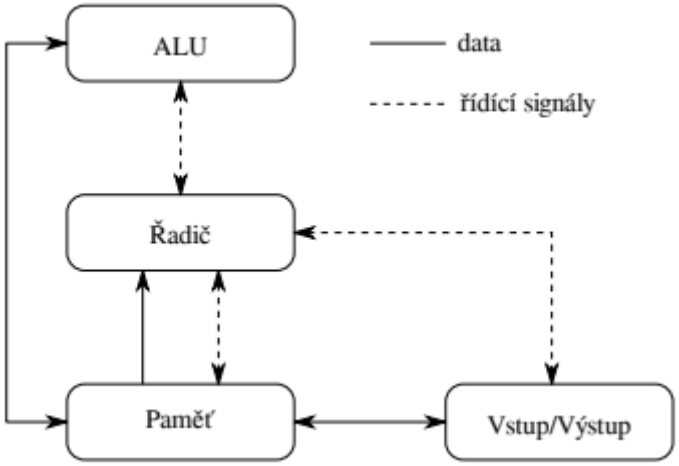
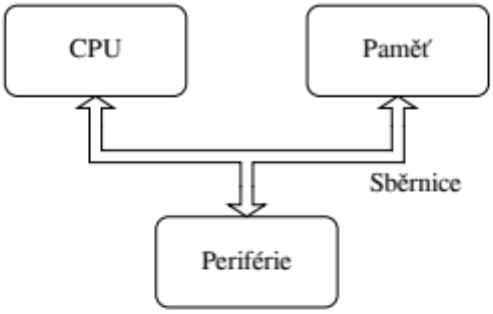
Architektura počítačů

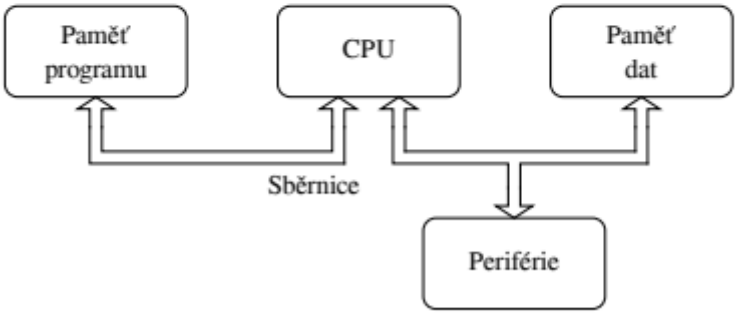
- Propojení menších částí, které spolu navzájem komunikují vyměňují si data atd… tímto se skládají do celku
- Podstatné vlastnosti počítačů:
    - Struktura, uspořádání: popis jednotlivých funkčních částí a jejich propojení
    - Součinnost, interakce: popis řízení dynamické komunikace mezi funkčními bloky
    - Realizace, provedení: popisuje vnitřní strukturu jednotlivých funkčních bloků
    - Funkcionalita, činnost: výsledné chování počítače
- Každý člověk nahlíží na zmíněná hlediska jinak, pro většinu je důležité poslední zmíněná vlastnost
- Pro pochopení je důležité chápat všechny části
- John von Neumann definoval v roce 1945 koncepci počítace EDVAC (electronic discrete variable automatic computer) – tým odborníků z Pensylvánie.
- Tento základ sloužil pro koncepci chápání počítače řízeného obsahem paměti
- Tento model byl několikrát modifikován a existovaly i jiné modely, ale většina moderních počítačů odpovídá tomuto modelu.



- Byly nastaveny principy, které byly univerzální:
    - PC se skládá z paměti, řídící jednotky, aritmetické jednotky a výstupu/výstupu
    - Struktura je neměnná na rozdílné úlohy, záleží na tom, jak budeme zacházet s pamětí
    - Kroky jsou lineární a závislé na sobě
    - Paměť je rozdělená do stejně velkých buněk, jsou označovány adresami
    - Program je posloupnost instrukcí, které se provádějí jednotlivě v pořadí, v jakém byly zapsány do paměti
    - Změna pořadí je možná pomocí podmíněných a nepodmíněných skoků
    - Instrukce, čísla, adresy a znaky jsou reprezentované v binární soustavě
- Na Neumanna navázal tým z Harvardu, který vytvořil harvardské schéma, které je v zásadě hodně podobné
- Odstraňovalo nedostatky – rozdělení paměti pro data a program (např RAM a HDD/SSD)
- Zjednodušené schémata pro obě zmíněná schémata:



Obrázek 2: Počítač podle von Neumanna



Obrázek 3: Harvardská architektura počítače

- Hlavní nedostatek obou koncepcí je sekvenční vykonávání instrukcí
- Díky tomuto je lehké implementovat systém, ale zamezuje paralelnosti (všechny věci co se v počítači dějí nejsou paralelní, vypadá že jo, ale pouze se dějí velmi rychle po sobě)
- Paralelismus se tedy simuluje v operačním systému
- Další problém jsou sběrnice, které nedovolují přistupovat do více míst v paměti najednou + data mohou cestovat pouze jedním směrem (přijímat/odesílat)
- Von Neumann
    - Good:
        - Řídící jednotka procesoru přistupuje do paměti pro data o pro instrukce jednotným způsobem
        - Jednoduchá výroba kvůli sběrnici
    - Bad:
        - Společné uložení dat a kódu je nebezpečné pro programátora, může si přepsat vlastní program
- Harvard
    - Good:
        - Oddělení paměti dat a programu má hodně výhod:
            - Program nemůže přepsat sebe sama
            - Paměti mohou mít jiné technologie
            - Každá paměť může mít jinou velikost její nejmenší adresovatelné jednotky
            - Dvě sběrnice umožnují minimální paralelismus, kde můžeme přistupovat k instrukcím i datům zároveň
        - Bad:

- Dvě sběrnice jsou náročnější na vývoj řídící jednotky procesoru, takže zvyšuje náklady na výrobu
- Nevyužitou část paměti nelze použít pro program a obráceně

- byte [B] - 8 bitů, slabika, obvykle nejmenší adresovatelná jednotka
- $1kB = 2^{10}B = 1024B$
- $1MB = 2^{20}B = 1048576B$
- $1GB = 2^{30}B = 1073741824B$
- $1TB = 2^{40}B = 1099511627776B$

- V praxi se zaokrouhluje 3 na 10 = 10 na 3, 1024 B -> 1000 B
- Tato odchylka koreluje s počtem dat, odchylka je cca 5% u MB a cca 7% u GB

- Vývoj počítačů se dělí do několika období, které jsou většinou pojmenované po technologii, kterou obnášeli
- Tento krok však nebyl pro výrobu podstatný, sloužil k obchodní propagaci

Generace	1	2	3	3.5	4
Rok	1951	1957	1964	1971	1981
Prvky	elektronky	tranzistory	SSI	MSI	LSI
Paměť	buben	ferity	ferity	MSI	LSI
Kapacita paměti	1kB	10kB	1MB	1MB	10MB
Řízení periférií	CPU syn.	CPU as.	kanály	adaptéry	perif. proc

- Z tabulky je vidět, patrný vývoj v oblasti komunikace s periferiemi, rychle rostoucí číslo operační paměti

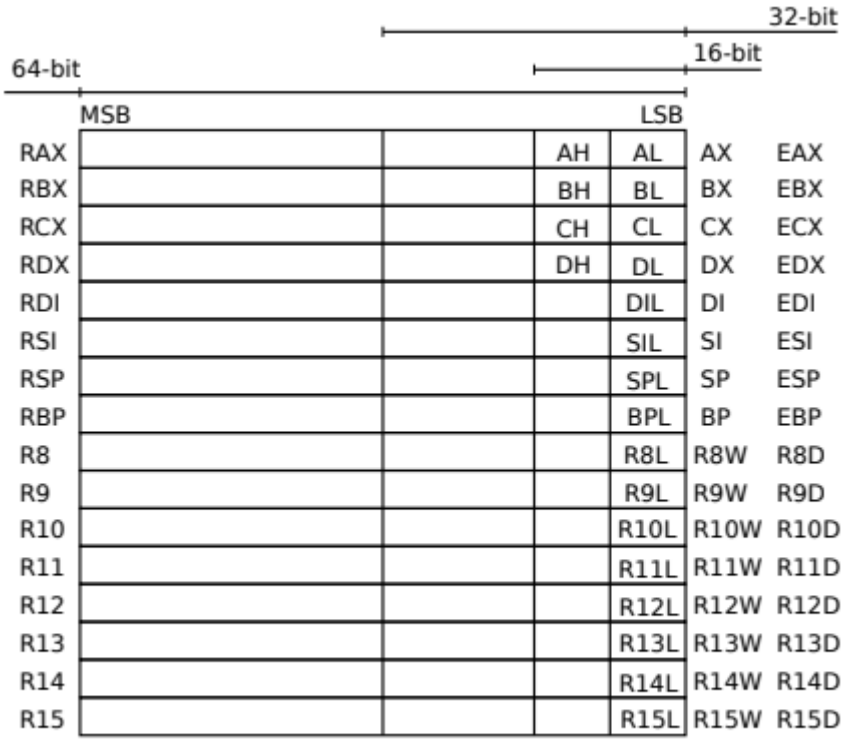
Otázky:

- Jakle jsou základní principy fungovaní počítače.
  - PC se skládá z paměti, řídící jednotky, aritmetické jednotky a výstupu/výstupu
  - Struktura je neměnná na rozdílné úlohy, záleží na tom, jak budeme zacházet s pamětí
  - Kroky jsou lineární a závislé na sobě
  - Paměť je rozdělená do stejně velkých buněk, jsou označovány adresami
  - Program je posloupnost instrukcí, které se provádějí jednotlivě v pořadí, v jakém byly zapsány do paměti
  - Změna pořadí je možná pomocí podmíněných a nepodmíněných skoků
  - Instrukce, čísla, adresy a znaky jsou reprezentované v binární soustavě
- Jaké má výhody a nevýhody von Neumann
  - Levnější na výrobu než harvardský
  - Přistupuje k datům a instrukcím stejným způsobem
  - Společné uložení dat a kódu je nebezpečné pro programátora, může si přepsat vlastní program
- Přinesla harvardská architektura nějaká vylepšení oproti von Neumannovi
  - Jednoduchý paralelismus, může zároveň vykonávat instrukce a přistupovat do paměti

    - Paměti můžou mít jinou velikost, takže podporují jiné druhy výroby
    - Program nemůže přepsat sebe sama
- Jaká je podpora paralelismu u obou architektur
    - Von Neumann – nic moc, žádná
    - Harvardská – Jednoduchý paralelismus, může zároveň vykonávat instrukce a přistupovat do paměti
- Je lepší mít oddělenou paměť pro data i program
    - Je lepší mít paměť oddělenou, programátor si s ní pak může zacházet, jak chce a není zde nebezpečí přepsání vlastního kódu, paměť mohou mít rozdílné technologie (mohou mít jinou minimální velikost adresovatelné buňky), jednoduchý paralelismus
- Může fungovat počítač bez paměti či periferií
    - Nemůže, celý základ toho, jak funguje počítač, je založený na používání více pamětí
- K čemu se v PC používá 2 soustava
    - Instrukce, čísla, adresy a znaky
- Zvyšují sběrnice výkon
    - Ne slouží jenom jako pomyslné médium mezi komponentami pro předávání dat
- Je možné, aby procesor prováděl instrukce jinak než sekvenčně
    - Pro Neumanna a Harvard schéma ne, jediné, co jde udělat jsou (ne)podmíněné skoky
- Jak je v PC organizovaná paměť
    - Rozdělení na: operační paměť, data, a paměti pro kód
    - Pokud jde o to více do hloubky, tak je pamět uložená na bloky (statické a dynamické)

Assembler pls end my suffering

- Assembler používá JSI – jazyk strojových instrukcí
- JSI -> strojový kód
- Hlavní: AMD, Intel x86 64-bit, celočíselné ALU
- První 64bit -> AMD, rozšíření ze sady 32bitů
- Registry byly zvětšeny a několik bylo přidáno -> 16 pracovních registrů



Obrázek 1: Přehled registrů 64bitového procesoru x86

Registry 64bitové pro všeobecné použití:

`RAX, RBX, RCX, RDX, RSI, RDI, RBP, RSP, R8 až R15.`

Registry 32bitové:

výše uvedené registry dovolují přístup ke své dolní 32bitové části přes:
`EAX, EBX, ECX, EDX, ESI, EDI, EBP, ESP, R8D až R15D.`

Pozor! Při zápisu do 32bitové části registru se horní část registru Rxx automaticky vynuluje!

Registry 16bitové:

výše uvedené registry dovolují přístup ke své dolní 16bitové části přes:
`AX, BX, CX, DX, SI, DI, BP, SP, R8W až R15W.`

Registry 8bitové:

první čtyři 16bitové registry jsou rozděleny na horní a dolní 8bitové části: `AH` (high), `AL` (low), `BH, BL, CH, CL, DH, DL.`

Další 8bitové registry jsou:

`DIL, SIL, SPL, BPL, R8L až R15L.`

-

- Na obrázku chybí pár registrů:
- Čítač instrukcí RIP (IP):
  - Ukazuje na aktuální vykonávanou instrukci. Jeho změny se provádí skokovými instrukcemi
- Stavový registr (FLAG):
  - Obsahuje stavové bity, které se využívají prostřednictvím podmíněných instrukcí, mezi nejpoužívanější patři ZF (zero flag), CF (carry), OF (overflow), SF (signum), DF (direction), další implementované jsou PF (parity) a AF (auxiliary) bit.
- Carry flag
  - Pokud při aritmetické operaci výsledné číslo překročí max hodnotu, tak vznikne přenos/výpůjčka do vyššího řádu. Jinak je tento bit vynulován. Tento bit značí přetečení aritmetiky bezznaménkových čísel. Používá se pro větší přesnost
- Overflow Flag
  - Pokud je výsledek moc velký (kladný záporný) a není jej možno vložit do cílového operandu, jinak je nulovaný, používaný u aritmetiky znaménkových čísel
- Sign flag
  - Nastavuje hodnotu nejvyššího bitu, ukazuje, zda je číslo kladné nebo ne (0 – kladné | 1- záporné)
- Zero flag
  - Nečekaně, bývá nastaven, když je výsledek strode.
- Registry jsou pouze v některých případech vázány účelově.
- Příklady:
  - RAX, EAX, AX, AL -> akumulátor -> násobení, dělení a řetězové instrukce
  - RCX, ECX -> čítač v pomyslných loopech
  - CL -> je použit jako čítač při bitových rotacích a posunech
  - RDX, EDX, DX -> je horní část argumentu při násobení a dělení
  - RSI, RDI -> indexový registr v řetězových instrukcích

Adresování

- Přímé X nepřímé
- Přímé -> pevná adresa -> moc se nepoužívá -> např globální proměnné
- Pokud adresu nelze určit přímo, lze použít adresování nepřímé přes registry
- Adresování pomocí dvou registrů:
- [Bázový + Indexový * Měřítko + Konstanta]
- Bázový a Indexový jde použít jakýkoliv 64 bitový registr
- Měřítko, bývá číslo 1,2,4,8 pro práci s polem (kvůli velikosti proměnných word)

Datové typy

- Nejsou nějak typové, znaménkové ani nic
- Pouze velikost v bitech
  - DB – data BYTE (8 bitů)
  - DW – data WORD (16 bitů)
  - DD – data DWORD (32 bitů)
  - DQ – data QWORD (64 bitů)
- Zkratka při deklarace, celé jméno v instrukcích kde se určuje velikost operandu

Spojování programů

- C–C
- Pokud chceme „forwardovat“ funkce/proměnné, musíme použít klíčové slovo extern
- Linker musí vědět o všech proměnných atd… program kolikrát půjde přeložit, ale linker jej nedovede spojit dohromady
- Je důležité dobře používat:
  - Veřejné symboly
  - Lokální symboly
  - Externí symboly
- JSI-C

```
    ; module.asm
    bits 64                         ; 64 bit code
    section .data                   ; data section
    global g_module_counter         ; public symbol
g_module_counter    dd 0            ; variable g_module_couter
g_module_sum        dd 0            ; variable g_module_sum

    section .text                   ; code section
    global inc_sum                  ; public symbol

inc_counter:                        ; function inc_counter
    inc dword [ g_module_counter ]; g_module_counter ++
    ret

inc_sum:                            ; function inc_sum
    inc dword [ g_module_sum ]      ; g_module_sum ++
    call inc_counter                ; inc_counter ()
    ret
```

- Rozdělení na datovou část a kód, v section data použijeme extern definujeme proměnné/funkce, které bereme z C souboru musí být extern
- Přesuny v paměti pomocí MOV, MOVSX, MOVZX
- MOV cíl, zdroj || cíl = zdroj
- Lze použít Registry Memory a Konstanty
- Možné přesuny

```
MOV R, R                ; přesun registru do registru
MOV R, M                ; přesun paměti do registru
MOV M, R                ; přesun registru do paměti
MOV R, K                ; přesun konstanty do registru
MOV M, K                ; přesun konstanty do paměti
```

- Pravidla:
  - Stejná velikost operandů
  - Nelze použít dva paměťové operandy
  - Vždy kdy je použitý registr, tak velikost přesunu určuje registr
  - Pokud není použit v přesunu registr, musí se ručně určit DWORD atd…
- U MOVSX a MOVZX lze použít pouze dvě kombinace

```
MOV R, R                ; přesun registru do registru
MOV R, M                ; přesun paměti do registru
```

- U obojího platí, že velikost cílového operandu musí být větší než zdrojového

- MOVZX provede rozšíření pomocí nuly do vyššího řádu
- MOVSX provede rozšíření pomocí znaménka
- Přesun mezi dvěma proměnnýma přes registr
- Pokud chceme kupříkladu přenést int do longu, musíme jej převést použitím MOVSX/ZX na „delší“ proměnnou a pak jí můžeme vložit do long proměnné
- Indexování v poli:

```
set_int_array_index :
    enter 0,0

    movsx rax, dword [ g_index ]     ; 64-bit address
    mov dword [ g_int_array + rax * 4 ], 646464
                                     ; g_int_array [ g_index ] = 646464
```

-
- Instrukční soubor procesoru i486 + následovníci se v praxi využívá ani ne polovina všech instrukcí celočíselné jednotky ALU
- Tyto instrukce se řadí do
    - Přesunových
    - Logické a bitové
    - Aritmetické
    - Skokové
    - Pomocné a řídící
- Všechny tyto instrukce jsou v dokumentaci daného procesoru -> dokumentace je však nepoužitelná pro běžného smrtelníka, pouze pan Profesor Petr Olivka ji rozumí :)
- Jsou i zkrácené verze dokumentace, která je více user friendly
- Přesunové instrukce:
    - MOV cíl, zdroj
        - Basic přesun
        - Stejná velikost operandů
        - Pouze pro pro paměť, registry a konstanty
    - CMOVcc cíl, zdroj
        - Přesun pod nějakou podmínkou (cc – je, nz, jg atd…)
        - Stejná velikost operandů
        - Přesun pro paměť nebo registr do registru
    - MOVZX cíl, zdroj
        - Přesun, pokud je zdroj menší než cíl
        - Provede se rozšíření nulou
    - MOVSX cíl, zdroj
        - Stejné jako MOVZX, ale provádí se znaménkově
    - XCHG cíl, zdroj
        - Vymění obsah mezi operandy
    - BSWAP cíl
        - Prohodí pořadí bitů
        - Konverze mezi little a big endian
    - PUSH zdroj
        - Uloží obsah zdroje na vrchol zásobníku a posune vrchol zásobníku
    - POP cíl
        - Uloží vrchol zásobníku do cíle a sníží vrchol zásobníku

- Logické a bitové instrukce:
  - AND cíl, zdroj
    - Bitové porovnání cíl = cíl && zdroj
  - TEST cíl, zdroj
    - AND ale výsledek se neukládá
  - OR cíl, zdroj
    - Bitové porovnání cíl = cíl || zdroj
  - XOR cíl, zdroj
    - Bitové porovnání cíl = cíl XOR zdroj
  - NOT cíl
    - Bitová negace operandu
  - SHL/SAL cíl, kolik
    - Bitový i aritmetický posun doleva (násobení dvěma)
    - Operand „kolik“ je konstanta nebo registr CL
  - SHR cíl, kolik
    - Bitový posun doprava (dělení dvěma)
    - Operand „kolik“ je konstanta nebo registr CL
  - SAR cíl, kolik
    - Aritmetický posun doprava (dělení dvěma)
    - Operand „kolik“ je konstanta nebo registr CL
  - ROL cíl, kolik
    - Bitová rotace doleva
    - Operand „kolik“ je konstanta nebo registr CL
    - Rozdíl od posunu je takový, že pokud bit přeteče z hranice, připojí se opět na konec bitového řetězce
  - ROR cíl, kolik
    - ROL akorát doprava
  - RCL cíl, kolik
    - Bitová rotace doprava pomocí CF operandu
    - Operand „kolik“ je konstanta nebo registr CL
  - RCR cíl, kolik
    - RCL akorát doprava
  - BT cíl, číslo
    - Zkopíruje do CF hodnotu bitu daného operandem číslo z operandu cíl
  - BTR cíl, číslo
    - Zkopíruje do CF hodnotu daného operandem číslo z operandu cíl a nastaví jej na nulu
  - BTS cíl, číslo
    - BTR akorát nastaví na jedničku
  - BTC cíl, číslo
    - BTR akorát CF neguje
  - SETcc cíl
    - Nastaví hodnotu cíle na 1/0 podle toho, zda je splněna podmínka (cc – stejně jako u podmíněného přesunu)
  - SHRD/SHRLD cíl, zdroj, kolik
    - Provede nasunutí „kolik“ bitů ze zdroje do cíle, zdroj se nemění

