CPU

CPU (**C**entral **P**rocessing **U**nit) je elektrický obvod, který vykonává instrukce uložené ve vnitřní paměti. Vykonání **jedné** instrukce spočívá ve třech krocích

Fetch → Decode → Execute

- **Fetch** je vyzvednutí instrukce z vnitřní paměti. CPU zkrátka přečte tolik bitů z paměti, kolik je délka jeho instrukce (mezi různými CPU se liší), z toho místa, odkud četl naposledy.
- **Decode** přeloží vyzvednutou instrukci. Opět, každý procesor má svoje specifikace, co se kódování instrukcí týče. Pro jedno CPU může třeba 10001111 znamenat „sečti“, pro jiný „načti vstup z klávesnice“.
- **Execute** vykoná přeloženou instrukci. Vykonání obvykle spočívá v delegaci instrukce jedné z jednotek CPU (popsány níže).

CPU má v sobě ukazatel (tzv. *counter* nebo *čítač*), pomocí něhož si pamatuje, na kterém místě v paměti zrovna je. Po každém vykonání instrukce posune čítač na následující místo v paměti. Slovo „místo“ zde může znamenat různý počet bitů, v závislosti na architektuře CPU.

Několik základních instrukcí, které CPU může z paměti přečíst, je:

- **LOAD** – načti číslo z paměti do CPU,
- **ADD** – sečti dvě čísla, která v paměti následují (CPU implementuje ostatní operace jako odčítání, násobení a dělení pomocí sčítání),
- **STORE** – ulož číslo do paměti,
- **COMPARE** – porovnej dvě čísla, která v paměti následují,
- **JUMP** – přesuň čítač na místo v paměti určené následujícím číslem,
- **JUMP if podmínka** – přesuň čítač na místo v paměti určené následujícím číslem, pokud byla splněna **podmínka**,
- **OUT** – předej výstup nějakému zařízení (třeba monitor, reproduktor apod.),
- **IN** – získej vstup od nějakého zařízení (třeba myš, klávesnice apod.).

Původní CPU vykonávaly vždy jednu instrukci za jeden úder *vnitřních hodin*. Počet úderů za vteřinu značí, jak rychle je CPU schopen vykonávat instrukce. Moderní CPU mají frekvenci vnitřních hodin měřenou v gigahertzích (tj. v miliardách úderů za vteřinu). Navíc mají obvykle více *virtuálních jednotek* (též *jader*) a jsou schopny během jednoho úderu vykonat souběžně více instrukcí. Konkrétně, CPU, který vykoná jednu instrukci za jeden úder a frekvenci úderů má 1 Ghz, vykoná přesně miliardu instrukcí za vteřinu.

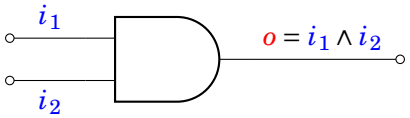
CPU bývá obvykle rozdělen do několika menších jednotek, z nichž nezanedbatelné jsou

- **Control Unit** („řídící jednotka“), která obdrží instrukce z vnitřní paměti a poté je rozešle ostatním jednotkám v závislosti na povaze instrukce;
- **Arithmetic Logic Unit** („jednotka výpočetní logiky“), která vykonává instrukce (obdržené od Control Unit) představující aritmetické (+, −, ∗, /, ...), logické (ne, a, nebo,...) nebo srovnávací (<, >, =) operace.

Mnoho moderních CPU má navíc ještě **Memory Management Unit** („jednotku správy paměti“), která obstarává instrukce typu **STORE** nebo **JUMP** – obecně instrukce související s pamětí.

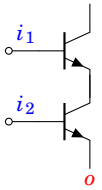
Logické obvody

Samotné CPU je složeno z pospojovaných logických obvodů, které jsou zase složeny z logických bran. **Logická brána** je prostě jen elektrický obvod, který je schopný „vykonat“ logickou operaci. Třeba AND brána vykonávající logickou operaci „a“ se kreslí takhle:



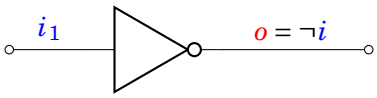
AND brána. Výstupem o prochází proud, jedině když jak vstupem i_1, tak i_2, prochází proud. To odpovídá logické operaci „a“. Totiž, když $i_1 = i_2 = 1$, pak $i_1 \wedge i_2 = 1$. Ve všech ostatních případech (tj, $0,1$; $1,0$ a $0,0$) je $i_1 \wedge i_2 = 0$.

Snadný způsob, jak postavit AND bránu je prostě vzít drát se zdrojem a po jeho délce kamkoli umístit dva transistory. Transistory jsou zařízení, která zesilují elektrický proud. Tedy, pokud oba transistory budou zapnuté, pak poteče drátem silný proud, který reprezentuje číslo 1. Pokud je byť i jeden z nich vypnutý, bude proud dostatečně slabý, aby byl interpretován jako 0. Pro jednoduchost si lze transistory v logických branách představovat tak, že když jsou zapnuté, tak „pouštějí“ proud, jinak ho „zastavují“.



