PFET. Oba tranzistory mají buď drain nebo source zapojený k výstupu. U spodního NFETu je source uzeměný a u PFETu jde do drainu konstantě proud.

Pokud bude do A přicházet proud, tak vzpomeneme li si na vlastnosti tranzistorů, spojí se obvod takto:



Výstup je uzeměn, aby nedělal problémy zbytkový náboj. Má-li totiž být výstup jasný, je potřeba tvrdá nula.

Pokud A není pod proudem, vypadá to takto:



PFET tedy vede proud pouze pokud G neni pod proudem. To umožňuje, aby konstatní proud z V tekl do výstupu. Spodní NFET proud nevede, protože jeho G také není pod proudem a proto se proud z V nedostane do země, výstup je cesta nejmenšího odporu.

RAM (Random Access Memory)

RAM je typ paměti klíčový pro počítače. V nějaké formě jí má snad každý počítač, který se vám dostane do ruky. Používá se jako operační paměť počítačů. tj. paměť, kde jsou uložen načtený operační systém a právě běžící programy, plus jejich data.

Jedná se o volatilní paměť -> vypnutí počítače (tj. odpojení RAM od přísunu elektřiny) jí vymaže, kdežto nevolatilní paměti nepotřebují napětí k uchování dat.

Česky se zkratka RAM přeloží jako paměť s náhodným/libovolným přístupem. Náhodným přístupem se zde myslí to, že dobra přístupu k obsahu není závislá na jeho fyzické lokaci v zařízení. Prostě kdykoliv si

můžeme přečíst/zapsat jakýkoliv kousek dat bez jakéhokoliv přetáčení. Dalo by se říci, že časová komplexita čtení RAM je teoreticky O(1) (byť v reálu se to třeba liší), ale o této notaci se budeme bavit později.

Opakem paměti s náhodným přístupem je paměť se sériovým přístupem, kdy je potřeba médium "přetáčet", a kdy je doba přístupu závislá na lokaci dat na něm.

Jasným příkladem paměti se sériovým přístupem je kazeta, dále také HDD, CD, DVD nebo disketa.

Přikladem pamětí s náhodným přístupem kromě RAMky jsou například SSD, flash disky nebo SD karty.

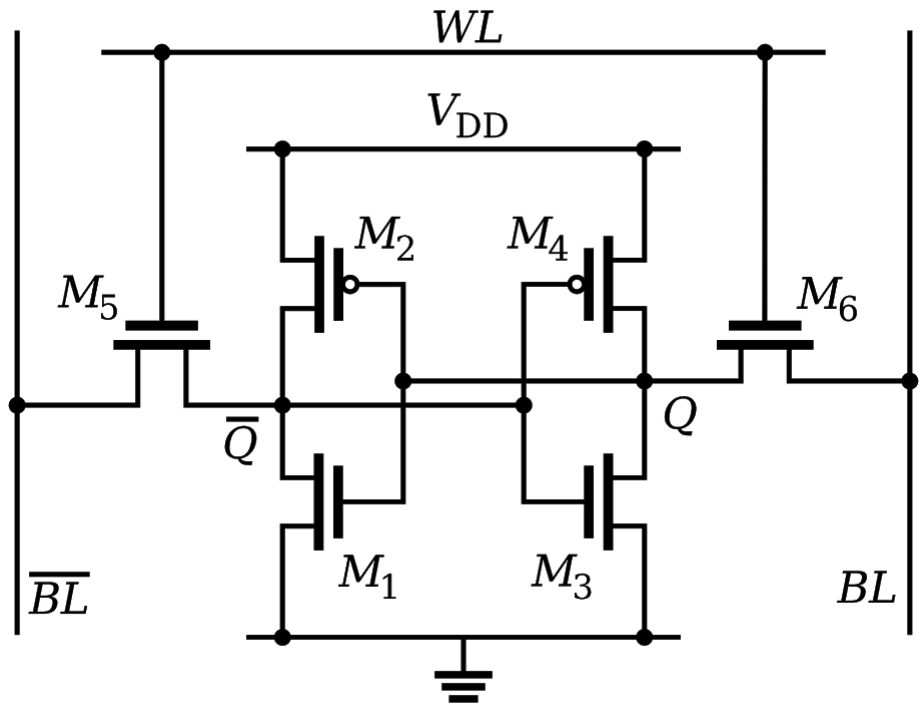
Je velká šance, že pokud se při operace nějakého média nic nehýbe, tak se jedná o paměť s náhodným přístupem, a pokud ano, tak je to sériová paměť.