Aritmetické instrukce

- ADD cíl, zdroj
  - Aritmeticky cíl += zdroj
- ADC cíl, zdroj
  - To stejné jako ADD ale sčítá se i CF cíl += zdroj + CF
- SUB cíl, zdroj
  - Aritmetické odčítání cíl -= zdroj
- CMP cíl, zdroj
  - Používá se u podmíněných skoků
  - Vrátí 1/0 jestli jsou čísla stejné
  - Používá aritmetické odečítání
- SBB cíl, zdroj
  - Aritmetické odčítání s výpůjčkou CF cíl -= zdroj – CF
- INC cíl
  - Inkrementace neboli i++
  - Nemění CF
- DEC cíl
  - Snížení o jedničku
- NEG cíl
  - Změna znaménka operandu
- MUL zdroj
  - Násobení dvou bezznaménkových čísel
  - Velikostí zdroje se násobí akumulátor (AL, AX, EAX)
  - Výsledek se uloží do (AX, AX-DX, EAX, EDX)
- IMUL zdroj
  - Stejné jako MUL ale pro znaménkové čísla
- DIV zdroj
  - Dělení dvou bezznaménkových čísel
  - Velikostí (8, 16, 32) zdroje se dělí hodnota v (AX, AX-DX, EAX-EDX)
  - Výsledek se uloží do (AL, AX, EAX)
  - Výsledek po dělení uloží do (AH, DX, EDX)
- IDIV zdroj
  - Stejné jako DIV ale pro znaménkové čísla
- CBW
  - Znaménkové rozšíření registru AL do AX, používá se před znaménkovým dělením
- CWD
  - Znaménkové rozšíření registru AX do AX-DX, používá se před znaménkovým dělením
- CDQ
  - Znaménkové rozšíření registru EAX do EAX-EDX, používá se před znaménkovým dělením
- CQO (64-bit režim)
  - Znaménkové rozšíření registru RAX do RAX-RDX, používá se před znaménkovým dělením

Skokové instrukce

- JMP cíl
  - Přesune tok programu na návěští dané jménem cíle
  - Může však jít i o paměť či adresu
- CALL cíl
  - Volání podprogramu
  - Stejné jako JMP ale na vrchol zásobníku se uloží adresa instrukce následující za CALL
- RET N
  - Odebere adresu následující instrukce z vrcholu zásobníku
  - Když se program vrátí z podprogramu, můžeme přeskočit nějaké N bitů instrukcí
- LOOP cíl
  - Řízený cyklus pomocí podmínky, počet opakování je dán registrem ECX
  - ECX se dekrementace vždy před vyhodnocením podmínky
- LOOPE/Z cíl
  - if (--ECX && ZF) goto cíl
- LOOPNE/NZ cíl
  - if (--ECX && !ZF) goto cíl
- JCXZ cíl
  - Provede skok na požadované místo, pokud je registr ECX (CX) nulový
- Jcc cíl
  - Skupina podmíněných skoků
  - JZ/E, JNZ/NE, JS, JNS, JC, JNC, JO, JNO testují jednotlivé bity v registru
  - Skupina pro porovnávání čísel

    `JB/JNAE/JC` - menší než, není větší nebo rovno,

    `JNB/JAE/JNC` - není menší, větší nebo rovno,

    `JBE/JNA` - menší nebo rovno, není větší,

    `JNBE/JA` - není menší nebo rovno, je větší,

    `JL/JNGE` - menší než, není větší nebo rovno,

    `JNL/JGE` - není menší, větší nebo rovno,

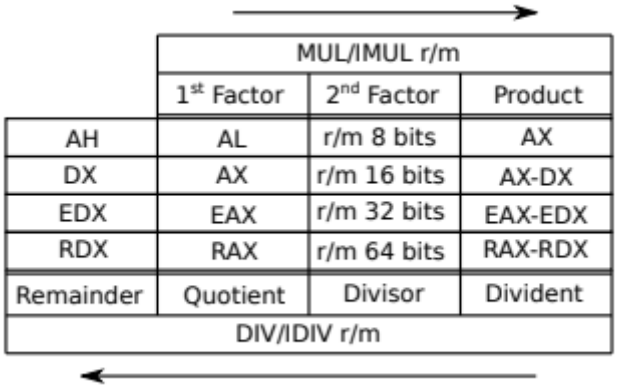
    `JLE/JNG` - menší nebo rovno, není větší,

    `JNLE/JG` - není menší nebo rovno, je větší.
  - A (above), B (below), L(less), G(greater), N(not), E(equal)

Pomocné řídící instrukce

- CLD
  - Nastaví DF na nulu
- STD
  - Nastaví DF na jedničku
- CLC
  - Nastaví CF na nulu
- STC
  - Nastaví CF na jedničku
- CMC
  - Negace CF
- NOP – prázdná instrukce

Instrukce násobení a dělení



	MUL/IMUL r/m		
	1st Factor	2nd Factor	Product
AH	AL	r/m 8 bits	AX
DX	AX	r/m 16 bits	AX-DX
EDX	EAX	r/m 32 bits	EAX-EDX
RDX	RAX	r/m 64 bits	RAX-RDX
Remainder	Quotient	Divisor	Divident
DIV/IDIV r/m			

- Obrázek pro „názornost I/MUL a I/DIV“ *topkek*
- Při chápání násobení je třeba se dívat nahoře vpravo -> doleva dolů
- EHM DÁL TO NECHÁPU ANI PÍČU, STRANA 24 SKRIPTA

Příklady použití instrukcí

- https://poli.cs.vsb.cz/edu/apps/soj/down/apps-soj-skripta.pdf
- Stránky 25 – 30 (cca 5 příkladů)

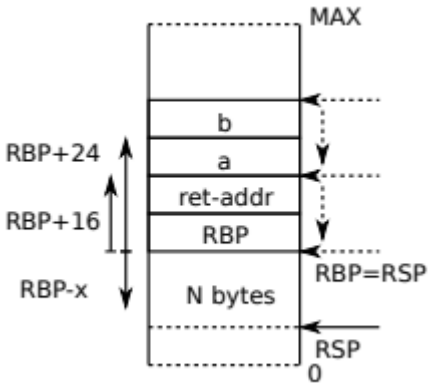
64bitové C – JSI rozhraní

- 64bit -> FPU jednotka nahrazena výhradně SSE
  - 8 bitů – AL
  - 16 bitů – AX
  - 32 bitů – EAX
  - 64 bitů – RAX
  - 128 bitů – RAX-RDX
  - Float/double – registr XMM0
- RDI, RSI, RDX, RCX, R8 a R9 slouží k předávání parametrů, mohou se ve funkci libovolně měnit
- RAX, R10 a R11 je možné v kódu měnit
- RSP a RBP slouží pro práci se zásobníkem, musí být upraveny na konci používání na původní hodnotu
- RBX, R12, R13, R14 a R15 musí být obnoveny do default hodnotu
- DF musí být na konci funkce vráceno na 0
- Registry FPU – ST0 až ST7 a registry SSE – XMM0 až XMM15 je možné ve funkci použít a nemusíme obnovit jejich hodnoty

Volání funkcí

- Používá se kombinace předávání přes registry a zásobník
- Pro předávání celočíselných parametrů a ukazatelů se využívá pro prvních šest parametrů zleva šestice registrů
  - RDI, RSI, RDX, RCX, R8 a R9
- Pro float/double se předávají přes registry SSE
  - XMM0 až XMM7
- V registru AL musí být uveden počet parametrů předávaných přes XMM0-9 registry

- Další parametry se předávají přes zásobník
- V zásobníku jsou všechny hodnoty 64bitové



-
- Obrázek je pro enter N,0
- https://poli.cs.vsb.cz/edu/apps/soj/down/apps-soj-skripta.pdf
- Příklady na zásobník (str. 33–40)

Číslicové obvody – výroba

- Dle druhu výroby se rozlišují číslicové obvody na:
  - Hybridní
  - Monolitické
- Hybridní obvody:
  - Obsahují pasivní i aktivní součástky, které se připevní na jednu nosnou destičku, propojí se a zapouzdří
  - Používají se hlavně u číslo-analog analog-číslo převodu
- Monolitické obvody (neboli čipy)
  - Všechny potřebné prvky jsou soustředěné na polovodičové destičce (křemík)
  - Aktivní i pasivní prvky
  - Většina elektronických číslicových systémů je vyrobena tímto způsobem
- Rozdělení dle stupně integrace
  - SSI (Small scale integration) – do 30 prvků
  - MSI (Middle scale integration) – do 1000 prvků
  - LSI (Large scale integration) – do 100k prvků
  - VLSI (Very large scale integration) – do 10 milionů prvků
  - ULSI (Ultra large scale integration) - do 1 miliardy prvků
  - GSI (Gigantic scale integration) – nad 1 miliardu prvků

Bipolární technologie

- Používají bipolární tranzistory
- Rychlejší obvody
- Né tak velká integrace
- Větší spotřeba ale jsou levnější

DTL

- Dnes už nepoužívané obvody
- Vůbec co číči nevím, co po mě chce třeba xD

TTL

- Tranzistor Tranzistor Logic
- Vícenásobný emitor, který realizuje logické funkce
- Většinou příznivější podmínky než DTL

Unipolární tranzistory

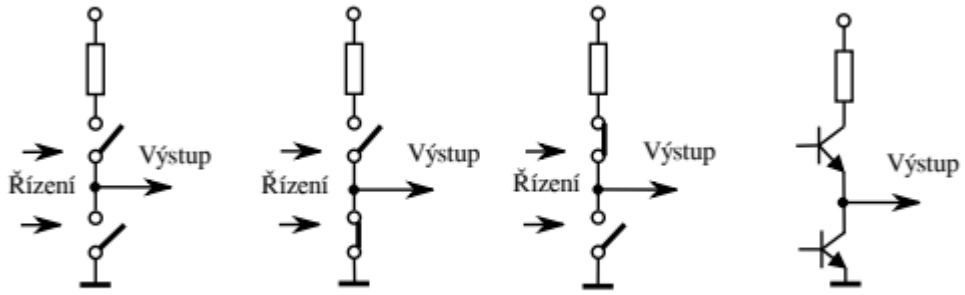
- Tranzistor řízený polem - FET
- Myšlenka je taková, že hloubka vniku do pole by měla být uměrná převrácené hodnotě konduktivity -> čím lepší konduktivita, tím lepší vniknutí do pole -> izolanty jsou nevhodné
- Pokud bude držet konduktivitu nízko, tak se i tak může do pole vniknout a ovlivňovat tím proud
- Tato proudová dráha se nazývá kanál ->

Komunikace s periferiemi

- Sběrnice = něco jako přenosník dat mezi částmi počítače
- Aby nedošlo ke kolizi atd.., je nutno určit předem některé věci
    - Který blok bude informaci sběrnici dodávat a který snímat
    - Kdy je informace na sběrnici platná
- Největší část celého tohoto procesu by měl být mikroprocesor
- Mikroprocesor určuje další údaje přenosu
    - Směr přenosu
    - S kým komunikuje
    - Kdy jsou informace platné
    - Další nutné údaje
- Sběrnice se dělí na:
    - Adresové
        - Zdrojem informací je mikroprocesor
        - Počet bitů/vodičů odpovídá počtu bitů adresy
        - Mikroprocesor vytvoří adresu a tím určuje využitelný adresovatelný prostor
        - Univerzální mikroprocesory mají dva adresové prostory
        - Jeden pro adresování paměti a druhý pro adresování I/O
        - Tyto prostory nejsou ty stejné a jejich rozlišení zajišťuje řídící sběrnice
    - Řídící
        - Souhrn signálů, které jsou aktivní v různých časech a významech
        - Různé zdroje
        - Signály dovede pouze k bloku, kam patří
        - Nejčastější signály
            - RESET – základní stav mikroprocesoru

- Memory Read (MR) – Zabezpečuje časování čtení z pamětí jiných bloku do mikroprocesoru
- Memory Write (MW) - Zabezpečuje časování zápisu z pamětí jiných bloku do mikroprocesoru
- Input/Output read/write (IOR/IOW) – zápis nebo čtení do zařízení
- READY – připravenost obvodu
-

- Datové
  - Přenos všech dat v PC
  - Vždy komunikují dva bloky (Paměť -> mikroprocesor)
  - Mikroprocesor se účastní jako přijímač a vysílač všech procesů v PC
  - V jakémkoliv okamžiku musí být aktivní pouze jeden vysílač
  - Jinak dojde k takzvané neurčitosti signálu nebo v horším případě ke zničení jednoho nebo obou vysílajících obvodů
  - Proto je nutné opatřit bloky které komunikují s datovou sběrnicí obvodem, který zajistí odpojení bloku od datové sběrnice -> **Třístavové budiče sběrnice**
  - Tyto budiče mohou být jednosměrné nebo obousměrné
  - Budiče také nemají pouze odpojovací činnost, zároveň posilují výkon



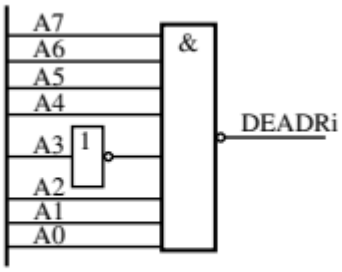
Stav vysoké impedance | Logický výstup 0 | Logický výstup 1 | Realizace

-
  - Nejdůležitější parametry datové sběrnice:
    - Šířka (počet bitů)
    - Časování
  - Šířka sběrnice ukazuje, kolik bitů je schopna sběrnice přenést
  - Šířka výrazně ovlivňuje rychlost sběrnice
  - Šířka sběrnice by neměla být větší jak šířka procesoru
  - Multiplexování sběrnic – může ušetřit počet vodičů
    - Vedení signálů dvou sběrnic (často adresové a datové) ve společných vodičích
    - Tyto signály nikdy nejsou aktivní ve stejnou dobu
    - Signály řídící sběrnice určují význam právě přenášených dat na multiplexované sběrnici
  - Sběrnice rozeznáváme
    - Vnitřní sběrnice mikroprocesoru
      - Architektura dána architekturou mikroprocesoru
    - Vnitřní sběrnice počítače
      - Propojuje mikroprocesor s ostatními prvky počítače
      - Dovoluje vytvoření různých uspořádání počítače
      - Optimální cenově a funkčně pro danou aplikaci

- Vnější sběrnice počítače
    - Pro styk s okolím a multiprocesorem
    - Vyskytuje se ve složitějších systémech
    - Sběrnice jsou rozšířeny a rozhodují o prioritě podřízených zařízení
    - Priorita se mění dynamicky podle aktuálních potřeb systému
    - Pomocí vnější sběrnice se k PC připojují přídavná zařízení

Připojení zařízení ke sběrnici, adresový dekodér

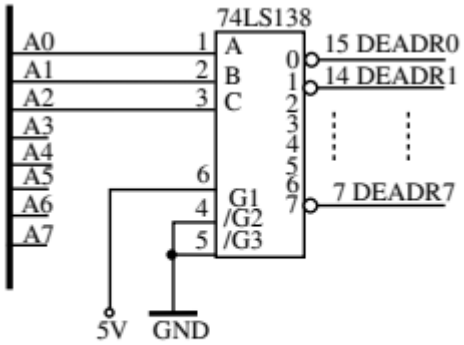
- Přenosy dat pomocí
    - Řídící technické prostředky
        - Mikroprocesor se podílí jen trochu nebo vůbec
        - Řízeno řadičem přímého přístupu do paměti (DMA)
        - Používá se pro přenos velkého množství dat
        - Např. přídavné zařízení -> RAM
    - Programové řízení
        - Výhradně mikroprocesor
        - Provádí instrukce setřízené do segmentu řídícího programu, případně mezi vnitřním registrem a buňkou vnější paměti
        - Výhoda: Levná realizace, lehká úprava algoritmu změnou třídícího algoritmu
        - Nevýhoda: Pomalý přenos na pomalých mikroprocesorech
        - Přenosy mezi vnitřními a vnějšími registry
            - Paměťové
            - Vstupně/výstupní
        - Výběr prostor je určen datovými signály pro určitý prostor
- Umístění periferií do nějakého z prostoru se nazývá mapování
- Výhody mapování periférií do V/V
    - Rychlejší přístup do prostoru
    - Díky instrukcím vstupu a výstupu IN a OUT
- Výhoda mapování do paměťového systému, je větší spektrum použitelných instrukcí (k dispozici jsou všechny instrukce pro práci s pamětí)
- Mapa paměti – zobrazení obsazení dílčích úseků paměťového prostoru jednotlivými pamětmi
- Paměťový prostor bývá většinou zaplněn více než jednou fyzickou pamětí nebo periferním zařízením, proto musí mikroprocesor rozhodnout, jaké zařízení je ke komunikaci určeno
- Na toto se používá **adresový dekodér**
- Jeho výstup je signál CS (Chip Select) pro jednotlivé obvody
- CS signál připojí vybraný obvod k datové sběrnici a sběrnici přepne ze stavu vysoké impedance do aktivního stavu (viz obrázek stránku nahoru)
- Adresový dekodér může být stavěn jako dekodér pro:
    - Úplné dekódování adresy



Obrázek 2: Princip úplného dekodéru adresy

- Unikátní adrese ADRi je přiřazen jeden signál DEADRi a obráceně
- Např AND na obrázku výše

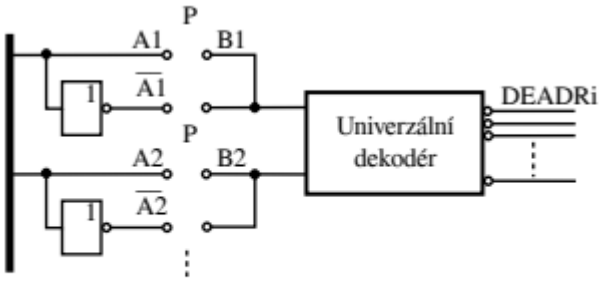
- Neúplné dekódování adresy



Obrázek 3: Princip neúplného dekodéru adresy

- Jistému signálu DEADRi přísluší adresový prostor ADRi i = 1,2,3…
- Tento dekodér je levnější, ale je nutné, aby se adresy pozorně přiřazovali k dotyčným obvodům
- Použití: Přiřazování paměťových prostorů k jednotlivým typům paměti, přidělování k periferiím v systému

- Lineární přiřazování adresy
  - Řády adresy jsou přímo pokládané za výstupní výběrové signály DEADRi = ADRi
  - Nejlevnější adresový dekodér
  - Nevýhoda je, že je možno připojit pouze M přídavných zařízení o adresách s N řády
- Univerzální přiřazení dekódovaných adres



Obrázek 4: Princip univerzálního dekodéru adresy

-
- „Custom“ pro určité řešení speciálního problému
- Na každé adresní lince má vložen invertor a přepínač P (DIP)
- Zde je možné libovolně zvolit skutečnou vstupní adresu
- Nastavování probíhá před zasunutím modulu do PC - krinž

- Tyto typy dekoderů patří mezi základní a hodně používané
- Lze udělat custom dekodéry atd… ale samozřejmě to bude dražší
- Jeden takový je vidět v principu univerzálního dekodéru nahoře

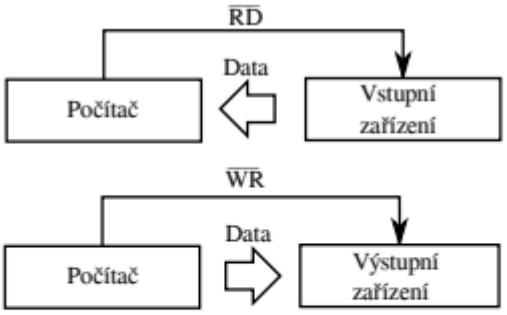
Řízení komunikace

- Dva případy zahájení komunikace počítače/mikroprocesoru s periferiemi
  - Iniciativa programu
    - Instrukce v programu volá pro začátek výměny dat
    - Například algoritmus programu, který potřebuje přijímat/vysílat s okolím informace
    - Je nutné, aby periferie byla v danou chvíli schopná přijmout/vyslat informaci anebo musí být schopný v daný moment vystavit nějakou informaci
    - Například spínače nebo sdělovače
  - Iniciativa periferie
    - Inicializuje se, za předpokladu, že periferie potřebuje přijmout nebo vyslat nějakou informaci
    - Zde nastává problém, že počítač nemusí být schopen v daný moment přijmout informaci
    - Pokud je komunikace iniciována periferií, je možné postupovat několika způsoby
      - Obvodové řešení – periferie může vyřešit danou situaci bez toho, aby o tom mikroprocesor věděl, toto řeší obvody nízké a střední integrace
      - Pro inicializaci lze použít nějakou Flagu, pokud počítač tento signál přečte a signál bude 1, může provést operaci. Skupiny příznakových bitů oznamuje periferie svůj stav (připraven, zaneprázdněn, porucha atd…) Skupina těchto bitů vytváří Status word (stavové slovo). Počítač tedy přečte slovo a rozhodne o další činnosti. Tomuto se říká programové řízení

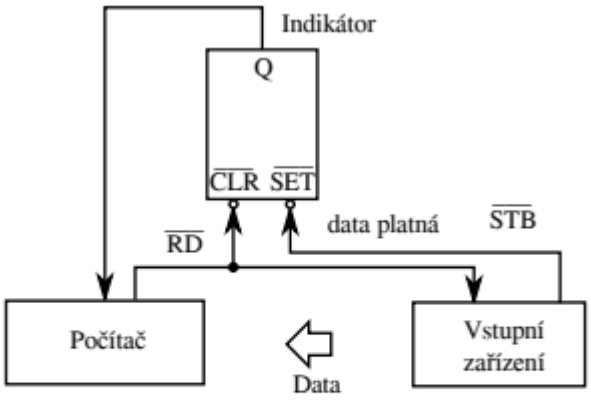
- Zařízení, které chce komunikovat, přeruší momentální akci pomocí speciálního signálu, poté se procesor vrátí na akci, kde byl přerušen, aby tento způsob mohl fungovat, musí mít procesor obsáhlý systém přerušení
- Pokud chce periferie komunikovat, přenese blok svých dat z periferie do paměti počítače nebo naopak, tato akce se může uskutečnit pomocí přímého přístupu do paměti (DMA)