AND brána jako dvojice transistorů.

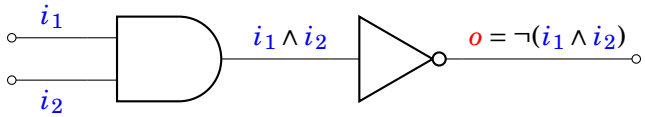
Další brána, kterou si musíme představit, je NOT brána. Ta odpovídá logické negaci. Vychází z ní proud, když do ní nepřichází, a naopak z ní nevychází, když do ní přichází. Kreslí se takhle:



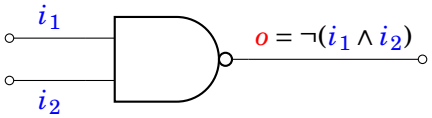
NOT brána. Výstupem o prochází proud, jedině když vstupem i proud neprochází. Odpovídá logickému „ne“. Totiž, $\neg 0 = 1$ a $\neg 1 = 0$.

Jednoduchou NOT bránu lze postavit jako drát se zdrojem a jedním rezistorem. Rezistor si lze v logických obvodech představovat jako zařízení, které „pouští“ proud, když je **vypnuté** a „nepouští“ proud, když je **zapnuté**.

Ve skutečnosti, logický obvod vykonávající libovolnou logickou operaci lze postavit pouze použitím tzv. NAND bran, což jsou jen brány vzniklé spojením AND a NOT brány. Kreslí se takhle:



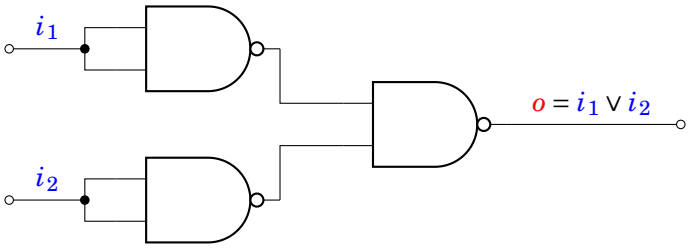
NAND brána jako spojení AND a NOT brány.



NAND brána zjednodušeně.

NAND brána. Odpovídá logickému výrazu $\neg(i_1 \wedge i_2)$. Tedy, NAND bránou **neprochází** proud jedině, když oběma vstupy i_1 i i_2 proud **prochází**.

Jako ukázku toho, že se každý logický obvod dá postavit pouze z NAND bran, z nich postavíme OR bránu, tj. logickou bránu implementující logické „nebo“.

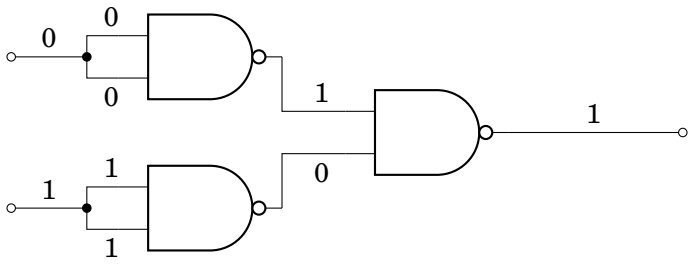


Obvod implementující logickou OR bránu. OR bránou **neprochází** proud jedině, když ani jedním ze vstupu proud rovněž **neprochází**. Ve všech ostatních případech jí proud prochází. Symbol • tady značí *rozbočovač*; zkrátka zařízení, které rozpůlí proud, jenž jím prochází.

Abychom ověřili, že obvod nahoře je opravdu logické nebo, museli bychom zkusit

za i_1 a i_2 dosadit všechny možnosti $0,0;0,1;1,0;1,1$, kde 0 (zase) znamená, že vstupem proud neprochází a 1, že ano.

Zkusíme například $i_1 = 0$ a $i_2 = 1$, pro které by měl OR bránou proud projít. Jednoduchý způsob, jak sledovat průchod proudu logickým obvodem, je zkrátka kreslit 1 nebo 0 nad každý kus drátu. Například takhle:



Příklad průchodu proudu OR bránou.

Vnitřní paměť

Vnitřní paměť (tou se myslí paměť, ve které má CPU uloženy instrukce) přichází ve dvou chutích – volatilní a nevolatilní (angl. volatile a non-volatile, asi to překládám blbě...). Volatilní paměť ztrácí veškerá data po vypojení z proudu, zatímco nevolatilní nikoliv.