Paměť RAM je rozdělena na adresy, jejichž maximální počet je omezen architekturou počítače

RAM paměti rozlišujeme dva typy, SRAM a DRAM.

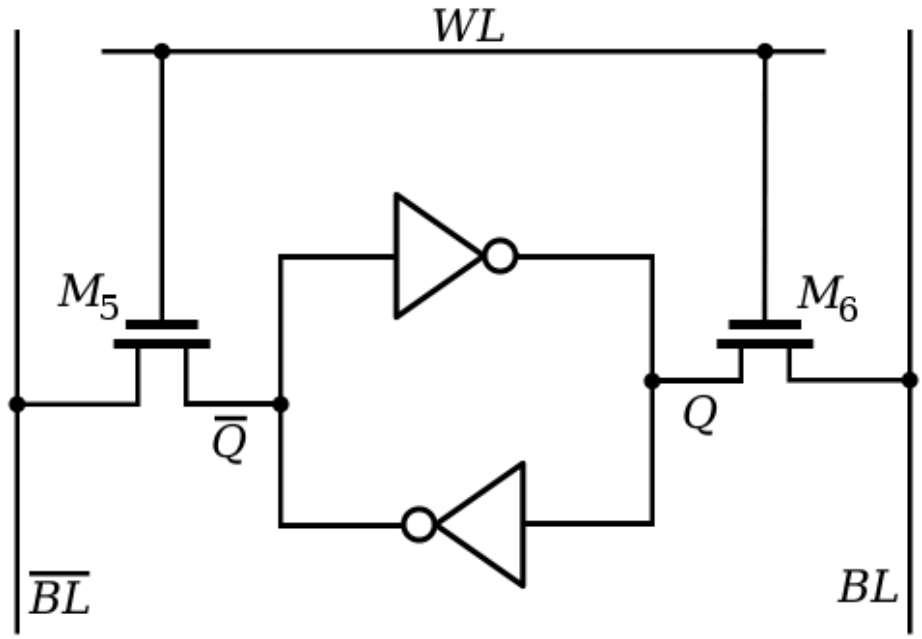
SRAM
Písmeno S zde znamená **Static**, tj. statická RAM. Jedná se o dražší typ paměti, se složitější strukturou, která ovšem potřebuje ke své operace méně energie (= nemusí se obnovovat tak často).

Jedna buňka (tj. obvod, ve kterém je uložen jeden bit) v SRAM vypadá zhruba takto:



Jestli máte trochu představivost a četli jste předcházející část tohoto emailu, může vám prostřední obvod začít něco připomínat :)
Rád bych vypíchnul **WL**, což je takzvaná **Word Line**, také se značí **w**, a **BL** a její negaci **BL s pruhem** je takzvaná **Bit Line** a její negace, také se značí **b** a **b'.** K tomu, k čemu se využívají se dostaneme za chvíli.

Čas na představivost skončil, zde je identická buňka zakreslena druhým způsobem:



Přesně tak, jádrem buňky SRAM jsou dvě NOT hradla.