Technika V/V bran

- Neboli I/O nebo input output
- Obvod, který zprostředkovává předávání dat mezi sběrnicí PC a periferiemi
- Lze použít bránu s pamětí nebo bez



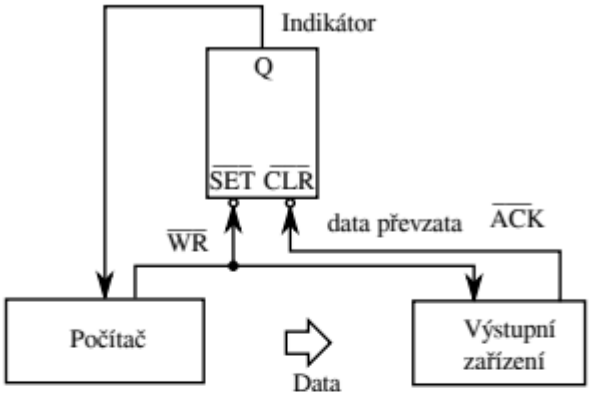
Obrázek 5: Technika nepodmíněného vstupu a výstupu dat



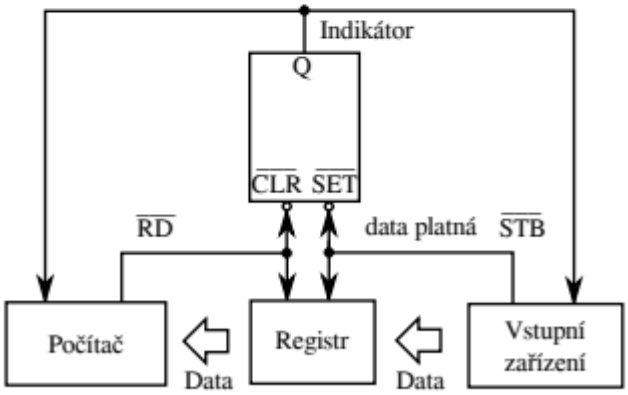
Obrázek 6: Technika podmíněného vstupu dat

-
- Oba typy těchto bran jsou zesilovače výkonu (budiče)
- Základem bran je záchytný registr (latch) s třístavovým výstupem
- Na obrázku č.5 je vidět {popisek}
- Při vstupu pošle počítač signál negace RD, čímž řekne počítači, aby předal data na vstupní bránu počítače
- Při výstupu počítač pošle data a negovaný signál WR a výstupní zařízení převezme data
- Tento způsob je velmi jednoduchý a vyžaduje stálou připravenost periferního zařízení ke komunikaci
- Technika podmíněného I/O je na obrázcích výše, přesněji na obrázku 6 je výstup
- Pokud jsou vstupní data platná, tak se díky signálu STB (strobe) nastaví hodnota Q na 1

- Stav Q je označován za indikátor (flag)
- Pokud je Q rovno 1, tak jsou předány data počítačem za pomoci impulsu negace RD a po dokončení je Q vynulován
- Následně se podíváme na výstup podle obrázku č.7

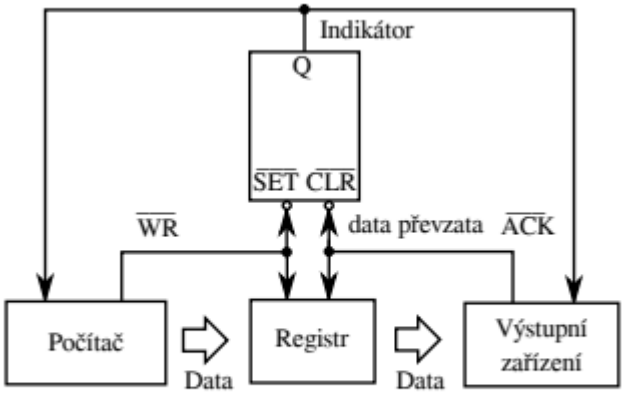


Obrázek 7: Technika podmíněného výstupu dat



Obrázek 8: Technika vstupu dat s vyrovnávací pamětí

-
- Počítač vyšle impuls negované WR pro přepis dat do výstupního zařízení a následně nastaví indikátor
- Výstupní zařízení jej po převzetí dat impulsem ACK indikátor nastaví na nulovou hodnotu
- Oba tyto případy jsou jednosměrné jednoduché korespondenční, kdy využíváme indikátor pro sdělování informací počítači (zahájení a ukončení přenosu) a kdy je vysílač povinen data udržovat
- Technika úplného obousměrného korespondenčního režimu využívá vyrovnávací paměti registru a klopný obvod který pracuje jako semafor (jeho stav testuje přijímač i vysílač dat)
- Lze se zde dostat do situace interlocku neboli vzájemného blokování
- Schéma vstupu dat s vyrovnávací pamětí z obr 8
- Semafor informuje procesor, zda jsou data ve vyrovnávací paměti připraveny
- Pro periférii semafor znamená, jestli může poslat procesor nová data, či zda je ještě nepřečetl



Obrázek 9: Technika výstupu dat s vyrovnávací pamětí

-
- Schéma pro výstup dat je na obr 9, význam semaforu je opačný než u řádku nad tímto

Programové řízení komunikace

- Využívají se instrukce pro I/O + testování logickým proměnných a skoků
- Pokud má počítač přímo vnější vstup příznakového (stavového) bitu a instrukce podmíněných skoků, umožnují větvení programu podle hodnoty příznakového bitu
- Obsluhu komunikace musíme organizovat vzhledem k hodnotám stavových bitů, takže dle své důležitosti
- Mějme nějakou posloupnost instrukcí, dnešní procesory nemají problém jí vykonávat tak rychle, že znakový bit můžeme zkoušet každých 10ms, pokud chceme častěji, musíme kontrolu zařadit vícekrát
- Existuje stavy, kdy procesor nastaví stavový bit na hodnotu a bude čekat na změnu/odezvu v nějakém časovém úseku, tomuto se říká čekací smyčky, lze to pochopit tak, že se bit testuje, dokud nebude změněn

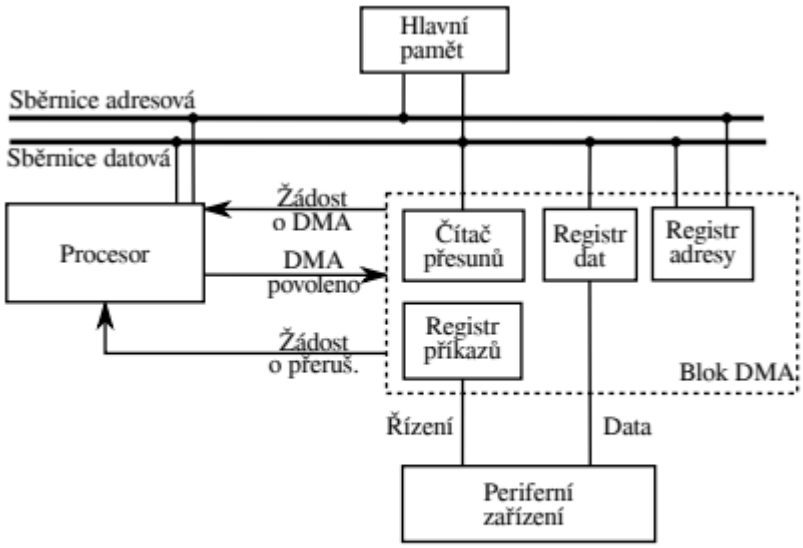
Řízení komunikace pomocí přerušení

- Přerušení usnadňuje řízení hlavně ve fázi zjišťování žádosti od obsluhu
- Jakmile počítač dostane přerušovací signál, vykoná instrukci, kterou má zadanou a poté se vrátí do programu kde skončil a pokračuje ve své práci
- Problém nastává při obsluze více zařízení -> je potřeba rozsáhlý přerušovací systém
- Informace o tom, které zařízení požádalo o přerušení jsou uloženy v tzv. vektoru přerušení neboli adresa toho obslouženého programu
- Princip žádostí je více vysvětlen později
- Možnosti identifikace zařízení žádající o obsluhu
    - Požadavky jednotlivých zařízení jsou logicky sečteny, procesor přečte stavové slovo, přečte z něj identifikační znak zařízení, které žádá o obsluhu, pokud je žádostí více, musíme toto přijetí požadavků programově oddělit
    - Opět jsou požadavky sečteny, signál potvrzení žádosti od mikroprocesoru putuje do všech periferních zařízení, priorita je tímto dána zařazením jednotlivých zařízení vůči tomuto signálu, pokud dostane procesor žádost k přerušení, zeptá se prvně periferie s největší prioritou, zda ono žádalo o přerušení, pokud ne putuje na další, takto dokud to pokračuje dokud se nedostane k hledanému zařízení, pak musíme zamezit

aby tento signál šel dál, tudíž procesoru vrátíme vektor přerušení (adresa programu), tento způsob již vyžaduje přídavné řešení technickými prostředky
    - Další možností je použití řadiče přerušení, tento systém spočívá v tom, že zařízení vyšle identifikační znak daného zařízení až po přijetí žádosti o přerušení, obsahuje dynamické změny priority jednotlivých zařízení
- Žádosti o přerušení jsou maskovány registrem masky
- Poté jsou uloženy do registru žádosti o přerušení
- Poté jsou předány do předány do bloku priorit
- Jsou rozřazeny podle důležitosti, poté se vybere nejvrchnější příkaz, zjistí se jeho adresa, ze které program požádal i pozastavení

Přímý přístup do paměti (DMA – direct memory access)

- Některé události nejdou bez přerušení
- Proces přesunutí dat/informací z periferie -> počítač nebo obráceně lze realizovat výhodněji pomocí DMA
- Princip spočívá v tom, že data se přenáší bez vědomí procesoru, ten se tudíž nemusí zastavovat
- Jediná podmínka je ta, že procesor uvolní sběrnice pro procesor -> přepne všechny budiče sběrnic na vysokoimpedační stav
- Sběrnice ovšem nemůže zůstat bez řízení, proto existuje blok, který generuje adresy a určuje okamžiky přesunu dát po sběrnici, tento blok se označuje jako DMA nebo též kanál DMA
- DMA příklad

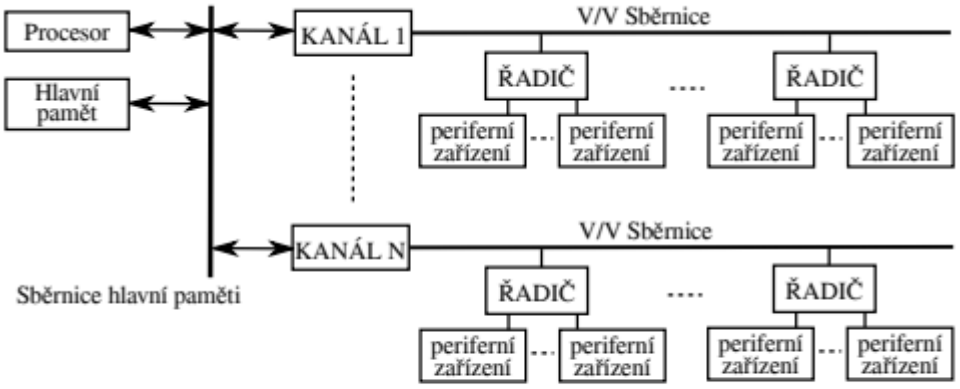


- Obrázek 10: Struktura bloku pro přímý přístup do paměti (DMA)
- Na obrázku jsou 3 registry určené pro styk se sběrnicí:
    - Registr dat – obsahuje slovo, jež má být přesunuto z periferního zařízení do paměti, nebo naopak
    - Registr adresy – Uchovává adresu hlavní paměti neboli adresa, na kterou bude toto slovo zapsáno, nebo ze které bude přečteno
    - Čítač přesunů – Počet slov, které mají být přesunuty v rámci jednoho spojení paměti a periferie

- Blok DMA pracuje ve dvou režimech
    - Přesouvání dat jednotlivě
    - Přesouvání dat v blocích
- Přístup do paměti v několika krocích:
    - Naprogramování procesorem bloku DMA
    - DMA spustí periferii a čeká až bude připraveno přijmout nebo vyslat data, poté periferie oznámí, že je připraveno žádat DMA o přímý přístup do paměti
    - Procesor nejdřív dokončí strojový cyklus a poté reaguje na žádost o DMA, poté se přístup uskuteční během normální činnosti procesoru tak, že DMA vysílá data synchronně v určitém čase, kdy procesor paměť nepoužívá
    - K přímému přístupu může dojít v okamžiku kdy procesor vyšle signál ACK a uvolní sběrnice, vybraná jednotka pak pošle na datovou sběrnici obsah svého registru a čeká na provedení cyklu paměti, poté zvětší obsah registrů adresy o jedničku a zmenší obsah čítače přesunů, dokud není obsah čítače přesunů nula, nadále se testuje, zda periferie předala nové slovo k přesune a opakuje se cyklus, pokud nebylo nové slovo přesunuto, ukončí se DMA a procesor převezme řízení
    - Procesor provádí svou činnost, dokud DMA nedostane a nevyšle novou žádost o DMA, toto signalizuje připravenost dat pro přesun v registru dat
    - Pokud je obsah čítače přesunů nulový, blok DMA končí celý přesun a uvolní sběrnice, také může zaslat žádost o přerušení, která spočívá v naprogramování jeho registrů
- Řadič DMA adresuje hlavní paměť a synchronizuje činnost mezi sběrnicemi a periferním zařízením, všechny data jdou mimo něj
- Jsou subsystémy, které jsou o něco chytřejší¨, než DMA, jsou schopné data kontrolovat, upravovat, opakovat atd… (Jedná se o kanál)
- Kanál je speciální procesor určený pro spojení hlavního procesoru s periferními zařízeními, řídí se kanálovým programem
- Kanály mohou být multiplexní a selektorové

Kanálová architektura

- Kanál je programově řízený procesor, který je schopen se samostatně řídit V/V operacemi
- Je umístěn mezi procesorem a řadičem periferního zařízení



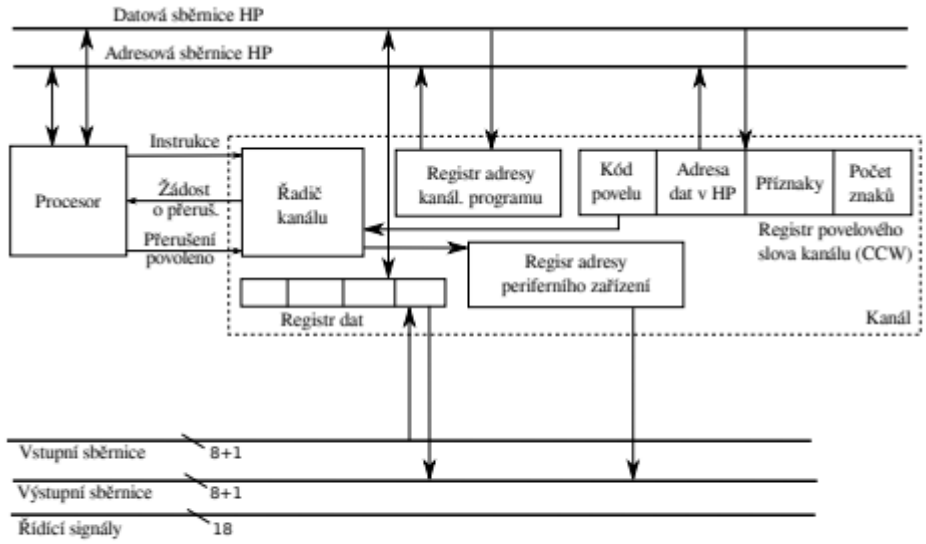
Obrázek 11: Kanálová architektura počítače

- Obvykle se používá v architektuře počítače

- Jeden systém může obsahovat několik kanálů, které jsou propojené na sběrnici hlavní paměti a procesoru
- Hlavní paměť řídí „organizátor“ nebo „přidělovač“ hlavní paměti, jemuž se musí kanál podřídit jako každý jiný účastník
- Každý jeden kanál řídí periferní sběrnici, k níž mohou být připojena jednotlivá periferní zařízení
- U některých systémů lze propojit jeden kanál s více počítači -> multipočítačové systémy
- Dva počítače se dají propojit pomocí adaptéru kanál – kanál, který se připojuje na V/V sběrnici, díky tomuto jsme schopní vytvořit multipočítačové systémy
- Činnost kanálu během V/V operace je řízena kanálovým programem, který se provádí přímo v kanálu
- Kanálový program je uložený v hlavní paměti a čte se po povelových slovech (CCW) -> přenáší se do CCW registru v kanálu, kde je řízena jeho činnost
- Používají se pro přenos dat mezi hlavní pamětí a periferiemi

Struktura a funkce kanálů

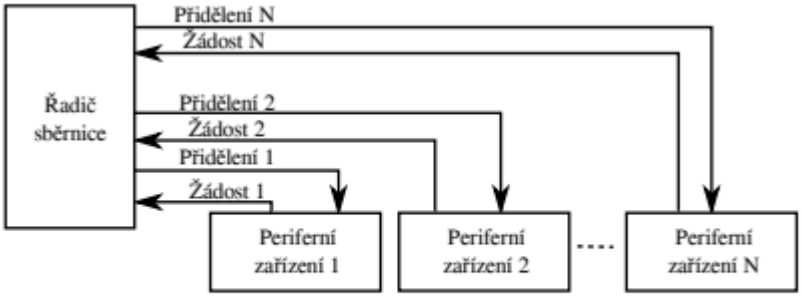
- Kanál má taky možnost přímého přístupu do paměti, který se v mnohém neliší od DMA
- Jedna z věcí, co má navíc je vlastní řadič a náležící kanál který mu umožnuje provádět kanálový program obrázek č.12
- Struktura kanálu záleží, zda se jedná o selektorový nebo multiplexní kanál
- Hlavní rozdíl mezi selektorovým a multiplexním kanálem
    - Multiplexní může obsluhovat více periferních zařízení najednou
    - Selektorový pouze jedno, používá se pro rychlý přenos, třeba disky



Obrázek 12: Struktura kanálu

- Selektorový přesun
    - Kanál naváže spojení s určitým periferním zařízením
    - Příkaz ke spuštění V/V operace
    - Přesun bloku dat z hlavní paměti do periferie nebo naopak
    - Ukončení spojení
    - Poté může navázat spojení s jinou periferií
    - Znovu uloží blok dat atd…

- Selektorové kanály se už nepoužívají, nahradili je multiplexní -> stejná rychlost + další výhody
- Aby byl schopen kanál vykonávat více věcí najednou, potřebuje podkanály
- Tyto podkanály jsou přiřazeny jednotlivým periferiím
- Až 256 podkanálů
- Multiplexní kanál se označuje za slabikový nebo blokový, podle toho, jestli se jedná o slabiky nebo o celé bloky
- Slabikový multiplexní kanál se používá pro obsluhu pomalých zařízení jako je klávesnice, tiskárna, komunikační linky
- Přesun několika slabik po sobě jdoucích se označuje jako dávkový (burst mode)
- Jeho výhoda je vysoká rychlost přesunu, až několik stovek tisíc slabik za sekundu



Obrázek 13: Samostatné žádosti

Vyhodnocování žádostí a jejich priorit

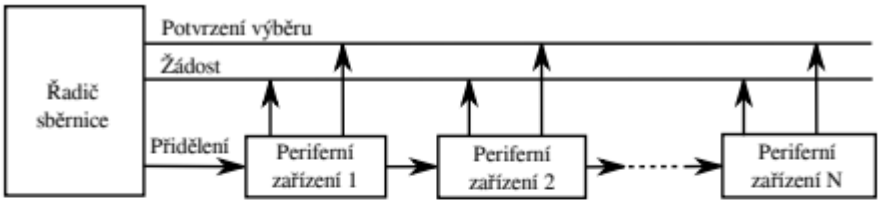
- Často se používají tzv žádosti neboli připravenost zařízení komunikovat, či dělat nějakou činnost
- Tyto žádosti chodí ze všech částí PC a neorientovaně, proto potřebujeme zjistit které potřebujeme řešit dřív než ty jiné
- Nejpoužívanější systémy rozhodování:
    - Samostatné žádosti
    - Zřetězení
    - Cyklické výzvy

Samostatné žádosti

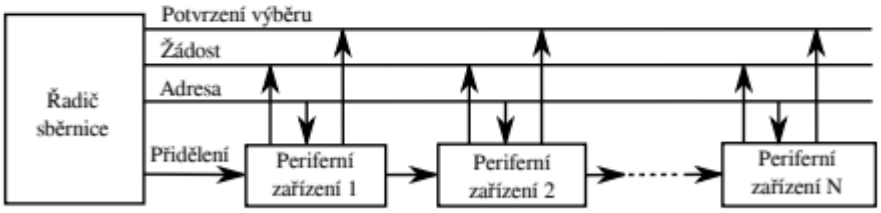
- Posílají se přímo po radikálních vodičích do procesoru
- Ke každému vodiči „žádost“ je přiřazen vodič „přidělení“
- Probíhají radikálně k žádající jednotce viz blokové schéma níže
- Díky tomuto jednoznačnému přiřazení procesor ví, která jednotka žádala o sběrnici
- Vyhodnocení priority se může provést jednoduchým prioritním dekodérem
- Priority jsou přiděleny pevně, což umožnuje minimální zpoždění
- Další možnost je vyhodnotit přijaté žádosti programem, který cyklicky prohledává registr a v kterém jsou tyto změny zapsány
- Výhodou programu je možnost lehké změny

Zřetězení

- Všechny jednotky vysílají najednou, takže žádost je anonymní a procesor jí sám nemůže vyřešit



Obrázek 14: Vyhodnocení priority zřetězením



Obrázek 15: Vyhodnocení priority cyklickými výzvami

-
- Tyto požadavky jsou logicky sečteny signál potvrzení žádosti je sériově veden po všech periferních zařízeních
- Pořadí spojení periferií odpovídá jejich pořadí priorit
- Pokud jednotka žádala o přidělovací signál, tak jej zablokuje a znemožní dalšímu zařízení se k němu dostat, tak se určí přiřazení této jednotky
- Toto potvrzení se udělá tak, že vygeneruje svoji adresu a pošle jí do procesoru