Moderní počítače používají jako vnitřní paměť téměř výhradně **volatilní**, neboť je levnější na výrobu a taky je mnohem kompaktnější. Vlastně jedinou stále používanou volatilní pamětí je RAM (**R**andom **A**ccess **M**emory). Výraz „random access“ značí, že se jedná o paměť, která umí zpřístupnit svůj obsah okamžitě, bez ohledu na to, v kterém místě je uložen. To právě umožňuje operaci **JUMP** vykonat ihned, stačí znát cílové místo v paměti (to není ten případ u externích pamětí, které je potřeba obvykle procházet skoro od začátku).

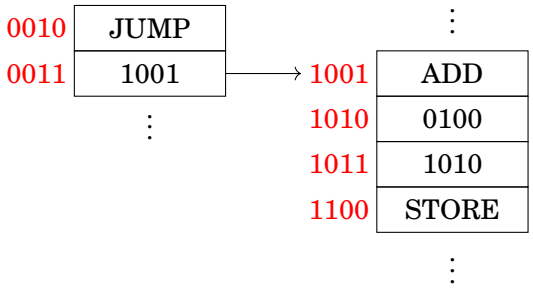
RAM se používá ve dvou formách:

- SRAM nebo (**S**tatic RAM) je typ volatilní paměti, ve které se data za běhu neposouvají, tj. za neustálého přísunu proudu je schopna udržet uložené údaje teoreticky nekonečně dlouho. Používá se hlavně pro vnitřní paměť samotného CPU (tzv. cache), protože je velmi rychlá díky tomu, že nepotřebuje žádnou správu – uložená data jsou vždy tam, kam byla uložena.
- DRAM nebo (**D**ynamic RAM) je ten druhý typ RAMky, kde se data s postupem času ztrácejí (i když do ní stále jde proud). Proto je v ní třeba důležitá data (třeba ta nutná pro běh OS) neustále obnovovat. Tato potřeba se u DRAMek, jejichž velikost se měří v GB, stala natolik častou, že se pro tyto účely začaly vyrábět CPU s jednotkou správy paměti (Memory Management Unit). DRAM je nejvíc používaná vnitřní paměť.

Z nevolatilních typů pamětí si zmíníme dva:

- NVRAM (neboli **N**on-**V**olatile RAM) je přesně to, co si myslíte, že je. Je to RAMka (buď statická nebo dynamická), která uchovává data i po odpojení od proudu. Jako vnitřní paměť počítače se nepoužívá hlavně z toho důvodu, že potřebuje baterii, která se tudíž musí časem měnit, a obrovská zátěž, kterou CPU klade na RAM, by ji vyčerpala v řádu dní. Největší využití zaznamenala jako úložiště BIOSu (jakoby operačního systému základní desky) a jako *flash* paměť v SSD discích.
- ROM (**R**ead-**O**nly **M**emory) je typ paměti, do které nelze ukládat data, ale pouze je z ní číst. Používá se hlavně jako úložiště firmwaru v různých typech externích zařízení (myši, klávesnice, tiskárny, ...). Firmware je vlastně takový malý operační systém, přes který posílá dané zařízení signály do CPU připojeného počítače.

Primárním smyslem RAM je být úložištěm pro CPU instrukce. Představíme-li si RAM jako jeden dlouhý sloupec, kde v řádcích jsou postupně instrukce, může jeden uložený počítačový program vypadat třeba takhle:



Příklad programu v RAM. Červená čísla představují *adresy* v paměti – prostě jenom říkají, kolikátá (v dvojkové soustavě) pozice v paměti to je. Po instrukci JUMP přečte CPU další řádek, kde je uloženo, na které místo v paměti má skočit. Tam najde instrukci ADD, po které očekává dvě čísla, která má sečíst. Nakonec narazí na instrukci STORE a výsledek předchozího součtu na následující místo v paměti uloží.

Základní deska

Základní deska (angl. motherboard) je zase jen elektrický obvod, který spojuje všechny komponenty počítače. Obvykle se dělí na dva kusy – severní (North Bridge) a jižní (South Bridge).

- North Bridge obsahuje sloty pro CPU a RAM a často taky porty pro propojení s I/O zařízeními (myšmi, klávesnicemi, tiskárnami, monitorem,...).
- South Bridge má sloty pro externí paměti a grafické/zvukové karty a také je

v něm umístěna NVRAM s BIOSem a její baterkou. BIOS (**B**asic **I**nput/**O**utput **S**ystem) je software zajišťující přenos dat mezi komponenty a CPU.

GPU

GPU (**G**raphics **P**rocessing **U**nit) je vlastně CPU ořezané o většinu svých funkcí. Primárním účelem GPU je provádět co nejvíce výpočtů najednou co největší rychlostí. CPU mívá až 32 výpočetních *jader* (jednotek schopných provádět výpočty najednou během jednoho úderu vnitřních hodin); moderní GPU mají výpočetních jader tisíce. Často umožňují posílat výstup přímo do monitoru a mají v sobě též vlastní RAMku, kam ukládají mezivýsledky, které není potřeba posílat zpět do CPU.