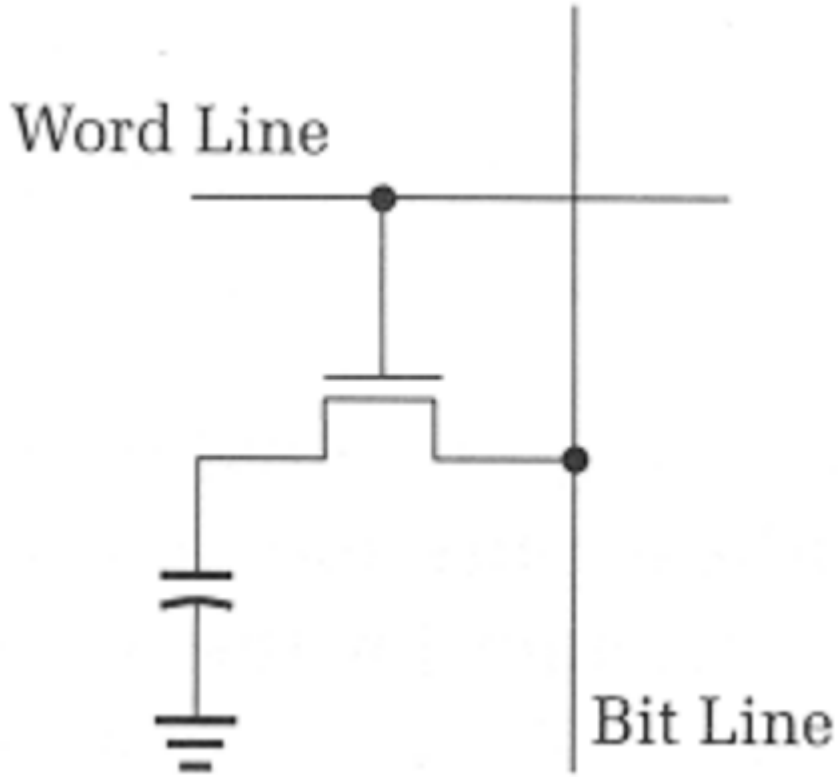
Pokud do této buňky chceme bit uložit, tak pustíme proud do všech tří linek, tj. **b, b'** a **w**. Ve vnitřním okruhu s NOT hradly se uloží náboj a je v podstatě živen stálým přísunem proudu do hradel (viz. NOT hradlo výše). Toto je velmi úsporné energeticky, ale náročnější na prostor a dražší, protože 6 tranzistorů na buňku je docela dost.

Pro čtení se pustí proud pouze do **w/WL**. To zapříčíní, že se spojí tranzistory M6 a M5 a náboj unikne z vnitřního obvodu. Na **b/BL** bude uložená hodnota a na **b'** bude její opak, tzn. pokud nebylo v buňce uložené nic, na **b'** bude 1 a na **b** bude 0, pokud byla v buňce uložená jednička, tak **b' -> 0** a **b -> 1.**

DRAM

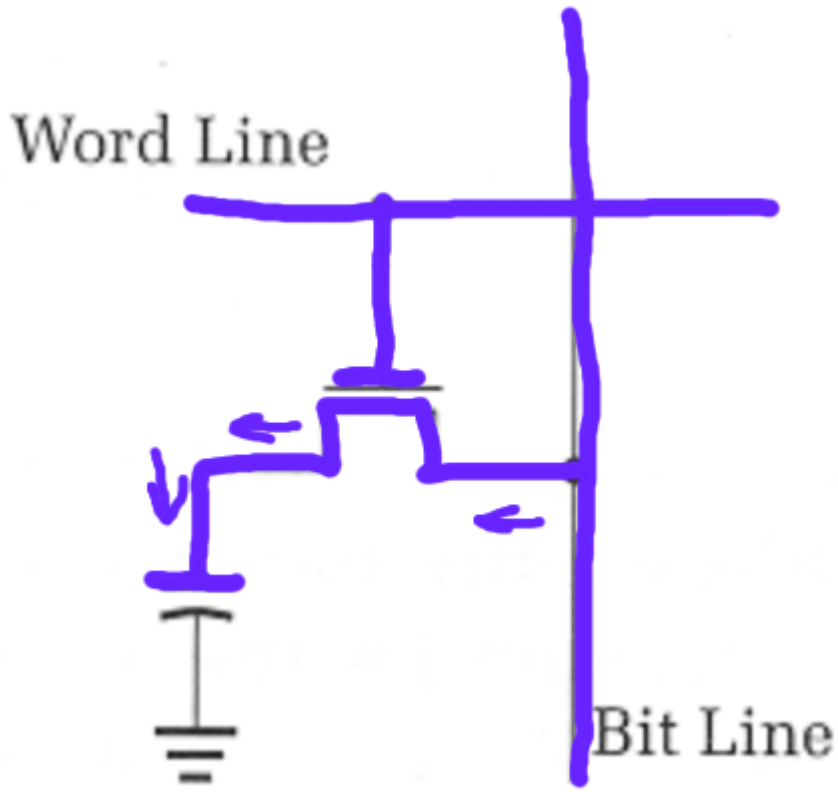
Druhým a trochu více rozšířeným typem RAM je DRAM, kde D znamená **dynamic**. Tato RAM je mnohem jednodušší a levnější (rozdíly jsou v ceně za GB jsou až desetitisících). Najdete jí jako primární operační paměť ve vašem desktopu, laptopu, tabletu, telefonu i třeba chytrých hodinkách.