Cyklické výzvy

- Jinak nazýváno (polling)
- Své žádosti opět vysílají po jednom vodiči sběrnicového typu
- Znovu bude žádost anonymní
- Namísto jednoho vodiče pro přidělení je však použita skupina vodičů „adresa jednotky“, která je schopný přenášet adresu kterékoliv z připojených jednotek
- Pokud je procesor připraven žádosti vyhovět, tak začne po adresových vodičích vysílat postupně adresy všech jednotek v předem stanoveném pořadí
- Jakmile se k jednotce, co žádala dostane její adresa, bude na ni reagovat signálem „přiděleno“
- Poté vyšle do procesoru signál „potvrzení výběru“ a tak se stane řídící jednotkou

Historie sběrnic PC

Typ	rok vzniku	Datová šířka [bitů]	takt [MHz]	Přenosová rychlost [MB/s]
XT bus	1987	8	4,77	až 2,38
ISA	1984	16	7,15 či 8,33	až 16,6
MCA	1987	32	10,22	40
EISA	1988	32	8,33	až 33
VESA Local BUS	1992	32	25–66	až 264
Opti Local BUS	1992	32	25–66	až 264
PCI	1993	32/64	20–66	132 až 528
AGP	1992	32	66/133	264–2112
PCI-X	2002	64	66/133	533–1066
PCI-Express	2004	1–128	250	až 32000

Tabulka 1: Přehled vývoje sběrnic PC

RISC (a CISC)

- Ustálené dělení počítačů do dvou základních kategorií
  - CISC
    - Počítač se složitým souborem instrukcí
  - RISC
    - Počítač s redukovaným souborem instrukcí
- V dnešní době jsou to všechny směsky, není procesor, který by měl čisté rysy jedné z těch architektur
- CISC – prominentní v 70. letech
  - Už bylo všechno moc složité, proto se začalo redukovat
  - Díky tomu vzniklo RISC
- Studie v té době ukázali, že instrukce jsou nerovnoměrně rozloženy a využívá se velmi úzké spektrum instrukcí

		%
LOAD	čtení z paměti	26.6
STORE	zápis do paměti	15.6
Jcond	podmíněný skok	10.0
LOADA	načtení adresy	7.0
SUB	odčítání	5.8
CALL	volání podprogramu	5.3
SLL	bitový posun vlevo	3.6
IC	vložení znaku	3.2

Tabulka 1: Statická četnost instrukcí

- První tabulka je statická četnost – instrukce v programu
- Druhá tabulka je dynamická četnost – frekvence vykonávání
- V obou tabulkách je vidět malé spektrum diverzifikace 3 instrukce = 50% nebo 8 instrukcí = 75%

		%
LOAD	čtení z paměti	27.3
Jcond	podmíněný skok	13.7
STORE	zápis do paměti	9.8
CMP	porovnání	6.2
LOADA	načtení adresy	6.1
SUB	odčítání	4.5
IC	vložení znaku	4.1
ADD	sčítání	3.7

Tabulka 2: Dynamická četnost instrukcí

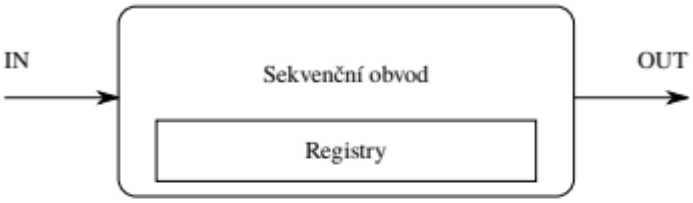
- Chtěli přidat další instrukce, ale kvůli tomu by se musel rozšířit řadič procesoru, ale i tak by se používalo hrozně málo instrukcí, takže to bylo worthless
- Řadič byl tím pádem také málo používaný
- Takže se nakonec začalo redukovat a vznikl RISC
  - Hlavní protagonisti:
    - Kalifornská Univerzita
    - Berkeley a Stanfordova Univerzita
    - IBM

- Poté na uni. Berkley vznikl v roce 1982 procesor RISCI, od jehož názvu je odvozený celý směr vývoje počítačů
- Další rok Stanford udělal MIPS
    - Procesor bez vzájemně se blokujících sekcí proudového zpracování
- Z těchto dvou projektů poté vzniklo několik prvních generací průmyslově vyráběných RICS procesorů
- Na trhu byly v polovině 18. stol
    - Byly rozměrné
    - MSI – middle sized integration
- Vývoj RISC a CISC je spjatý, dnes nemá ani jeden procesor čistě CISC nebo RICS, jde spíše o marketing

Vlastnosti RICS architektury

- Charakteristická vlastnost – malý instrukční soubor
- Tato vývojová větev měla vývojáři stanovena zásadní kritéria např:
    - Metodiku návrhu počítače
    - Metodika návrhu procesoru
    - Procesoru nechat pouze akce které jsou nutné a rozdělit zbylé funkce do:
        - Architektury počítače
        - Programového vybavení
        - Kompilátoru
- Výsledné vlastnosti RISC:
    - V každém strojovém cyklu by se měla udělat jedna instrukce
        - To neznamená že každá instrukce musí trvat právě jeden cyklus
        - Imo to může být tak, že těch instrukcí je několik a prostě se jedna finishne na každém konci
    - Mikroprogramový řadič může být nahrazen rychlejším obvodovým řadičem
    - Používání zřetězených instrukcí
    - Celkový počet instrukcí a způsobů adresování je malý
    - Data jsou z hlavní paměti ukládána a čtena pomocí dvou lehkých instrukcí LOAD a STORE
    - Instrukce mají pevnou délku a jednotný formát, ten vymezuje význam jednotlivých bitů
    - Používá se vyšší počet registrů
    - Složitost se z technického vybavení přesouvá částečně co kompilátoru, který z části optimalizuje
- Tyto „pravidla“ tvoří promyšlený a provázaný celek
- Například:
    - Navýšení počtu registrů, nám umožnuje pracovat s pamětí pomoci pouze dvou instrukcí LOAD a STORE
    - Pokud ostatní instrukce nemohou používat paměťové operandy, musí procesor schraňovat více dat
    - Další souvislost je zpracovávání zřetězeným a formátem instrukcí
        - Jednotná délka instrukcí umožnuje rychlejší výběr instrukcí z paměti, tím se rychleji naplní fronta instrukcí
        - Jednotný formát pak umožnuje rychlejší dekodování instrukcí
- Výsledem výroby počítačů je výhodný pro uživatele i výrobce

    - Zkracuje se doba vývoje
    - Procesory pracují správně
- Přesto, že má RISC své nedostatky, tak se většina výrobců odklonila od výroby CISC a začala vyrábět RISC procesory
- Nevýhody RISC
    - Nutný nárůst délky programů, kvůli omezenému počtu instrukcí a jejich délce
    - Zpomalení, kterého se báli, tak se v praxi nepotvrdilo
    - Je potřeba si uvědomit kolik instrukcí muselo být rozepsáno a zřetězeno -> urychlení práce procesoru



Obrázek 1: Procesor (CISC) jako sekvenční obvod

Krok		Význam
1.	**VI**	Výběr Instrukce
2.	**DE**	Dekódování
3.	**VA**	Výpočet Adresy
4.	**VO**	Výběr Operandu
5.	**PI**	Provedení Instrukce
6.	**UV**	Uložení Výsledku

- Tabulka 3: Příklad možných kroků zpracování instrukce

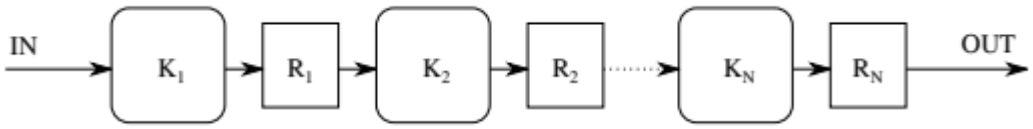
Proudové zpracování instrukcí

- Procesor si lze představit jako sekvenční obvod z obrázku č.1
    - Vstup jsou strojové instrukce
    - Na výstupu se opět data ukládají do paměti
    - Každá instrukce projde celým obvodem, dokud se celá instrukce nedokončí, tak není možné začít novou
- Provedení instrukce musí projít vždy stejnými fázemi
- Zjednodušeně:
    - Instrukce se musí vybrat z paměti
    - Dekodovat
    - Vypočítat adresu operandu
    - Připravit data
    - Provést instrukci
    - Uložit výsledky
- Znázorněné je to v tabulce č.3
- Pokud bude provedení jednoho elementárního kroku trvat jeden cyklus, lze si to představit takto, cykly ovšem nemusí být stejně dlouhé

	T_1	T_2	T_3	T_4	T_5	T_6	T_7	T_8	T_9	T_{10}	T_{11}	T_{12}	T_{13}
VI	I_1						I_2						...
DE		I_1						I_2					
VA			I_1						I_2				
VO				I_1						I_2			
PI					I_1						I_2		
UV						I_1						I_2	

Tabulka 4: Postup provádění instrukcí procesorem CISC

- Kdyby se povedlo osamostatnit jednotlivé části tohoto sekvenčního obvodu z obrázku č. 1, tak aby každý byl samostatný obvod, tak aby každému obvodu připadala jedna fáze zpracování instrukce
- Potom by se jednalo o posloupnost zřetězených instrukcí (princip výrobní linky)
- Obvod by pak šel realizovat dle obrázku č.2



Obrázek 2: Zřetězené zpracování (v procesoru RISC)

- Mezi jednotlivé fáze však musí být zařazen registr pro mezivýsledek, tento registr slouží jako předávací místo mezi po sobě jdoucími prvky
    - Z tohoto plyne zpomalení, které jde ale zanedbat, protože je to worth za ten přínos
- Aby se zřetězení vyplatilo, musí být všechny části řetězu stejně časově náročné (zároveň je to jeden z předpokladů)
    - Jinak by nejslabší článek brzdil zbytek procesu (jako klasický řetěz)
- V tabulkách taky můžeme vidět vypočítanou účinnost, v první tabulce se za 12 cyklů provedly 2 instrukce, v druhé tabulce to je 7 instrukcí za 12 cyklů
- V teorii to tedy znamená, že v nekonečném čase nám N úrovní zřetězení zrychlí vykonání instrukcí Nx
- V realitě je však nějaká limitace pro tento přínos
    - I/O zpomalení
    - Zpomalení při předávání mezivýsledku
    - Využívání pomocných obvodů
        - Sběrnice
        - Registry atd…
- Nejkritičtější problém je však plnění zřetězení -> vzniklá fronta rozpracovaných instrukcí
    - Podmíněné skoky znehodnocují frontu rozpracovaných instrukcí
        - Rozpracované věci se zahodí a pak se to začne dělat znovu
        - Čím víc hluboké zřetězení, tím větší zdržení
- Každá šestá instrukce je podmíněný skok, takže zřetězení není tak insane
- Lze najít ideální hloubku zřetězení

Problémy zřetězeného zpracování

- Zřetězení sice nabízí zvýšení výkonu, ale také i spoustu nástrah a problémů
- Například hazardové datové i strukturální problémy s plněním fronty instrukcí

Datové a strukturální hazardy

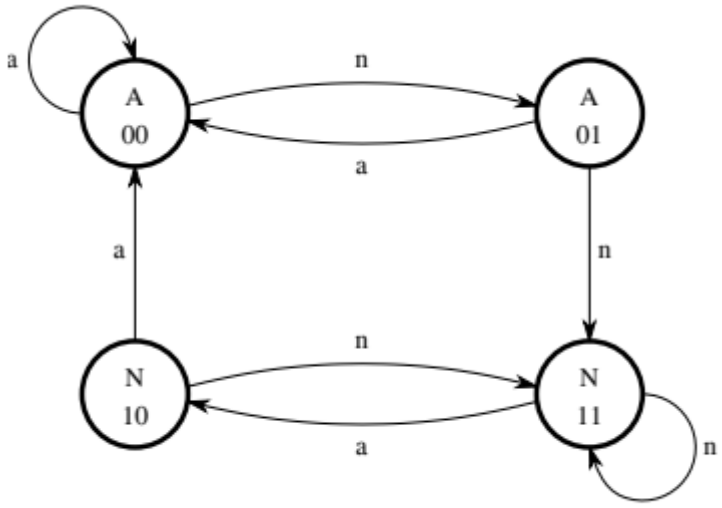
- Vznikají například, pokud určitá instrukce, potřebuje data z předchozí instrukce, ale ty ještě nejsou ready – lze si představit pomocí tabulky č.5 – příklad na to ve skriptech
- Pro vyřešení problému musí být zřetězená jednotka uzpůsobená svou konstrukcí, tento problém se také řeší hned v překladači, aby se tomuto problému zabránilo
- Lehká charakterizace: Problém omezených prostředků procesoru a počítače jako celku
- Například při výběru sběrnice, instrukce, operandu je za potřebí komunikovat se sběrnicí, ale ta je k dispozici pouze jedna, takže se musí čekat atd…
- Tyto problémy jsou pouze špička ledovce této složité problematiky

Problémy plnění fronty instrukcí

- Pro správnou funkci zřetězení je důležitá reakce na skokové instrukce
    - Nejjednodušší je řešení nepodmíněných skoků a volání programů s pevnou adresou
    - Horší je, pokud se adresa nepodmíněného skoku musí vypočítávat
    - Pokud se tato adresa nepočítá dost rychle, tak hrozí poškození zásobníku
    - Je nutné omezit četnost těchto případů optimalizací pořadí instrukcí atd…
        - Např optimalizující překladač vyšších programovacích jazyků
- Podmíněný skok:
    - Většinou známe adresu skoku, ale nevíme, jestli se skok provede či ne
    - Nemá smysl čekat, zda se provede nebo ne, prostě se pokračuje dál ve zpracovávání sekvenčním způsobem a pokud se skok neprovede, tak se fronta provede
    - Pokud se skok provede, tak se dočasná fronta zahodí a naplní se nová
- Pro vysoko výkonné počítače se většinou implementují dvě paralelní fronty, druhá většinou kratší než ta první a bývá omezená pouze na první 4 kroky zpracování
    - Prováděcí jednotka pak vybírá z front věci, co se mají dít
    - Je jasné že vybírání z front zabere nějaký čas, ale je to ta efektivnější varianta
- V RISC se používá kromě těchto dvou krajních metod i předpovídání skoků atd…

Metody pro zlepšení plnění fronty instrukcí

- Největší problém jsou podmíněné skoky – cca 15-25%
- Je nutno tedy hledat levné a účinné řešení, jak toto chování skoků předpovídat



- Obrázek 3: Dvoubitová predikce jako stavový automat

Bit predikce skoku

- Ve formátu instrukce je jeden bit vyhrazený predikci, zda skok provede nebo ne
- Jednotka výběru instrukce, pak vybírá instrukce z predikované adresy
- Predikce může být statická nebo dynamická
    - Statická
        - Bity se vkládají už při kompilaci anebo přímo programátorem do kódu programu
    - Dynamická
        - Tato predikce si při každém cyklu zaznamená, zda se provedl nebo ne
        - Je lepší, protože se přizpůsobí aktuálním podmínkám
        - Tohle může být ale problém, protože program potřebuje přepisovat tento bit, což může být často nemožné (kvůli read only)
- Jednobitová predikce skoku
    - Jednoduchá
    - Problém v podmínce cyklu
    - Dojde vždy k selhání na začátku cyklu a u dynamické predikce i na konci, to by nebyl takový problém, ale při více cyklech se toto číslo exponenciálně zvětšuje
- Predikce dvoubitová
    - Tady se snižuje počet selhání pouze na jedno na konci cyklu
    - Toto chování je ukázané na obr č.3
- Jedná se o čtyřstavový automat, kde stav A predikuje provedení skoku, zatím co stav N říká, že skok se provádět nebude
- Přechody „a“ a „n“ ukazují, zda se krok naposledy provedl nebo ne
- Pokud se tedy například skok dvakrát po sobě provede, tak se predikce změní z A na N
- V ustáleném stavu se tedy predikce nastaví na A (00) nebo N (11)

	T_1	T_2	T_3	T_4	T_5	T_6	T_7	T_8	T_9	T_{10}	T_{11}
VID	I_1	I_2	I_3	I_4	I_5	X	X	I_6	I_7		
VVD		I_1	I_2	I_3	I_4	I_5	X	X	I_6	I_7	
PUV			I_1	I_2	I_3	I_4	I_5	X	X	I_6	I_7

Tabulka 6: Výpadek fronty po skokové instrukci

	T_1	T_2	T_3	T_4	T_5	T_6	T_7	T_8	T_9
VID	I_1	I_2	I_5	I_3	I_4	I_6	I_7		
VVD		I_1	I_2	I_5	I_3	I_4	I_6	I_7	
PUV			I_1	I_2	I_5	I_3	I_4	I_6	I_7

- Tabulka 7: Využití zpožděného skoku - změna pořadí provedení instrukcí

Zpoždění skokové instrukce

- Představme si nejprve zjednodušenou zřetězenou jednotku pouze se třemi úrovněmi zřetězení. V první fázi se provede výběr instrukce a dekódování (VID), následuje výpočet adres a výběr dat (VVD) a poslední třetí krok je provedení instrukce a uložení výsledku (PUV).
- Pokud se podíváme a bude instrukce I5 skok, tak to, co je označené X bude zahozeno a ignorováno

- Kdybychom činnost procesoru po skoku neskončili -> rozpracované instrukce by se provedly a pokračovalo by se v I6
- Toto by zjednodušilo logiku řízení procesoru, ale narušilo by samotnou strukturu, protože by se potom provedlo ještě několik rozpracovaných instrukcí a mohlo by to narušit logiku programu
- Místo toho vyplníme X instrukcemi NOP, ty jsou prázdné, vykonají se, ale nezrychlí program
- Pokud by se však našlo v této mezeře něco, co nepatří k podmíněnému skoku, tak bychom je mohli zařadit na toto prázdné místo X
- Toto upravené schéma bude v tabulce č.7
- V optimalizačním kompilátoru je poměrně snadné realizovat přeskupení instrukcí a využít tak zpoždění skoku k vyšší efektivitě práce zřetězené jednotky
- Pro názornost si to můžeme ukázat na tomhle obrázku jazyka C a přepisem do assembly:

```
/* Příkaz v jazyce C */
if ( i++ == j++ ) { /* blok příkazů ... */ }
k++;

;Přepis do assembleru
CMP i, j
JNE pokracuj
INC i
INC j
; blok příkazů ...
pokracuj:
  INC k
```

- Je vidět že inkrementace i a j se musí provést ještě před vykonáním bloku příkazů
- Pokud víme že skoková instrukce bude zpomalená, můžeme to nechat až na konec bloku kódu
- Pro analogii s tabulkou č.7 si představme že I5 je podmíněný skok JNE pokracuj a inkrementace proměnných budou instrukce I3 a I4

Použití paměti skoků

- Velmi rozšířeným řešením se stala tabulka provedených podmíněných skoků -> součást procesoru
- Velikost je pevně daná
- Ukládají se zde adresy posledních pod. skoků
- K těmto údajům se připojuje jednobitová nebo dvojbitová predikce
- Tato metoda nevyžaduje úpravu formátů
- Není spojena s činností překladače a je možno ji použít v nových generacích procesorů, kdy je vyžadovaná zpětná kompatibilita strojového kódu

CISC

- Nelze popsat všechny výrobce a modely čipů CISC, takže se budeme držet řady Intel x86

Historie procesorů Intel x86

- Každému procesoru, který se za posledních 35 let objevil bude věnována krátká kapitola
- Budou zde popsané vlastnosti, typické rysy, nové technologie
- Bude zde vidět změna ze CISC architektury na RISC, a přitom jsou zpětně kompatibilní

Intel 8080

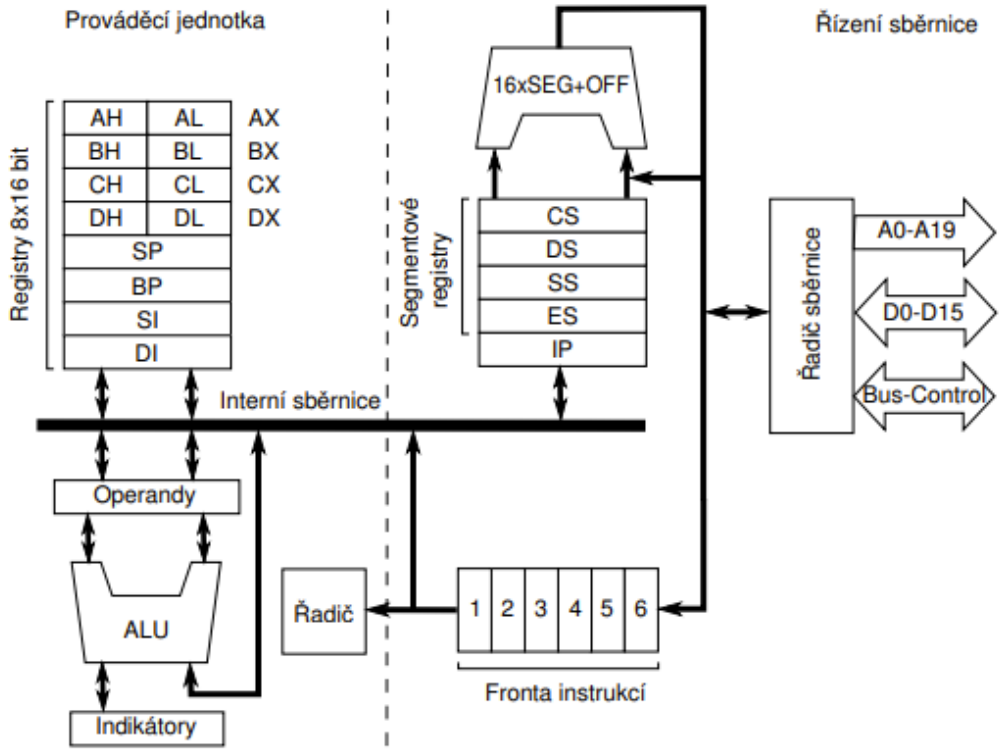
Uvedení	1974
Technologie	NMOS, $6\mu m$
Tranzistory	6000
Frekvence	2 MHz
Datová sb.	8 bit
Adresní sb.	16 bit