Skládá se pouze z jednoho tranzistoru a jednoho kondenzátoru. Zde je schéma:

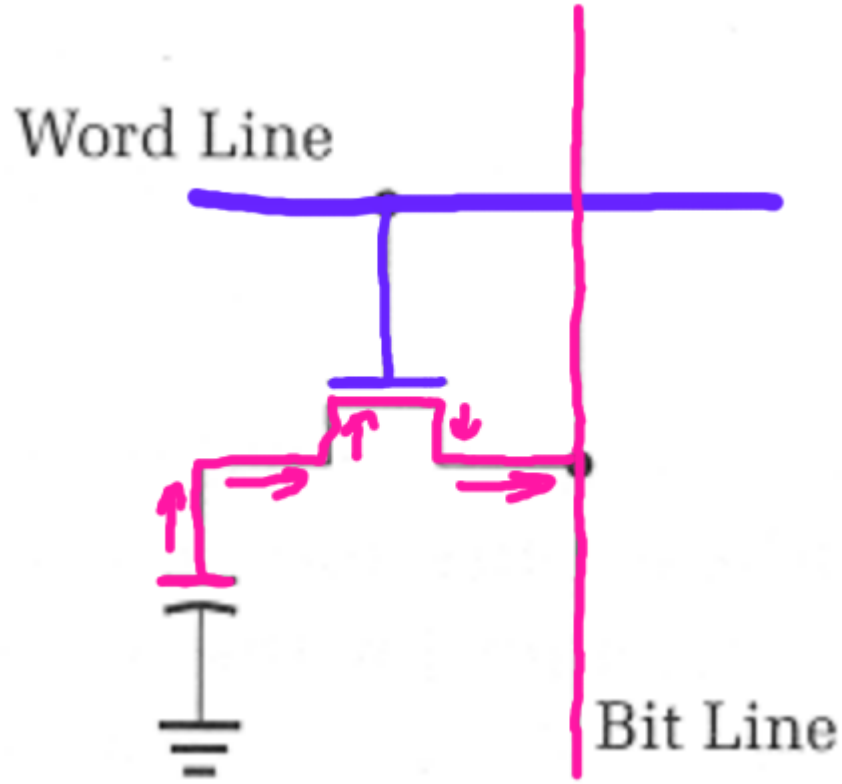


Stejně jako SRAM zde najdeme **World Line** (někdy se také nazývá **Select Line** nebo **Adress Line**) a **Bit Line**, ale nenajdeme zde jeho negaci. **w** je zapojené k **gate** tranzistoru, a **b** je připojené buď k **drain** nebo **source** tranzistoru. Z druhé strany tranzistoru je zapojen uzeměný kondenzátor.