- Není sice prvním v řadě x86
- Základ pro celou řadu prvních jednodeskových počítačů
- Svou instrukční sadou inspiroval další výrobce při vývoji 8bitových procesorů
- Na úrovni assembleru kompatibilní s 16bitovou verzí 8086
- Vyráběl do se 80. let
- Nahrazen v roce 1977 -> intel 8085

Intel 8086

Výroba	$1976 \div 1990$
Technologie	HMOS, $3.2\mu m$
Tranzistory	29000
Frekvence	4.77 MHz $\div$ 10 MHz
Datová sb.	16 bit
Adresní sb.	20 bit

- První 16bitový procesor
- Až 1MB paměti -> segmentace paměti na 64kB bloky



Obrázek 1: Architektura procesoru i8086

-
- Jednoduchá vnitřní architektura
- Oddělení výkonné a adresovací části procesoru
- 8x 16bitové registry -> základ pro všechny následující procesory generace x86

Intel 8088

- Následovník 8086
- Architektura úplně stejná, jenom se změnil výstup datové sběrnice -> zúžení na 8bitů
- Zpomalení práce procesoru -> multiplexování do 16bitů
- Bylo to hlavně kvůli ekonomice -> levnější konstrukce
- Možnost využít celou řadu periferních obvodů, navržených pro 8bitové procesory
- Nový standard PC-XT od IBM založený na tomto procesoru

Intel 80186/188

Výroba	1982 ÷ 2007
Technologie	HMOS, 1.3μm
Tranzistory	55000
Frekvence	6 MHz ÷ 25 MHz
Datová sb.	16/8 bit
Adresní sb.	20 bit

-
- Roku 1982 -> vylepšená verze 8086/88
- Vylepšena: Architektura, přidány/upraveny/zrychleny některé instrukce
- Procesor měl vestavěná zařízení -> obsahoval celou řadu pomocných obvodů přímo na čipu
- Například DMA, hodiny, časovače a porty
- Tímto se stal nekompatibilní s PC-XY ikdyž podporoval zpětnou kompatibilitu
- Byl vyráběn 25 let a během těchto let byl modernizován

Intel 80286

Výroba	1982 ÷ ˜1993
Technologie	CMOS, 1.5μm
Tranzistory	134000
Frekvence	6 MHz ÷ 25 MHz
Datová sb.	16 bit
Adresní sb.	24 bit

- Druhá generace procesoru s označením 80286 -> navýšení výkonu a odemknutí taktování
- Představil řadu novinek a problémů
- Podporuje „Protected mode“ ve kterém může adresovat až 16MB paměti
- V tomto modu mohou mít aplikace a programy 4 stupně povolení
- Real mode – režim práce s předchozími kompatibilními procesory
    - R-M nemohou v P-M pracovat
- Nebylo možné se dostat do úplné kompatibility, takže vývojáři byly v patové situaci, nebylo možné vytvořit OS, který by používal všechny možnosti procesoru a zároveň využíval již existující software
- 100 % v R-M, navíc nebylo možné změnit na P-M a zpátky bez restartu
- Procesor byl vybaven MMU (Memory management unit) -> umožnění stránkování -> v té době pouze u sálových počítačů
- Kromě toho uměl procesor vytvořit virtuální pamět
- Byl schopný pracovat se 30bitovou adresou, mohl tedy virtuálně adresovat až 1GB paměti
- Schéma na obr č.5

Intel 80386DX

Výroba	1985 ÷ 2007
Technologie	CHMOS, 1.5 ÷ 1 μm
Tranzistory	275000
Frekvence	16 MHz ÷ 40 MHz
Datová sb.	32 bit
Adresní sb.	32 bit

- První plně 32bitový procesor třetí generace 80386
- Zpětná kompatibilita zůstala, všech 8 registrů -> 32 bitů, stejně tak i datová a adresní sběrnice
- Celá řada nových instrukcí
- Nově Virtual mode -> možnost přepnutí do P-M a vykonávat R-M programy -> zpětná kompatibilita starých programů v nových OS
- Řadič vyrovnávací paměti -> Cache L1 -> nezbytné pro takt 33mhz a výše -> řádově jednotky až desítky kB
- Virtuální pamět -> možno až 64TB paměti -> systém zůstal zachován pro nástupníky x86 až do 64bitových technologií
- Ukázka na obr. 6
- Poprvé v historii někdo předehnal IBM ve výrobě prvního čipu, byla jím firma COMPAQ

Intel 80368SX

- Roku 1988
- Jednalo se spíše o malý krok zpět

- Ekonomické důvody, redukce na 16bitů stejně jako u 8088
- Používala se základna pro 80286

Intel 8087/287/387

- Pouze pro výpočty s celými čísly
- Pokud chtěli i plovoucí čísla (FPU) potřebovali matematický koprocesor
- Tento procesor byl pouze základ pro rozšíření, pokud bylo nutné
- Musela se výrobcem umístit patice do které se koprocesor vkládal

Intel 80486DX/i486DX

Výroba	$1989 \div 2007$
Technologie	CHMOS, $1 \div 0.6\ \mu m$
Tranzistory	$1.2 \cdot 10^6$
Frekvence	$16\ MHz \div 100\ MHz$
Datová sb.	32 bit
Adresní sb.	32 bit

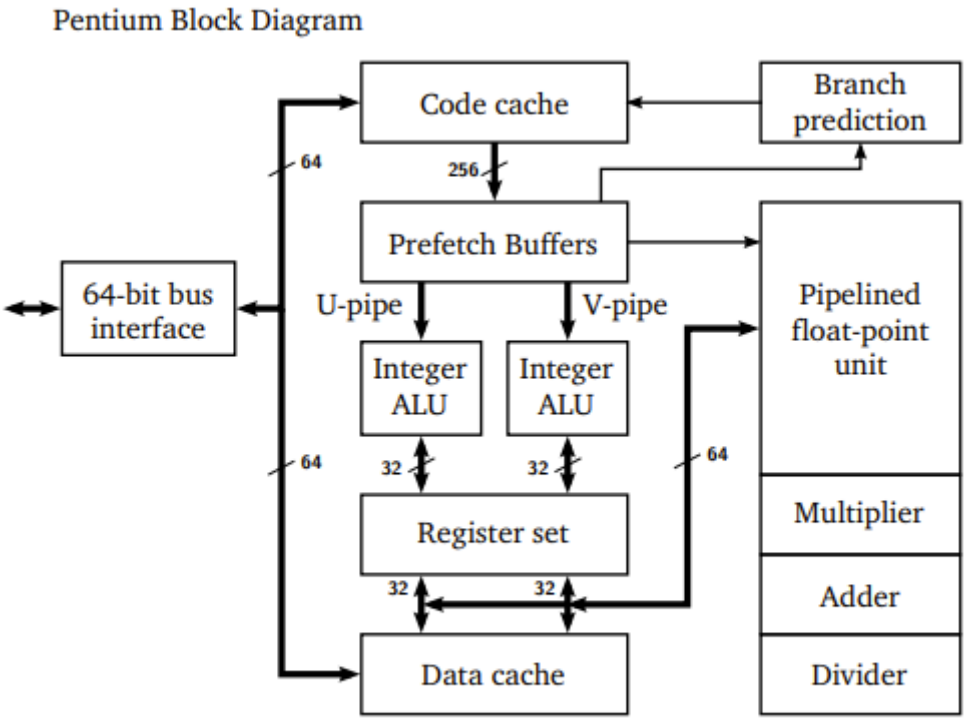
- Čtvrtá generace zůstává plně 32bit
- 5x počet tranzistorů
- Dvojnásobný výkon oproti 80386, při stejné frekvenci -> díky lepšímu ALU, delší frontě instrukcí a zlepšení podpůrnosti mezi jednotlivými jednotkami procesoru + impedance cache (L1) paměti přímo v procesoru
- Společná pamět pro data i kod o velikosti 8kB
- Vylepšení MMU -> rychlejší činnost v protected modu
- Obrázek č.7
- Integrace matematického procesoru -> nebyla nutnost další modul pro výpočet floating numbers
- 1991 představil intel verzi procesoru i486SX
    - Zlevněná verze původního procesoru
    - Tentokrát však nezmenšila sběrnici, ale odstranila FPU jednotku
- V roce 1992 představil intel verzi i486DX2
    - Vnitřní frekvence se násobí dvěma
    - První verze 50/25 MHZ, 66/33 MHZ
- Výhoda byla, že se dřívejší verze DX dali nahradit přímo DX2

Intel Pentium

Výroba	$1993 \div 2001$
Technologie	BiCMOS, $0.8 \div 0.25\ \mu m$
Tranzistory	$3.1 \cdot 10^6$
Frekvence	$60\ MHz \div 300\ MHz$
Datová sb.	64 bit
Adresní sb.	32 bit

- Pátá generace procesorů -> První procesor v řadě x86
- Typické uplatnění RISC
- Superscalární architektura
- Dvě paralelní ALU jednotky -> v ideálním případě může procesor zpracovávat dvě instrukce najednou

- Při složitějším ukonu, by pak mohli obě jednotky spolupracovat
- Obrázek č.2
- Cache L1 pamět rozdělena na dvě, pro data a pro kod
- Rozšíření sběrnic + jednotka pro predikci skoků
- Zůstává FPU jednotka -> velmi výkonná FPU jednotka



Obrázek 2: Zjednodušené blokové schéma procesoru Pentium

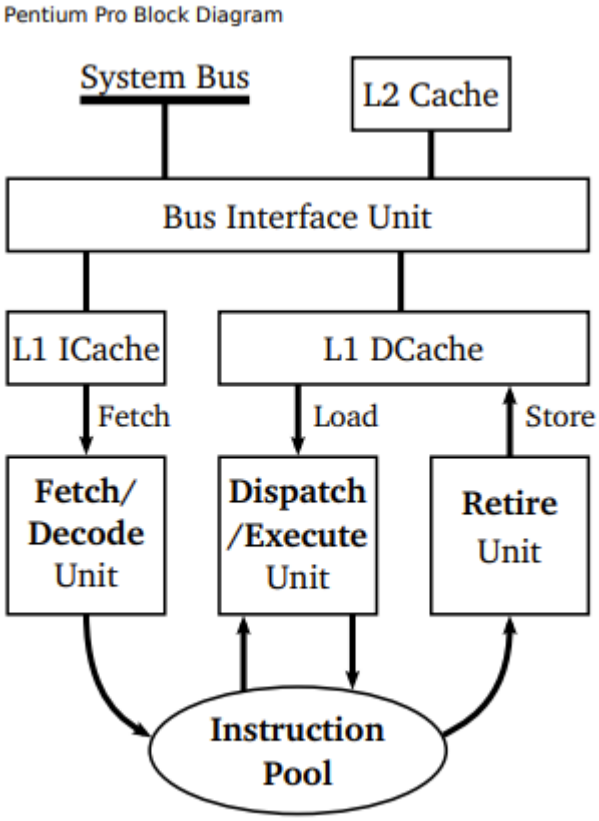
- Napájeny 5V používá 0,8 nanometrovou technologií, frekvence 60/66MHZ
- Nazývány ohřívače kávy
- Rok po představení se vydala nová verze s frekvencemi 75/90/100/120MHz napájené 3,3V
- Procesor procházel v dalších sedmi letech vývojem -> hlavní v roce 1997 -> rozšíření o jednotku MMX -> podpora multimediálních instrukcí¨

Intel Pentium Pro

Výroba	$1995 \div 1998$
Technologie	BiCMOS, $0.5 \div 0.35\ \mu m$
Tranzistory	jádro $5.5 \cdot 10^6 + 15.5 \cdot 10^6$ pro 256 kB cache L2
Frekvence	$150\ MHz \div 200\ MHz$
Datová sb.	64 bit
Adresní sb.	36 bit

- Delší vývoj než klasického Pentia, protože cíle byly vyšší

- Představen až 1995
- Primárně pro servery -> cena a výpočetní výkon + 50% oproti klasickému pentiu



- Obrázek 3: Zjednodušené blokové schéma procesoru Pentium Pro
- Přímo do procesoru dána pamět cache L2 -> samostatný čip pracující na stejné frekvenci jako samotný procesor
- Bylo nutné pro zvýšení výkonu zvolit RISC architekturu
- Nutnost zachovat kompatibilitu s instrukční sadou pro i8086 + rozšířit o pár instrukcí
- Fetch/Decode jednotka z paměti instrukce x86 vybírá 118bitové RISC instrukce neboli mikro-operace
- Dekodované instrukce se neskládají do fronty, ale do nové banky dekodovaných instrukcí -> intruction pool
- Do této banky se vleze až 40mikro-operací a díky tomuto banku si může jednotka vybírat jednotlivé libovolné instrukce out-of-order -> konzistentnější datový tok
- Tyto provedené instrukce jsou uloženy zpátky do banky odkud je vybírá a ukončuje retire-unit -> odtud plynou data zpět do registrů a paměti cache L1 a přes rozhraní L2 zpátky do hlavní paměti
- Nutno se podívat na podrobnější schéma procesoru na obr. 9
- Sama jednotka se díky bance může rozhodovat co bude dělat, takže sama maximalizuje svůj potenciál
- Predikce skoků může obsahovat až 512 hodnot a její přesnost je cca 90%

Intel Pentium II

Výroba	1997
Technologie	BiCMOS, $0.35 \div 0.18\ \mu m$
Tranzistory	jádro $7.5 \cdot 10^6 + 20 \cdot 10^6$ pro 265kB cache L2
Frekvence	233 MHz ÷ 533 MHz
Datová sb.	64 bit
Adresní sb.	36 bit

- Žádný extra pokrok, větší množství tranzistorů
- Procesor přebírá řešení z Pentia pro
- Rozšíření o jednotku MMX
- Kvůli vyšším frekvencím procesoru se musela podtaktovat pamět L2 na polovinu frekvence procesoru
- Roku 1998 měla nejvýkonnější serverová konfigurace 512kB cache L2 taktovanou frekvencí samotného procesoru
- Tyto high end procesory nesly označení Xeon
- Teprve v roce 1999 byla představena integrovaná pamět cache L2 256kB společně s procesorem na jednom čipu -> procesor byl určen pro přenosné počítače

Intel Pentium III

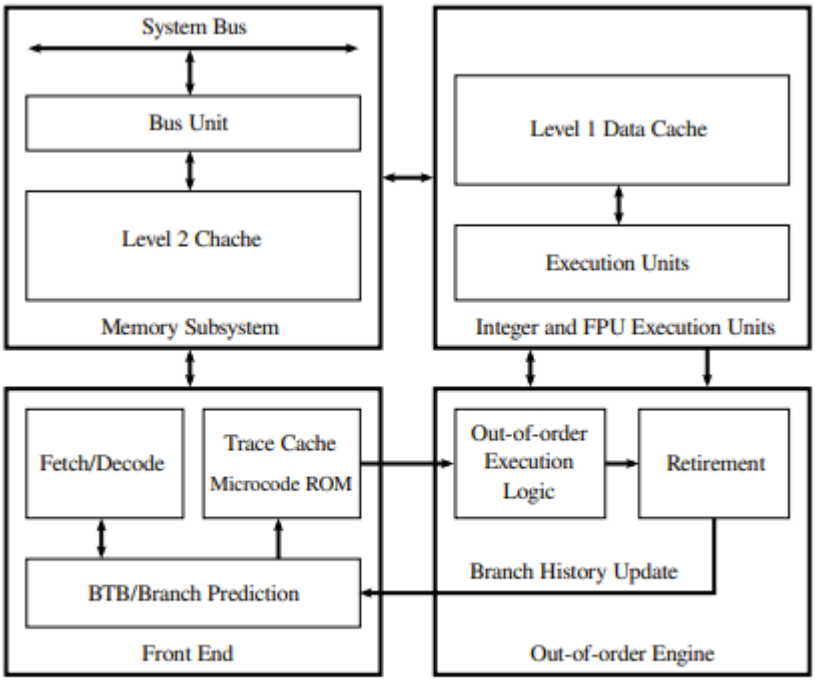
Výroba	1999 ÷ 2003
Technologie	BiCMOS, $0.25 \div 0.13\ \mu m$
Tranzistory	jádro $9.5 \cdot 10^6 + 18.5 \cdot 10^6$ pro 265 kB cache L2
Frekvence	450 MHz ÷ 1.4 GHz
Datová sb.	64 bit
Adresní sb.	36 bit

- Rozšíření o prováděcí jednotky 10x
- Pro uživatele bylo podstatné přidání SSE
- Zlepšení predikce skoků, optimalizace řízení obvodů procesoru s ohledem na minimalizaci spotřeby
- Několik let nejvhodnější procesor pro přenosné počítače

Intel Pentium 4

Výroba	2000 ÷ 2008
Technologie	BiCMOS, 0.18 ÷ 65 nm
Tranzistory	$42 \cdot 10^6$ včetně 256 kB cache L2
Frekvence	1.3 GHz ÷ 3.8 GHz
Datová sb.	64 bit
Adresní sb.	36 bit

- Nová mikroarchitektura NetBurst
- Reklamy propagovaly změnu pro multimediální svět
- Procesor však nebyl nějak výkonnější na stejné frekvenci a velmi se hřál
- Intel hodil klacek pod nohy sám sobě



Obrázek 4: Zjednodušené blokové schéma procesoru Pentium 4

- Rozdělení na 4 logické obvody
- Superskalární dekodovací jednotka předá mikrooperace do banky instrukcí -> vykonání pomocí superskalární jednotky mimo pořadí
- Cache L1 pro instrukce se změnila v Trace Cache
    - Instrukční cache obsahuje dekodované mikro-operace
    - Tímto se šetří práce dekodovací jednotce, nemusí tolik akcí opakovaně dekodovat
- Další změny jsou pouze v podrobném nákresu na obr. 10
- Celočíselné jednotky mají 2x frekvenci, než zbytek procesoru
- Predikce skoků je díky vyrovnávací paměti dvouúrovňová
    - Jedna před dekodováním
    - Druhá pomáhá s předáváním do banky instrukcí
- NetBurst zřetězuje jednotky 20 úrovňově, to je dvojnásobek oproti PPro

Intel Pentium 4 EM64T

Výroba	2004
Technologie	BiCMOS, 90 nm
Tranzistory	$125 \cdot 10^6$ včetně 1 MB cache L2
Frekvence	2.8 GHz
Datová sb.	64 bit
Adresní sb.	40 bit

- Zásadní milník vývoje řady x86 -> 64bitů
- Původně chtěl přejít na 64bitů pomocí modelu Itanium
- Kvůli uspěchu AMD však byl intel donucen strategii narychlo přehodnotit -> převzít 64bitový návrh od AMD
- I za cenu neoptimálního řešení se roku 2004 objevuje první procesor s technologií EM64T (později rename na Intel 64)
- Zdvojnásobení registrů, zvětšení registrů na 64bitů, datová sběrnice 64bitová

- Pro dosažení potřebného výkonu bylo aplikováno 30 úrovňové zřetězení
- Procesory museli být taktovány na 3GHz a výše aby podávali potřebný výkon -> přehřívali se
- Jejich ztrátový výkon byl od 85-115W

Intel Pentium M

Výroba	od 2003 ÷ 2008
Technologie	BiCMOS, 130 nm ÷ 90 nm
Tranzistory	$77 \cdot 10^6$ včetně 1 MB cache L2
Frekvence	900 MHz ÷ 2.2 GHz
Datová sb.	64 bit
Adresní sb.	32 bit

- Pentium M bylo původně představeno pro přenosné počítače
- Velmi výkonný procesor s nízkou spotřebou
- Vývojáři převzali energeticky usporný P3 a využili zkušenosti z P4
- Systémové sběrnice z P4
- Vylepšení dekódování instrukcí, zlepšení predikce skoků
- Přímo na čipu byla 1MB cache L2
- Tento procesor vykazoval při taktu na 1,5GHz jen o trochu menší výkon než P4 na 2,5GHz, toto ovšem zvládl se třetinovou spotřebou
- Tento procesor se dával do přenosných zařízení v desktopech skoro nebyl

Intel Core, Core Duo, Core Solo

Výroba	od 2006
Technologie	BiCMOS, 65 nm
Jádra	1, 2
Frekvence	1.5 GHz ÷ 2.2 GHz
Datová sb.	64 bit
Adresní sb.	36 bit

- PM pokračovalo jako procesory označené Core
- Rozšíření adresní sběrnice na 36bitů
- Díky 65nm se příkon procesoru snížil tak, že bylo možné na jeden čip dát dvě jádra i pro přenosné počítače
- Cache L2 se rozrostla na 2-4 MB a bylo společná pro obě jádra
- Stával se i součástí desktopů, né jen přenosných PC

Core 2

Výroba	od 2006
Technologie	65 nm, 45 nm
Tranzistory	$291 \cdot 10^6 \div 2.3 \cdot 10^9$
Jádra	1, 2, 4
Frekvence	1 GHz ÷ 3.3 GHz
Datová sb.	64 bit
Adresní sb.	36 nebo 40 bit

- Navázání na Core
- 64bit (EM64T)

- Tímto procesorem se ukončilo vývoj P4 na NetBurst
- Modernizace jádra procesoru -> schéma na obr.11
- Posílení dekodovací jednotky, přidání prováděcích jednotek
- Lepší cesta z instrukční cache L1 až do banky instrukcí

Intel Atom

Výroba	od 2008
Technologie	45 nm
Tranzistory	$47 \cdot 10^6$ včetně cache L2
Jádra	1, 2
Frekvence	800 MHz ÷ 2 GHz
Datová sb.	64 bit
Adresní sb.	32 bit

- Konkurence pro AMD Geode a VIA Eden/C7 ultralehké procesory s nízkým příkonem
- V roce 2008 intel kvůli tomuto vydal Atom
- Na první pohled se jednalo o ochuzený Pentium M
- Reálně ale Atom měl úplně jinou architekturu Bonnell která neměli nic společného s PPro a NetBurst -> ppouze dekodování CISC instrukcí x86 na mikro operace
- Architektura zase popsána na obr. 12
- Implementace HT (hyperthreading) -> fronta instrukcí a sada registrů je zdvojená
- Logické celky procesoru nazývány clusters
- Při odmyšlení HT je fungování docela snadné
- Instrukce jsou z paměti vybrané jednotkami XLAT/FL a ukládány do fronty, z této fronty se pomocí mikro operací si je vybírají prováděcí jednotky, a to bud FP/SIMD nebo celočíselná
- Výpočet adres (AGU) a řízení vyrovnávací paměti má na starosti jednotka správy paměti
- Komunikaci jednotka BIU
- Jednoduchý procesor který je směrován ke snížení spotřeby -> CCA poloviční výkon oproti Pentium M

Itanium, Itanium 2

Výroba	od 2001 / 2002
Technologie	180 nm ÷ 65 nm
Tranzistory	$220 \cdot 10^6 \div 2 \cdot 10^9$
Jádra	1, 2, 4
Cache L3	1.5 ÷ 24 MB
Frekvence	733 MHz ÷ 1.7 GHz
Datová sb.	128 bit
Adresní sb.	50 bit

- Přestaven 2001 -> Itanium 2 -> 2002
- Itanium 2 je na trhu dodnes
- RISC určený pro výkonné servery
- Zahájení vývoje roku 1999 HP a dokončen Intelem
- 64bit IA-64 -> nová instrukční sada od procesorů x86
- Využívá paralelismus na urovni vykonávaných instrukcí a potřebné pořadí instrukcí musí být připraveno překladačem

- Vnitřní schéma na obr. 13
- Slabina procesoru byla zpětná kompatibilita s 32bit architekturou x86
- Starší aplikace kvůli tomu běželi hrozně cca jako na Pentiu 100MHz
- Roku 2006 byl navrhnut emulátor který byl rychlejší než HW
- Moc se neprosadil kvůli 32bitům
- V roce 2008 4. nejprodávanější procesor -> dostupné systémy windows server, komerční UNIXy i několik Linux distribucí
- **VŠECHNY OBRÁZKY JSOU VE SKRIPTECH**
    - https://poli.cs.vsb.cz/edu/apps/down/intel.pdf

Monolitické Počítače

- PC se skládá z CPU, paměti a periférie
- Monolitické počítače jsou počítače, které jsou na jedné desce všechny tyto komponenty, takže tvoří pomyslné pouzdro
- Jiné názvy pro tyto počítače jsou: mikrokontrolery, jednočip, mikropočítač, mikročip
- Využívají hlavně harvardskou architekturu, protože odděluje paměti -> kvůli tomu, že paměti mohou být vyrobeny jinou technologií a mají jiné velikost buněk
    - Např data jsou po bitech, ale při bitových slovech můžou být až 16 bitů velké
    - To stejné strojové instrukce (12,14,16,24,32)
- V dnešní době jsou monolitické počítače spíše ve zjednodušené formě RISC
- Nelze zobecnit a generalizovat architektura

Paměti monolitických počítačů

- Data a Program pamět
- Pro data používáme většinou energeticky závislé paměti typu RWM/RAM (read-write memory a random access memory)
    - Tedy paměti s náhodným přístupem pro zápis i čtení
    - Jedná se o statické paměti
    - Bunky jsou klopné obvody
- Pro program se používají energeticky nezávislé paměti, takže ROM (read only memory)
    - Nejčastěji se dneska používají: EPROM a flash EEPROM
    - Někdy také mají pamět PROM -> OTP (one time programable)
    - Některé počítače nejsou osazeny ROM pamětí, takže musíme připojit externí, označují se jako ROM-less

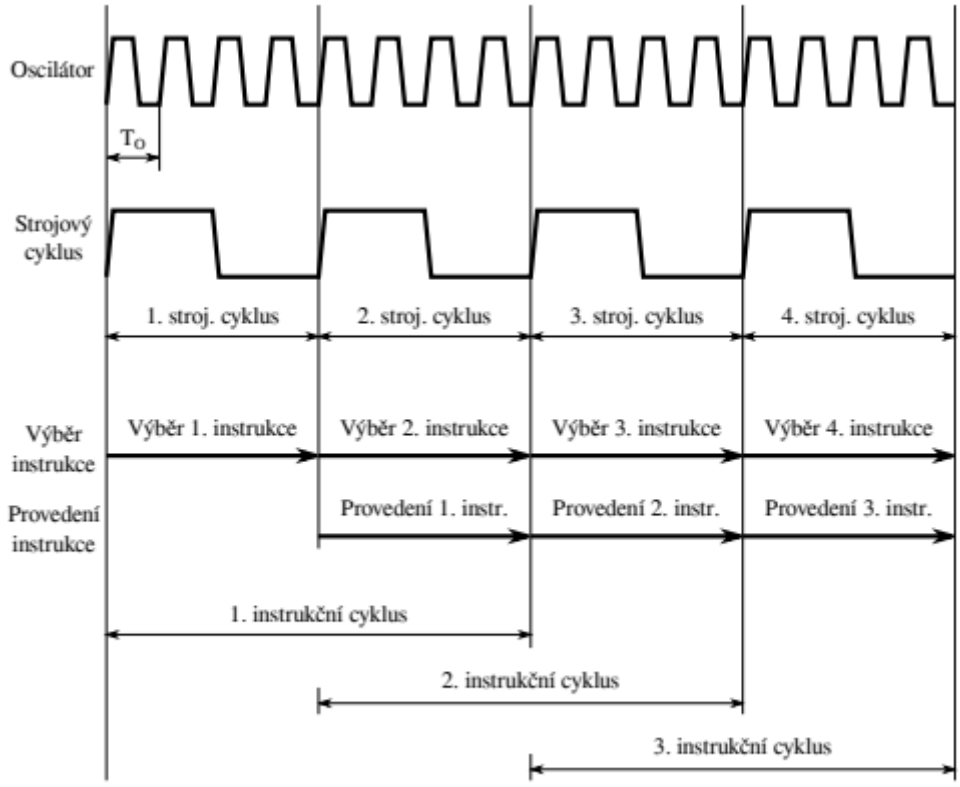
Organizace paměti dat

- Podle potřeby můžeme použít 3 základní typy pamětí
    - Stradačové (pracovní) registry:
        - Většinou jsou v procesoru jeden nebo dva
        - Ukládají se zde aktuálně zpracovávaná data
        - Nejčastěji operandem strojových instrukcí
        - Také výsledky operací
        - Krátkodobé uložení dat
    - Univerzální zápisníkové registry
        - Nejčastěji používaná data
        - Instrukční soubor dovolí, aby se část strojových instrukcí prováděla přímo těmito registry
        - Většinou malý rozsah adresovatelných registrů
        - Vyhnout se omezení dá tak, že založíme stejné skupiny registrů vedle se a budeme se moci mezi nimi přepínat -> registrové banky
    - Paměť dat RWM
        - Rozsáhlejší nebo méně používaná data
        - Instrukční soubor nedovoluje přímo pracovat s touto pamětí, kromě instrukcí přesunových -> např přesunutí do pracovního registru
        - Některé procesory umožnují, aby tyto data byly jako druhý operand strojové instrukce

        - Výsledek nelze do paměti uložit přímo
- Další speciální registry: čítač instrukcí -> ukazuje která instrukce se právě provádí a je v instrukčním registru
- Pro ukládání návratových adres slouží zásobník, ten může být přímo v paměti anebo ve speciální paměti LIFO, jejíž obsah nelze programově číst
    - Ještě potřebujeme jednoúčelový registr, který ukazuje na adresu vrcholu zásobníku
    - Hloubka zásobníku bývá na mikropočítačích omezena na 2-8 úrovní

Zdroj synchronizace

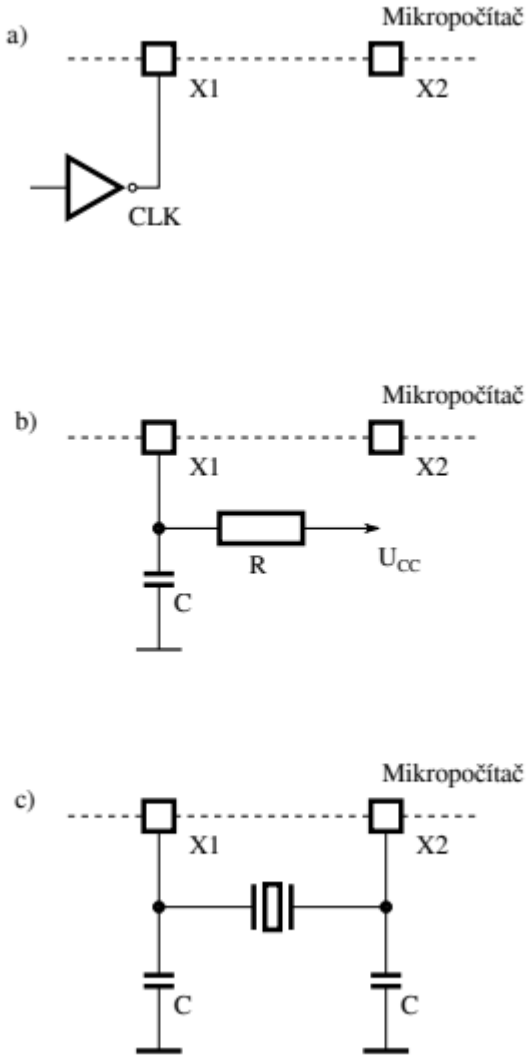
- Tyto počítače stejně jako jiné sekvenční obvody vyžadují pro svou činnost zdroj synchronizace
- Ten časuje kdy se co má dít basically řečeno



Obrázek 1: Příklad časových závislostí v počítači

- Tady jde o příklad procesoru, kde se strojový cyklus vypočítá z hodinového dělením čtyřmi
- Zase platí že se v jednom cyklu musí dokončit jedna instrukce -> výběr strojové instrukce/vykonání strojové instrukce jsou 2 cykly chápem se ne? 😊
- Pokud je synchronizační čip přímo na čipu, není možné zajistit úplnou přesnost -> vliv teploty atd… může docházet k odchylkám i 10%
- Častěji se používají generátory, které nechají uživatele zvolit kmitočet pro řízení
- Většinou jsou na pouzdře dva vývody pro časovací prvky

- Obvykle se používá:



Obrázek 2: Zdroje synchronizace a) externí zdroj b) RC c) s krystalem

-
- Krystal, keramický rezonátor, obvod LC, obvod RC
- Pro stabilitu je vhodný keramický rezonátor nebo krystal
    - U krystalů se musí vybrat správné hodnoty
    - Pokud je to budget na minimum, tak pak RC oscilátory
        - Kmitočet je ovlivněný teplotou i velikostí napětí
- Je potřeba myslet na to že CMOS technologie je proudový odběr závislý na řídící frekvenci
- Proto je potřeba volit nároky
- Řada mikropočítačů mají speciální režimy – IDLE, SLEEP, HALT -> přepínání do úsporného režimu -> sníží se kmitočet nebo se úplně zastaví

Reset monolitického počítače

- Počítač je sekvenční obvod -> závislost na strojových instrukcích a jejich argumentech, stavech, které předchází instrukci, takže vstupní něco, co dovnitř vchází
- Takže aby šel program spustit musí být správně definován počáteční stav PC -> takzvaný RESET -> v počítači jsou implementovány inicializační obvody, které toto zajistí

- Po provedení inicializace se nastaví čítač instrukcí většinou na nulu nebo na samé jedničky
- Oproti sekvenčnímu obvodu je však microPC tvořen mnoha na sebe navazujícími obvody
- Proto je RESET tvořen sérií navazujících činností -> inicializovat registry, procesor a periferní obvody
- Proto výrobci říkají, jaký čas musí RESET trvat, aby bylo vše provedeno správně
- RESET může mít vnější nebo vnitřní zdroj -> obvykle realizován tlačítkem s RC obvodem, aby se zajistilo správné časování
- I když je toto zapojení velmi jednoduché, je velmi důležité, aby bylo vše provedeno správně a věnovat pozornost dokumentaci výrobce
- Schmittův klopný obvod -> jakmile zaznamená potřebné napětí, tak se inicializuje správná inicializace

Ochrana proti rušení

- Velmi důležité u mikroPC -> většinou špatné prostředí -> třeba arduino u mě na zemi v prachu 😊
- Mechanické zabezpečení -> musí jej být bezpečné připojit na desku bez nutnosti dalšího mechanického zabezpečení -> nárazy/vibrace atd…
- Musí být odstíněn nebo galvanicky oddělit od okolí -> kvůli elektromagnetickým vlnám
- Programátor nebo okolní vlivy -> čím dýl počítač běží, tím větší šance že se to vysere
- Pokud počítač „zabloudí“ kvůli vlivům okolí bývá implementován obvod WATCHDOG
- Běží nezávisle na časovači, pokud přeteče nebo podteče, provede pomocí vnitřního RESETu vnitřní reinicializaci mikroPC
- Pro tuto kalibraci je využívána speciální instrukce, takže pokud je v programu používaná často, nebude fungovat
- Další způsob ochrany je větší rozsah pracovního napětí
    - Pokud je rozsah třeba 3-6 a napětí klesne pod 5/4V, tak nedojde v mikroPC k žádným nežádoucím změnám hodnot a kolizím
- Často se v mikroPC potkáme s obvodem, který hlídá pokles napětí pod minimální uroven, tak se automaticky provede RESET, tento obvod se často provede jako připojení na externí RESET signál

Přerušovací podsystém

- V dnešní době jsou procesory vybaveny přerušovacím systémem -> efektivnější programová odezva na změny stavu periferií a chování okolí
- Řeší se zde problematika toho, že příkazy na přerušení přichází naprosto asynchroně a procesor musí být připraven přerušit se kdykoliv, takže pak se musí uložit registry a pak se nahrát zpátky
- Mělo by tedy být důležité, jak povolit a zakázat přerušení a jak detekovat co je zdroj požadavku na přerušení -> může se využít priorita
- Nebo se taky používá vnoření, kdy se provede více přerušení najednou
    - Toto může být někdy problém, protože se to pak rekurzivně zacyklí a může to být nežádoucí (zaplní se zásobník adres atd…)

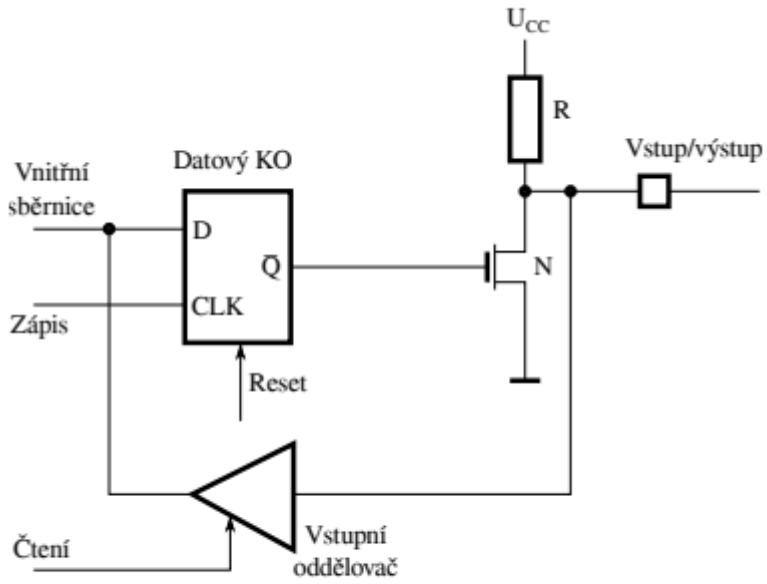
Periferie – obvody vstupu a výstupu

- Obvody, které zajišťují komunikaci s okolím -> periferie
- Jejich kvality určují efektivnost systému

- Pokud například potřebuje modul komunikovat po sériové lince a mikropočítač to neumí, tak to musíme emulovat -> ztráta rychlosti

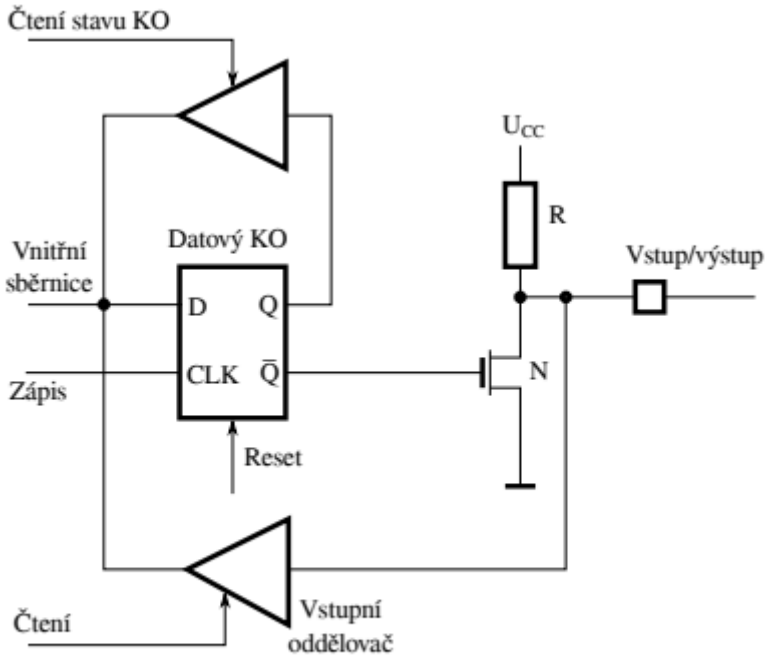
Vstupní a výstupní brány

- Nejčastější a nejjednodušší je paralelní brána – port
    - Skupina 4/8 jednobitových vývodů
    - Zároveň lze zapisovat a číst logické 1 a 0
- U většiny bran lze nastavit jaké bity budou sloužit jako input a jaké jako output
- Brány jsou implementovány tak, že s nimi instrukční soubor komunikuje jako s množinou vývodů nebo jako s jednotlivými bity
- Rozsah použití těchto univerzálních I/O je insane
- Lze s nimi dělat dost věcí -> čtení hodnot dvoustavových snímačů, ovládání přepínačů, komunikovat po synchronní/asynchronní lince

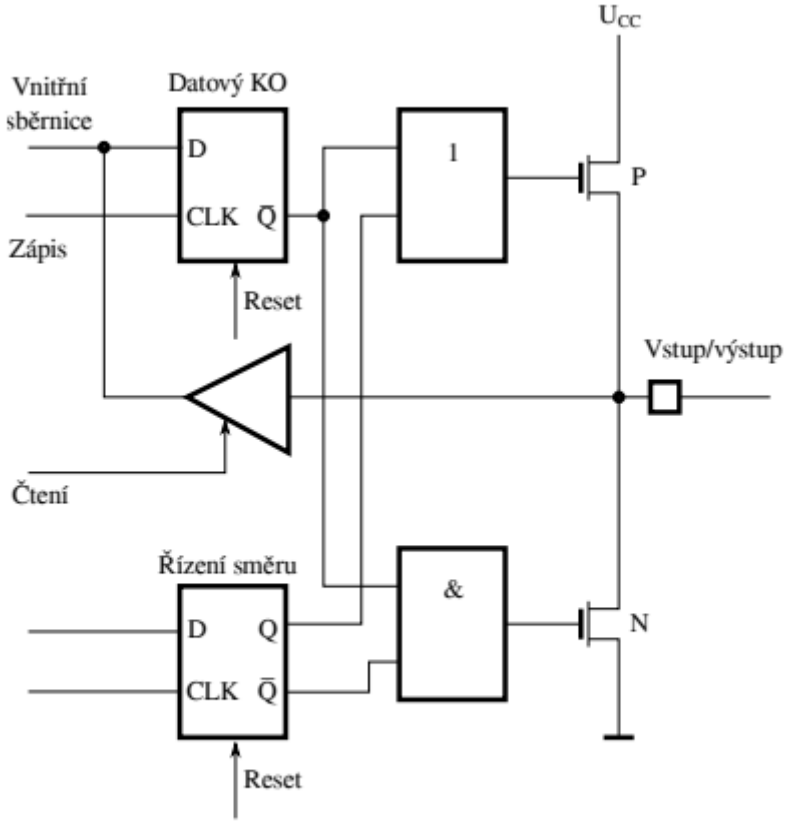


- Obrázek 3: Struktura V/V bran - otevřený kolektor
- Obecně platí, že programové řešení může ubrat pár speciálních obvodů
- Musíme však zvážit, jestli je to výhodné vzhledem k výkonu procesoru
- Vnitřní strukturu oboustranných I/O k řízení směru přenosu se používá způsob známý ze sběrnicových systémů číslových počítačů
- První způsob jsou analogické spojování logických obvodů s otevřeným kolektorem obr 3
    - Řízení směru komunikace není nutný samostatný registr
    - Pokud zapíšeme logickou 1, tranzistor se nesepne a vývod je možné použít jako vstupní
- Obr 4 je podobný
    - Liší se jen tím, že se data zapíšou zpátky do záchytného registru a nemusí být stejná jako vstupní
- Další příklad na obr 5
    - Zapojení s tranzistory N a P