Když chceme do kondenzátoru uložit náboj, tj. uložit do buňky jedničku, tak pustíme proud jak do **w**, tak do **b**, tím se spojí tranzistor a do kondenzátoru se uloží náboj. Takže takto:



Pokud si chceme buňku přečíst, tak pod proudem bude pouze **w**. Tím se spojí tranzistor a uložený náboj unikne na bit line:



Bohužel, náboj se v kapacitoru neuloží na dlouho, proto je potřeba DRAM obnovovat, což má za následek větší spotřebu elektřiny.

To je zhruba k RAM zatím vše.

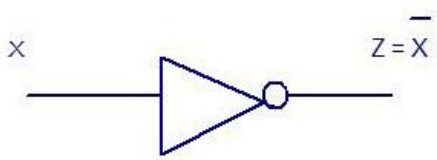
Logická hradla/spojky

Logická hradla (anglicky logic gate) neboli logické členy jsou základním prvkem logických obvodů. Pracují s pravidivostními hodnotami, pokud jste již měli na matematice výrokovou logiku, bude vám toto povědomé. Pravdivostní hodnoty jsou **pravda/true/1** a **nepravda/false/0**.

Každé hradlo má nějaký vstup/y a nějaký výstup. Definujeme si tyto hradla: **NOT, OR, XOR, AND, NAND** a **NOR.**

Zde je pro zopakování rychle **NOT** s jeho pravdivostní tabulkou:

NOT Gate



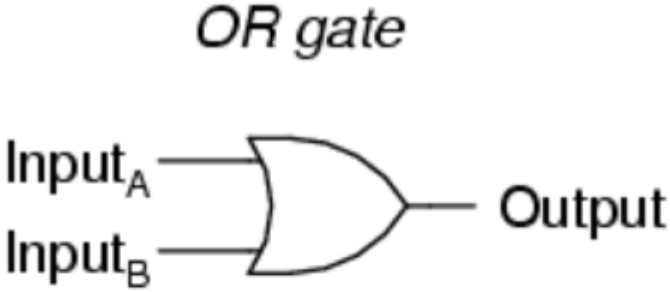
TRUTH TABLE

INPUT	OUTPUT
X	Z
0	1
1	0

Zkrátka když je na vstupu jedna, dostaneme na výstupu nulu a obráceně.

OR/disjunkce

Toto hradlo má dva vstupy a na výstupu vrací jedničku, pokud je alespoň jeden ze vstupů jedna:



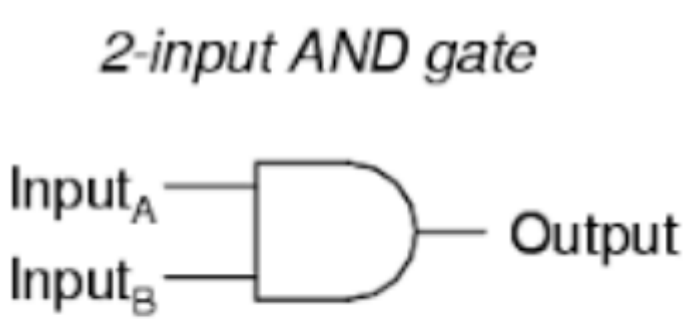
A	B	Output
0	0	0
0	1	1
1	0	1
1	1	1

To znamená, pokudu je A nebo B jedna, tak výstup bude jedna.
V logických výrazech se jako operátor pro OR používá **+,** takže **A OR B ~ A + B**

XOR/exkluzivní disjunkce
Exkluzivní disjunkce je podobná ORu, akorát na jejím výstupu je jednička, **právě, když jeden ze vstupů je jedna**. To znamená, že pokud budou na vstupu dvě jedničky, výstupem bude nula.
Tady je, jak se kreslí v obvodech a jak vypadá její pravdivostní tabulka:
2019-03-03-213615_273x325_scrot.png

Takže pouze v prostředních dvou případech je výstupem jedna.
V logických výrazech se používá jako operátor znak ⊕**,** takže **A XOR B ~ A ⊕ B**

AND/konjunkce
Konjunkce je logická operace, jejíž výstupem je pravda pouze, pokud jsou oba vstupy pravda. Takže jedničku dostaneme pouze, když dostaneme na vstup dvě jedničky.