    - Přepínání směru se řeší speciálním klopným obvodem, který povoluje sepnutí výstupních tranzistorů
    - U oboustranných bran musí vývojáři dát pozor, aby se neobjevily dva výstupy s opačnou logickou úrovní, proto jsou většinou brány během RESETu nastaveny na vstupní režim
- Kromě těchto uvedených zapojení mohou být na obr 5 vidět zapojení pull-up a pull-down odpory které zajišťují po přepnutí do vstupního režimu definovanou hodnotu logické hodnoty 0 nebo 1
- Na vstupu hradla také můžeme použít Schmittův klopný obvod
    - Zajistí definované chování při pomalých přenosech hodnotami 0–1 a zpět
- Vnitřní zapojení také silně ovlivňuje elektrické vlastnosti hradla -> proto se zveřejnují dokumentace, abys mohl udělat to nejlepší zapojení possible, jako náš mentor pan profesor Petr Olivka



Obrázek 4: Struktura V/V bran - s odděleným KO



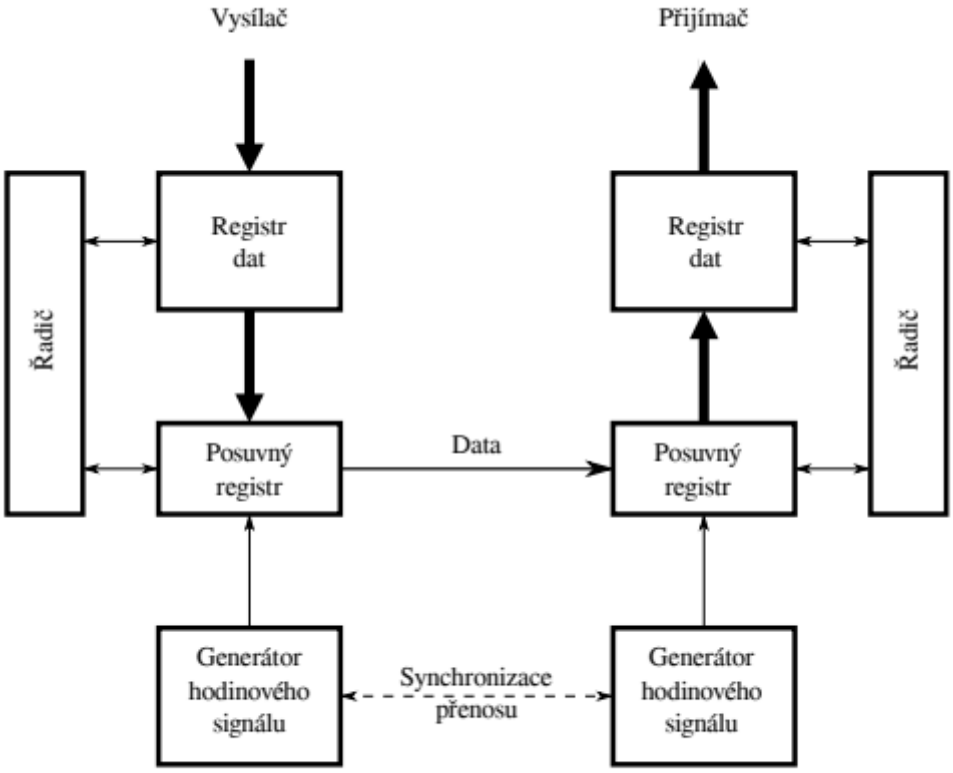
Obrázek 5: Struktura V/V bran - Dvojčinný výstup s volbou směru

Sériové rozhraní

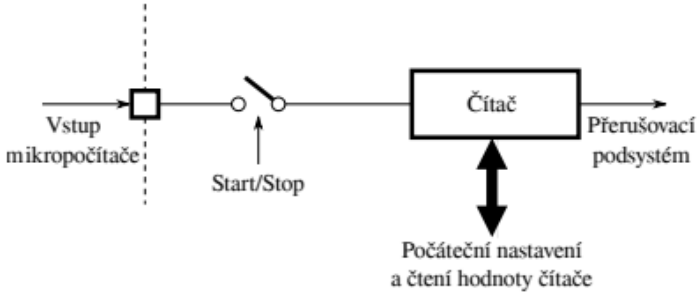
- Sériový přenos se používá čím dál více
- Přenáší data na velkou vzdálenost pomocí mála vodičů
- Hlavní nevýhoda je malá přenosová rychlost
- Další nevýhoda je, že cestují po jednom vodiči, takže se všechna data musí vystřídat na jenom vodiči
    - Je tedy nutné je dobře zakódovat a na konci dekódovat
- Používá se často = hodně standardů, které se liší
    - Počtem přenosových vodičů, velikost napěťové hodnoty pro logické hodnoty, rychlost přesunu, zabezpečení atd…
- Podle vzdálenosti přesunu děláme komunikace na dvě základní kategorie:
    - Komunikace mezi elektrickými zařízeními
        - Vetší vzdálenost -> např řídící počítač a podřízené stanice
        - Jde o přenos synchronní nebo asynchronní
        - Typicky pomocí RS232 nebo RS 458
    - Přenos uvnitř elektrického zařízení
        - Přenos mezi integrovanými obvody
        - Typický standard je I*nadruhou*C pro komunikace mezi integrovanými obvody

Sériový synchronní a asynchronní přenos dat

- Nejstarší a nejčastěji používaný způsob komunikace
- Každý PC je dneska vybavený sériovou linkou RS232
    - U které by mělo být možné nastavit parametry přesunu, rychlost, parita, počet bitů ve slově a zabezpečení
- Napěťové hodnoty pro logické 0 a 1 jsou dané konverzí TTL -> speciální obvody
- Na obr 6 je blokové schéma sériového přenosu dat jedním směrem, při synchronním přenosu jsou generátory hodinového signály synchronizovány a čekají jeden na druhého -> zabezpečenost signálu a správnost bitů
- Nevýhoda synchronizace je ta, že potřebuje další vodič nebo pomocné obvody, které rekonstruují



Obrázek 6: Sériový přenos



Obrázek 7: Čítač vnějších událostí

- A asynchronního se počítá s menší odchylkou u přijímače a vysílače
- Sériová linka zůstává v nepřítomnosti dat v klidovém stavu v urovni logické 1
- Přenos dat se zahajuje START bitem, který je vždy 0
- Přijímač musí detekovat sestupnou hranu na lince a od toho okamžiku zahajuje proces příjmu dat
- Délka souvisle přenesených dat je pak omezena, aby se odchylka nevymkla mezím
- Výhoda je volnost komunikace, nevýhoda rychlost, protože se pokaždé musí přenést signál START a STOP

Čítače a časovače

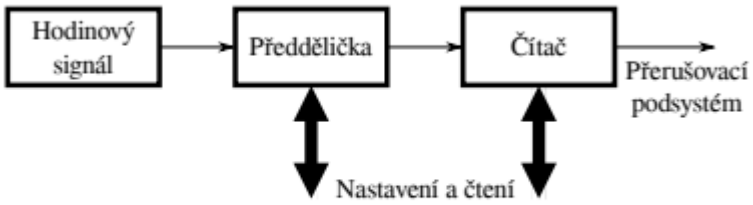
- Nejpoužívanější periferie mikroPC

Čítač vnějších událostí

- Jedná se o registr o N bitech, který čítá vnější události
- Obvykle pozná, zda se jedná o náběžnou nebo sestupnou hranu vnějšího signálu (inkrementací)
- Při přetečení se předá automaticky výzva do přerušovacího podsystému mikropočítače
- Počáteční hodnota čítače se nastavuje programově, čítač lze odpojit a připojit k vnějšímu signálu
- Příklad zapojení na obr 7

Časovač

- Neliší se příliš od čítače
- Není inkrementován vnějším signálem, ale vnitřním hodinovým signálem používaným samotným mikroPC, tímto lze uměrně zajistit přesnost zdroje hodinového signálu a řízení chování v reálném čase
- Pokud časovač přeteče opět se předá signál na přerušení
- Předděličkou jde přizpůsobit chování časovače
- Schéma na obr 8



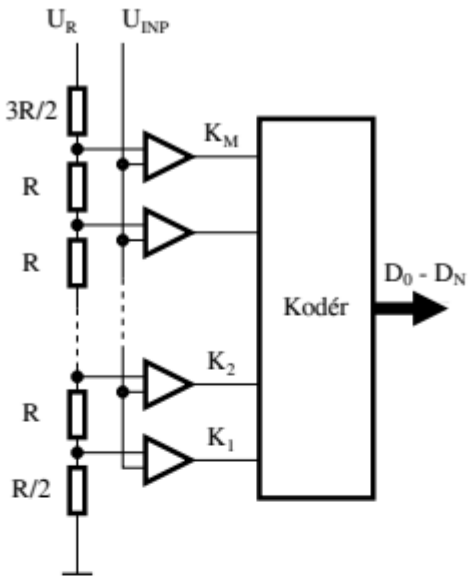
Obrázek 8: Časovač

-

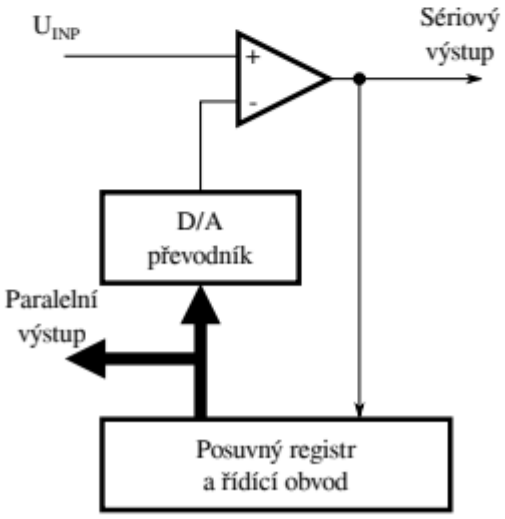
AD převodníky

- Fyzikální veličiny které vstupují do mikro PC jsou většinou analogové
  - Napětí, Proud, Odpor
- Pro práci však potřebujeme digitální hodnotu neboli číselnou formu
- K tomuto slouží AD – DA převodníky
- Existuje několik druhů nejčastější tu jsou

- Komparační AD převodníky
    - Porovnávání měřené veličiny (napětí) s referenční hodnotou, rozdělenou na několik hodnot v určitém poměru, například odporem
    - Rozdělením původního napětí na několik hodnot dostáváme paralelní převodník
    - Takovéto převodníky jsou velmi rychlé a hodí se pro zpracovávání signálů
    - S větším počtem komparátorů roste rozlišovací schopnost převodníku
    - Rozdělení měření do kroků snižuje nároky na komparátory, ale prodlužuje délku měření
    - Obrázek č.9



Obrázek 9: Komparační převodník



Obrázek 10: Převodník s postupnou aproximací

-
- AD převodníky s DA převodem
    - Základ těchto převodníků je použití jediného komparátoru a proměnného zdroje referenční hodnoty
    - Podle způsobů řízení referenční hodnoty rozlišujeme dva základní druhy převodníků
        - Sledovací
            - Mění referenční hodnotu o jeden krok nahoru nebo dolů
            - Není vhodný pro skokově se měnící veličiny – 8bitový převodník by musel projít všech 256 hodnot
            - Proto se hodí pro věci, kde se veličina mění pomalu, teplota/vlhkost

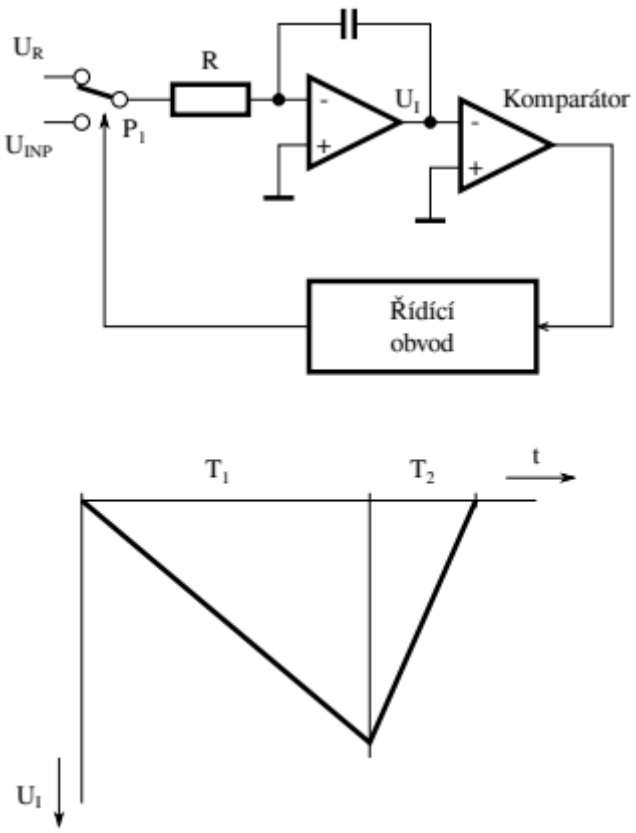
        - Aproximační
            - Využívá se systém půlení intervalu v setříděném poli
            - Referenční hodnota se nastaví na poloviční hodnotu mezi maximem a minimem
            - Poté se podle výsledku komparátoru posune o půlku tohoto pole nahoru nebo dolů
            - Počet porovnávacích kroků je tedy maximálně roven počtu bitů rozlišovací schopnosti převodníku
            - Na obr 10 jednoduché schéma
- Integrační AD převodník
    - Obr 11 schéma DA převodníku s velkou rozlišovací schopností cca 12 bitů a více
    - Integrátor integruje po pevně stanovenou dobu T1 vstupní napětí UINP
    - Po skončení této doby se přepne vstup P1 integrátoru na napětí Ur opačné polarity
    - Nyní se bude po dobu T1 integrovat referenčním napětím
    - Doba T2 je závislá na napětí na konci periody T1
    - Délka T2 je tedy dána vztahem:

$$T_2 = \frac{U_{INP}}{U_R} \cdot T_1$$

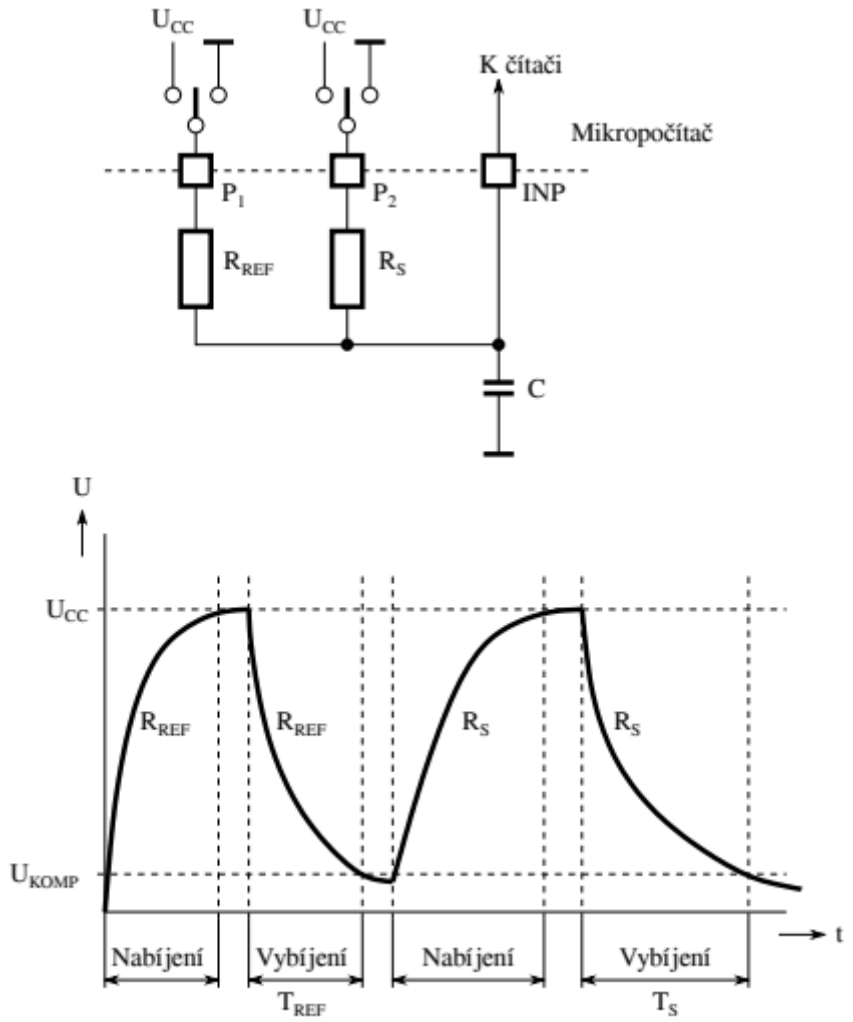
Pokud je T_1 a U_R dáno, změříme čas T_2 a můžeme vypočítat:

$$U_{INP} = \frac{T_2}{T_1} \cdot U_R$$

    -
    - Tento princip je metodou dvojité integrace
    - Kondenzátor použitý v zapojení intergračního převodu však nelze vyrobit jako součást integrovaného obvodu, proto se používá externě
- Převodníky s časovacím RC článkem
    - Né vždy je analogová veličina proud či napšetí
    - Často se měří něco pomocí odporu, zařízení, které mění svůj odpor v závislosti na okolí – termometry atd…
    - Zde by byl převod na napětí o jeden krok delší a to nechceme
    - Nejčastěji se používají RC články, kde se podle toho, jak dlouho se nabíjí/vybíjí kondenzátor je možné určit jaký je odpor
    - Měření času je lehce proveditelné čítačem nebo časovačem
    - Při měření je nutné srovnávat výsledek s referenčním odporem
    - Porovnáváním dvou odporů eliminujeme vlastnosti kondenzátoru



Obrázek 11: A/D převodník s dvojitou integrací



Obrázek 12: Převodník R/C s měřením doby vybíjení kondenzátoru

- Na obr 12 je zapojení snímače R, kondenzátor necháme nabíjet, až dosáhne požadované hodnoty, tak jej zase stejnou dobu budeme vybíjet, měli bychom se dostat na referenční hodnotu

  Ze dvou známých časů a hodnoty R_{ref} můžeme vypočítat:

$$R_s = R_{ref} \cdot \frac{T_s}{T_{ref}}$$

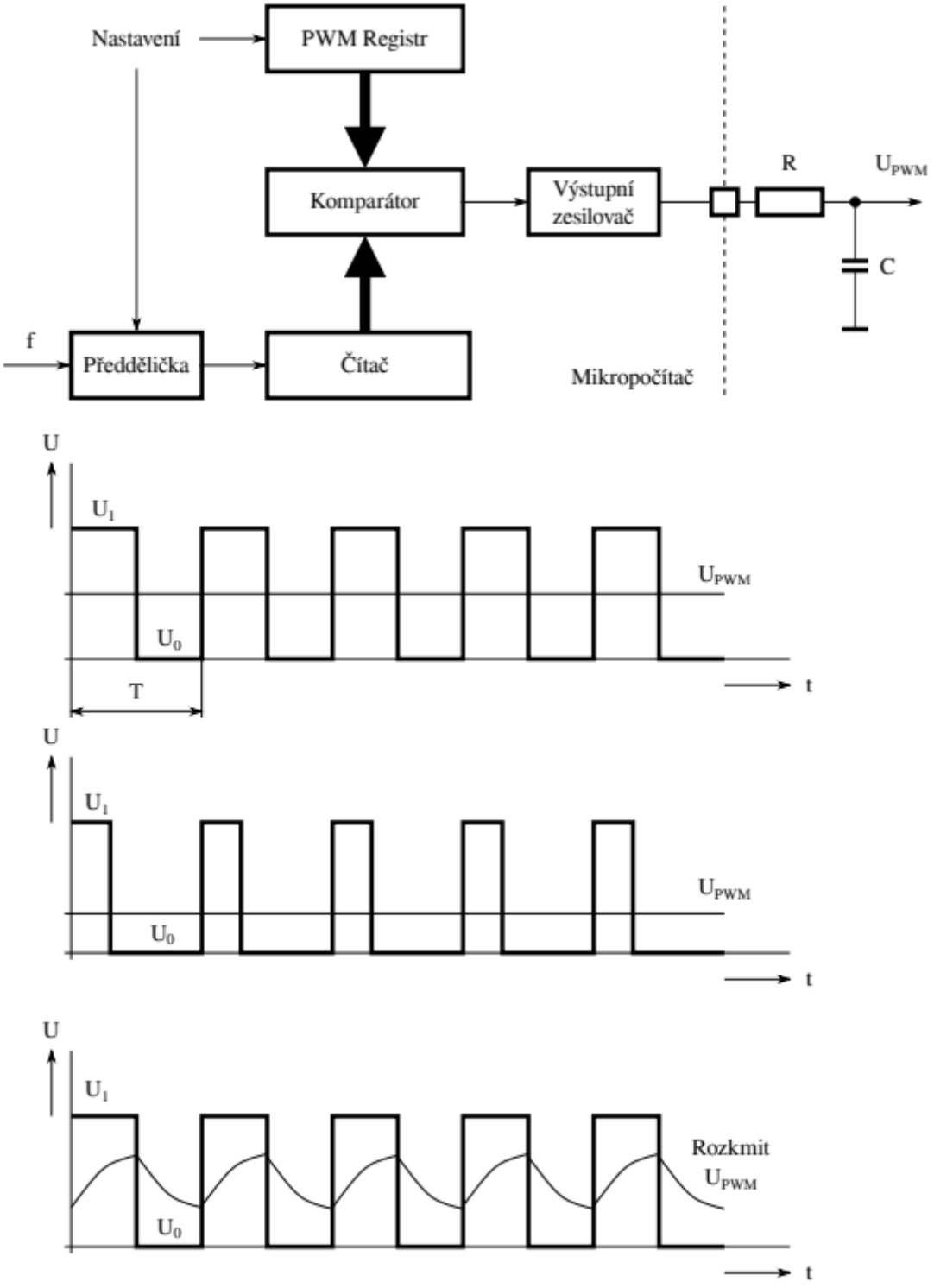
  Tímto jednoduchým způsobem získáme převodníky s rozlišovací schopností 12 a více bitů.