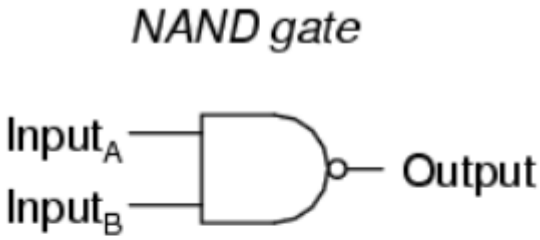


A	B	Output
0	0	0
0	1	0
1	0	0
1	1	1

Pouze v posledním případě je výstupem jedna. Jako operátor používáme · nebo jej nepíšeme vůbec, takže **A AND B ~ A · B ~ AB**

NAND/negace konjunkce

NAND je logické hradlo, jež kombinuje operaci **AND** a **NOT**, takže kdybychom výstup **AND** připojili jako vstup **NOT**, tak výstup tohoto obvodu by byl identický **NAND** hradlu.



A	B	Output
0	0	1
0	1	1
1	0	1
1	1	0

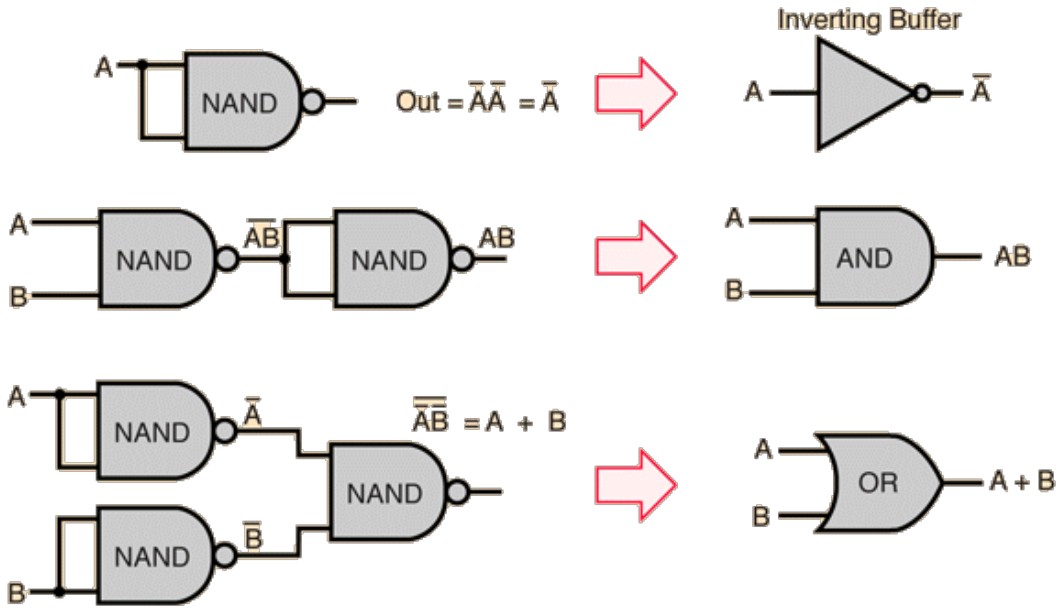
Zkrátka nula pouze pokud jsou oba vstupy jedna. Jako operátor se používá takzvaná **Shefferova šipka/spojka (** $\uparrow$**)**, takže **A NAND B ~ A** $\uparrow$ **B.**

NOR/negace disjunkce

V podstatě představuje pro **OR** to samé, jako **NAND** pro **AND**. Zkrátka negace.

2019-03-03-215930_273x329_scrot.png
Tedy, protože se jedná o negaci operace **OR**, tak výstupem je jedna pouze, pokud ani jeden ze vstupů není jedna. Jako operátor se používá **Peircova šipka/spojka (↓),** takže **A NOR B ~ A ↓ B.**

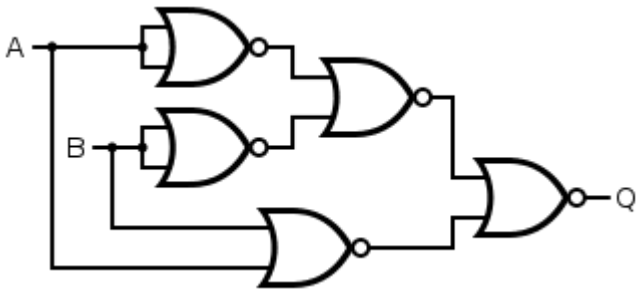
Univerzální hradla
Poslední dvě zmíněná logická hradla se také nazývají univerzální. Je to proto, protože s nimi lze poskládatat všechna ostatní. To je také důvodem proč se v počítačích primárně používají: je mnohem efektivnější a levnější vyrobit jednu součástku miliardakrát a pak to nějak pospojovat pomocí drátků, než se pachtit se specifickými součástkami. **NAND** se pro tento účel používá více než **NOR** (např. pokud si vzpomenete, že jste někde viděli, že nějaká paměť je typu **NAND Flash,** tak teď už víte, kde se to **NAND** vzalo :)).



2019-03-03-221252_490x152_scrot.png
Takto se to nějak poskládá, aby vyšly ty ostatní pomocí **NAND**ů. Na tomto obrázku chybí **NOR** složený z **NAND**ů, ale to není těžké domyslet. Prostě se za **OR** poskládaný z **NAND**ů viz obrázek přilepí ještě jeden NAND, viz první schéma, abychom to znegovali, a dostaneme tím **NOR**.

Poskládat všechny spojky pomocí **NOR**ů to vypadá zhruba takto:

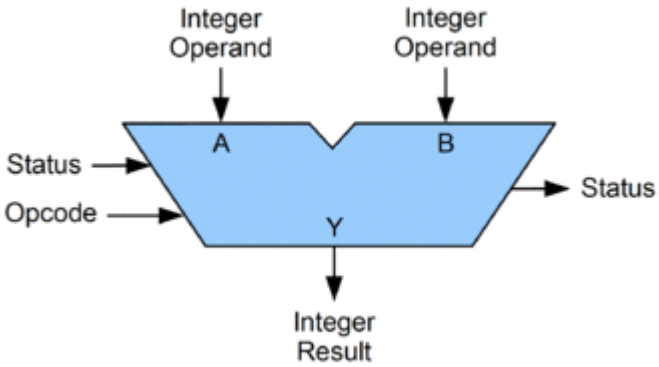
log26.gif



XOR

Pokud bychom chtěli vytvořit **NAND**, tak budeme postupovat stejně jako u vytváření **NOR**u z **NAND**ů.

ALU - aritmetická a logická jednotka



ALU je součást procesoru, která se stará o vykonávání aritmetických a logických operací na celých číslech. Obrázek výše je symbolická reprezantace ALU.
Vidíme zde dva vstupy, **A** a **B** a jeden výstup **Y**. Kromě toho také máme na vstupu takzvaný **opcode**, což je číslo, které ALU říká, jakou operaci má vykonat, tedy kód operace. Často také máme status na vstupu a status na výstupu, protože některé operace, které ALU vykonává jsou na více kroků, a tedy je potřeba si předávat nějaký status.

ALU je primárně složené ze dříve zmíněných logických hradel. Na hodině jsem vám sice ukazoval sčítačku, tedy half-adder a adder a jak sečíst osmibitová čísla, ale bylo to trošku na rychlo, takže to ještě zopakuji a nakreslím zítra, stejně jako základní informace o procesorech, které také nejsou v tomto emailu.

Díky,
LH