DA převodníky

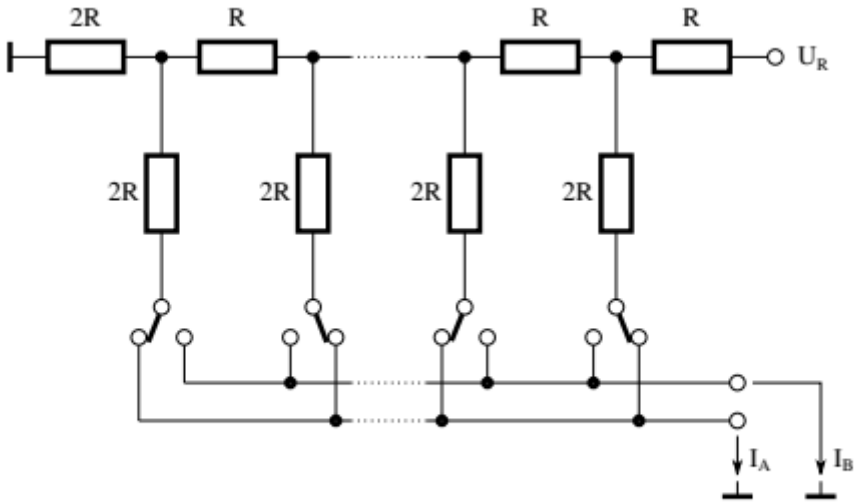
- Potřebujeme i zasahovat do prostředí -> tyto podněty musí být v analogové formě
- K tomuto použijeme D/A převodník, které používají základní dva principy
- PWM (pulse-width modulation / šířková modulace bitů):
  - Číslicový kanál má na výstupu mikropočítače obvykle dvě konstantní napětové urovně, logickou 1 při napětí U1 a logickou 0 při napětí U0
  - Pokud necháme na výstupu hodnotu 1 po dobu T1 a hodnotu 0 po dobu T0, můžeme vypočítat střední hodnotu napětí

$$U_{PWM} = U_0 + (U_1 - U_0) \cdot \frac{T_1}{T_0 + T_1}$$

- Součet časů T1 + T0 bude T -> časová perioda
- Popis chování na obrázku č 13
- Pro to, aby filtr fungoval správně je nutné, aby časová konstanta RC byla výrazně větší než T -> musí platit R*C>>T
- Nebude-li tato podmínka platit, bude výstup nestabilní -> vnitřní odpor se připočítá k RC článku a časovou konstantu ještě prodloužíme
- Časová konstanta RC článku určuje poměrně velké zpoždění převodníku na požadovanou změnu výstupu
- PWM lze realizovat programově nebo pomocí čítače
- Počet bitů použitého čítače určuje rozlišovací schopnost převodníku
- Výstupní hodnota může mít 2 na N různých úrovní



Obrázek 13: Příklad PWM převodníku a jeho chování



Obrázek 14: Příklad R-2R převodníku s proudovými spínači

- Paralelní převodníky
    - Základní nedostatek PWM převodníků je velké zpoždění, toto odstraňují paralelní převodníky
    - Jsou založeny na přímém převodu číselné hodny na stejnosměrný proud
    - Základem tohoto převodníku je většinou odporová sít, na níž se vytvářejí jednotlivé částečné proudy
    - Ta může být sestavena z odporů s váhově řazenými hodnotami -> výroba je poměrně náročná, protože je nutno dodržet poměr mezi odpory -> 1-2-4-8-16-32-64-128-256…
    - Proto se používá sít R-2R -> používají se jen odpory hodnoty R a 2R -> obrázek 14
    - Výstupem je proud Ia a komplementární proud Ib
    - Součástí převodníku musí být také zdroj referenčního napětí

Obvody reálného času (RTC – real time clock)

- Často je potřeba dodržovat časovou souvislost řízených událostí -> řízení v reálném čase
- Často se musí dodržovat skutečný čas ->hodiny, minuty, sekundy a popřípadě i zlomky sekund
- Pro řadu aplikací nestačí čas, ale chcou i kalendář -> dny měsíce roky -> k tomu slouží RTC
- Řeší se dva hlavní problémy
    - Když zařízení vypadne elektrický proud, tak ztratí data, takže i současný čas -> je potřeba udělat zálohování
    - Musíme ale myslet na spotřebu elektřiny -> většinou fungují z baterií atd… takže je to docela important
    - Druhý problém je čtení dat z RTC obvodu
    - Čas se neustále mění -> je složité číst všechny hodnoty
    - Proto se to řeší pomocnými registry v RTC obvodu nebo vhodným programovým řešením

Speciální periferie

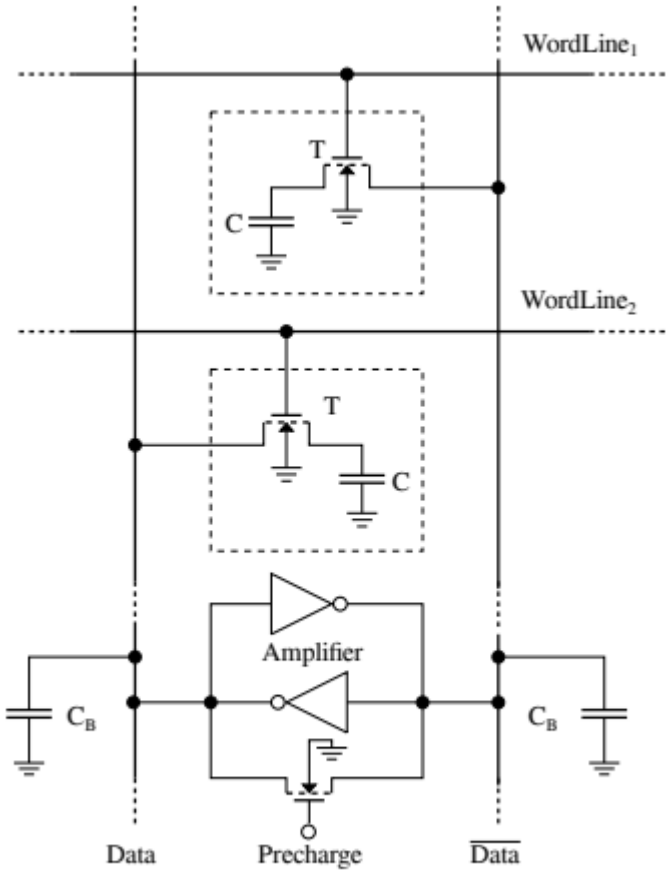
- Řada mikroprocesorů je navrhována pro speciální účely, takže potřebujeme i speciální periferie
    - Řízení dobíjení baterií, a nejen suchých typu NiCD, Li-Ion, ale i baterií s elektrolytem
    - Pro telekomunikační techniku, některé mikroPc jsou vybaveny dvoutonovým multifrekvenčním generátorem a přijímačem
    - Pro TV přijímače obsahují obvody zobrazovací informaci „On screen“
    - Vysílače a přijímače IR signálu, teda modulaci, demodulaci a kodování
    - USB rozhraní typu klient
    - Řadiče LCD a LED zobrazovacích jednotek, které vyžadují dynamické řízení

PAMĚTI POČÍTAČŮ

- 1955 -> feritová pamět -> zmagnetizovaná feritová jádra
- Bubnové paměti -> magnetický materiál nanesen na nemagnetický buben -> rotace velkou rychlostí
- Bublinové paměti -> magnetické paměti -> založené na využití velkokapacitních magnetických posuvných registrů
- Polovodičové paměti -> jedno/vícebitové posuvné registry
- 1960 -> polovodičová technologie MOS

Dělení

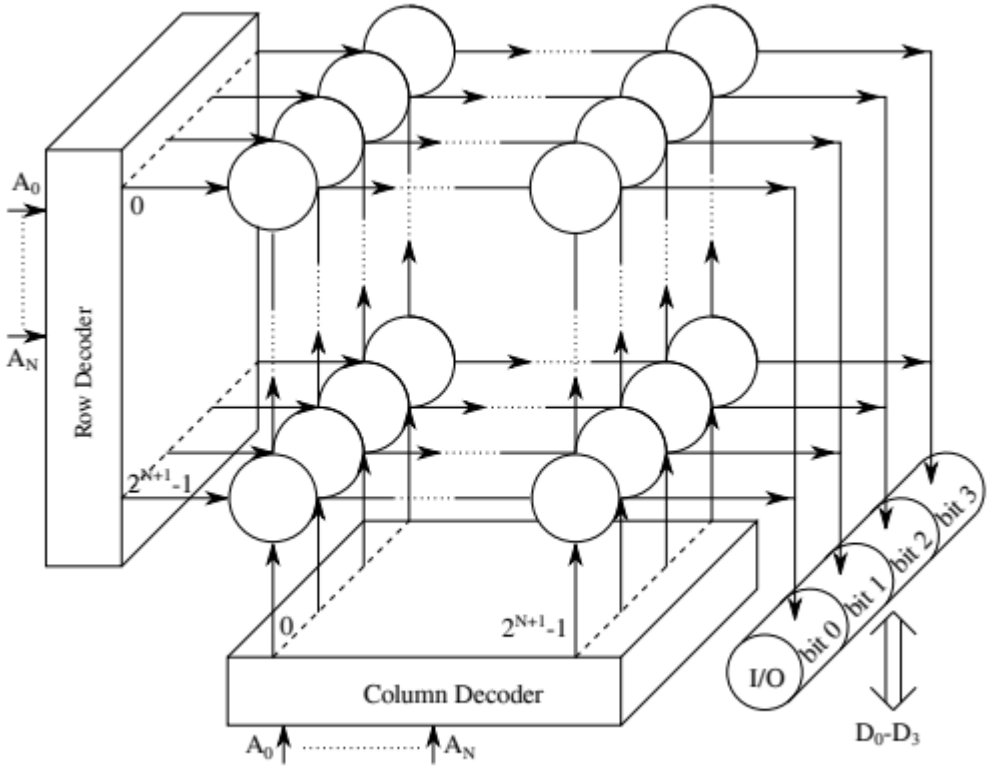
- RAM – random access memory -> paměti s libovolným přístupem
- SAM – serail access memory -> paměti se sériovým přístupem
- Paměti se speciálním způsobem přístupu -> asociativní pamět, pamět typu fronta, zásobník, vícebránové paměti, paměti s kombinovaným řízením
- RWM (read write memory) čte i zapisuje
- ROM (read only memory) jen čte
- Kombinované paměti
    - NVRAM (non volatile RAM) – kombinace RWM a EEPROM
    - WOM (Write only memory) – only zápis
    - WORM (write once read many times memory) -> optické disky neboli cédéčka
- Rozdělené podle elementárního principu
    - SRAM -> statické
    - DRAM -> dynamické
    - PROM, EPROM, EEPROM, FLASH ->Programovatelné paměti
- Podle uchovávání paměti pokud jsou bez napájení
    - Non volatile -> data zůstávají i po odpojení -> xxROM (EPROM atd…)
    - Volatile -> ztratí data po odpojení -> DRAM / SRAM
- Popis není uplně dokonalý -> např klasická ramka by byla něco jako volatile RWM DRAM



Obrázek 1: Paměťové buňky DRAM

Dynamické paměti

- Data v podobě náboje v kondenzátoru -> vybitý nabitý logická 1 0
- Obr 1 -> dvě jednotranzistorové pamětové bunky
- Tranzistory jsou miniaturní -> hodnoty jsou miniaturní -> jednotky v řádech desítek fF (femto faradu)
- Možnost uchovat informaci omezenou dobu -> nesrovnalost v proudu -> obnovovat napětí kondenzátoru -> občerstvení neboli refresh -> implementace spešl obvody -> přečte řadu buněk -> refreshne je



Obrázek 2: Organizace paměti DRAM

- Dynamické paměti zapomenou data po cca 10ms
- Na obrázku je kolečko = paměťová buňka -> čtvercová matice -> jedna nebo více vrstev -> data se vybírají pomocí dvou dekodérů -> přenos na I/O buffer
- Rozdělení na dva dekodéry je dobré pro adresování -> polovina adresních vodičů -> pro 1Mbit je místo 2na20 bude 2x 2na10
- Nejdřív adresa řádku až poté sloupce
- Menší počet vodičů -> zvětšená kapacita
- Ještě je potřeba pro výběr signály RAS a CAS

Konvenční DRAM

Organizace čipu DRAM

- Různé formy organizace kvůli vývoji
  - 1 Mbit čip s jedním datovým vývodem má organizaci 1Mword na 1bit
    - 1M slov o šířce 1bit na každý vývod
  - 1Mbit čip je 256 Kword x 4 bitová organizace -> 256Kwords se šířkou 1bity -> 1Mbit
- První je vždy slovo a poté šířka v bitech
- Hlavní vlastnost je šířka neboli počet bitů, po kterých může být slovo předáno

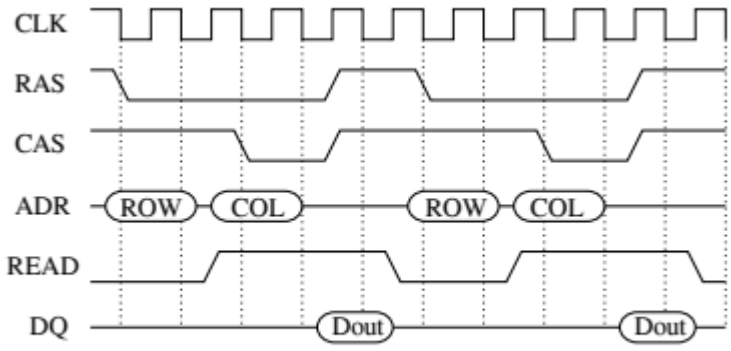
Princip činnosti DRAM

- CPU poskytne adresu -> adresový buffer dostane adresu paměti jako výstup externího paměťového controlleru -> adresa se rozdělí na dvě části -> adresa sloupce a řádku -> adresy jsou čtené pamětovým bufferem jedna po druhé -> multiplexing
  - Multiplexuje se kvůli tomu, že by byla potřeba moc adresových bitů a pamět by měla velké pouzdro

- Tato akce je kontrolovaná RAS a CAS (row/coll adress strobe) řídícími signály
- Pokud se pošle adresa řádku a zároveň se aktivuje RAS signál, tak DRAM ví že je adresa v pořádku -> aktivace adresového bufferu -> získání adresy -> přesune jí do dekodéru řádků -> dekóduje jí
- To samé u sloupců akorát s CAS
- Pamětová bunka takto adresovaná předá na výstup data -> zesílení čtecími zesiilovači -> I/O buffer
- Buffer nakonec poskytne Dout přes datové vývody paměťového čipu
- Pokud mají být data zapsána aktivuje se WE (write enable) signál a přesune zapisovaná Din data do I/O bufferu -> zesíleno pomocí čtecích zesilovačů -> přesun do adresované bunky
- Pamětový kontroler má 3 věci na práci
    - Rozdělení adresy získané z CPU na adresu řádku a sloupce -> přesunutí do paměti jedna po podruhé
    - Správná aktivace RAS, CAS, WE a READ signálů
    - Přesune uložená data a přijímá data k zápisu
    - Pokud by data z CPU nebyla upravena, tak by byl problém a adresy by nebyly vhodné

Čtení a zápis dat

- Kondenzátor udržuje náboj informace -> má přístupový tranzistor -> přepínač pro výběr kondenzátoru
- Báze (gate) tranzistoru je připojena na adresový vodič
    - Pole paměťových buněk má jen jeden adresový vodič číslovaný od 0 až (2naN+1) – 1 na každou zformovanou řadu
- Pole paměťových buněk obsahuje páry datových vodičů DATA a negace DATA -> existuje jeden pár na každá sloupec v poli pam buněk
- Datové vodiče se střídavě připojují k emitorům přístupových tranzistorů
- Jedna z elektrod kondenzátoru je připojena na kolektor odpovídajícího tranzistoru a druhá je uzemněná
- Jakmile je dána adresa -> všechny přístupové tranzistory k této adrese se zapnou
- Dekodér sloupce dekoduje adresu a podle toho zapne jeden datový vodič -> I/O buffer zesílí signál a předá ho na Dout -> přečtení jedné bunky vede k občerstvení celého řádku
- Obr 3 ukazuje chování nejdůležitějších paměťových prvků při přenosu dat
- Zápis je skoro stejný jako čtení -> Adresa řádku -> aktivace RAS -> Řídící WE signál -> Din do datového bufferu -> zesílení -> přepnutí do I/O bufferu

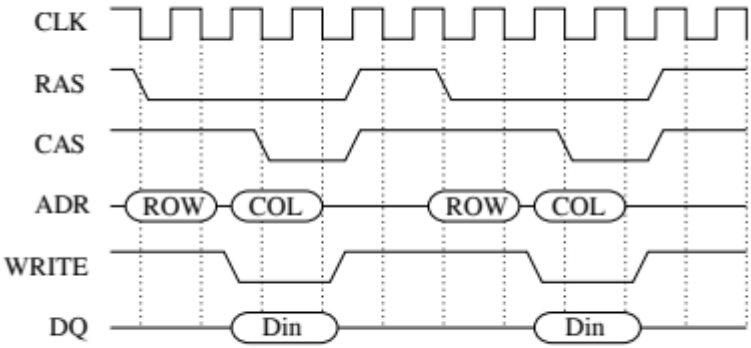


Obrázek 3: Časové schéma čtení z DRAM

-

- Dekoduje se signál adresy řádku -> aktivace odpovídajícího adresového vodiče -> obsažený náboj jde do vodičů DATA a negace DATA -> potom to stejné se sloupci pomocí CAS

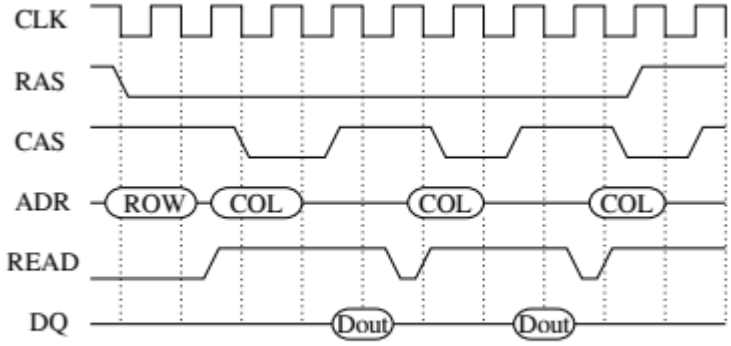
Obrázek 4 ukazuje chování nejdůležitějších paměťových signálů během vykonávání procesu zápisu dat.



Obrázek 4: Časové schéma zápisu do DRAM

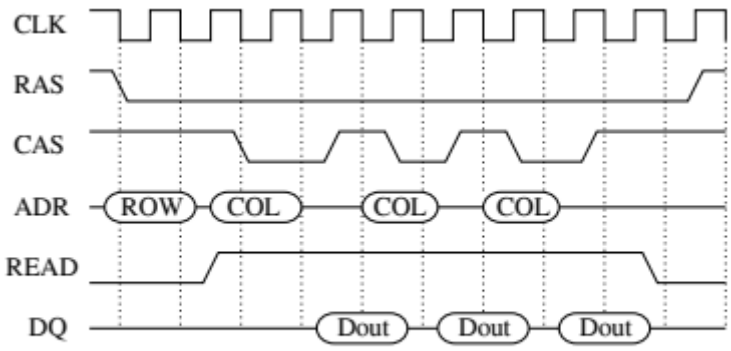
Další operační mody

- Page mode -> snížení doby přístupu obrázky 5–8
- V podstatě myšlenka je taková, že pokud jsou buňky na stejném řádku, tak nemusíme víckrát číst adresu stejného řádku -> jedna stránka = jeden řádek -> obr 5



Obrázek 5: Časové schéma čtení z FP DRAM

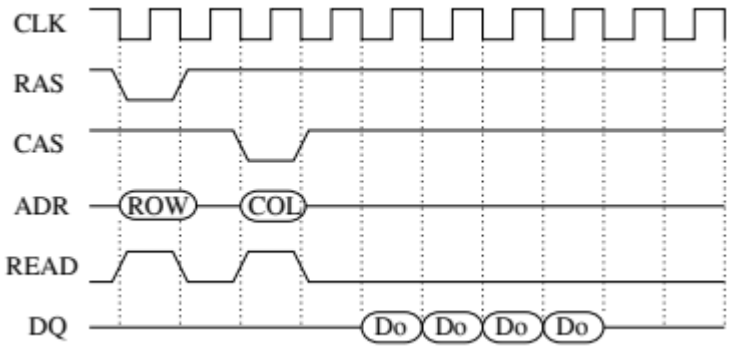
- Při stránkovém režimu se RAS nedeaktivuje -> pouze CAS signál se na krátkou dobu deaktivuje
- Všechny přístupové tranzistory jsou pořád sepnuty a všechna přečtená data jsou na konci datových vodičů
- Rychlost bývá o 50-70% větší, vztahuje se až na druhý až N vstup
- EDO Mod (extended data out)
- Časová vzdálenost mezi dvěma následnými CAS aktivacemi je kratší než u stránkového modu (obr. 6)
- Adresy sloupců se přesunují rychleji cca o 30% než ve stránkovém režimu
- V EDO musí CAS signál být deaktivován před poskytnutím nové adresy sloupce



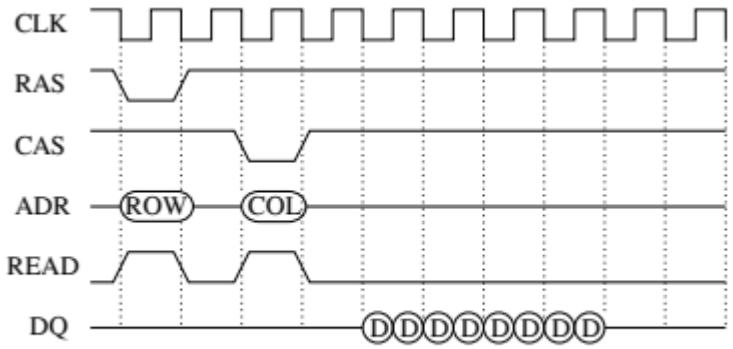
Obrázek 6: Časové schéma čtení z EDO DRAM

Vylepšené typy DRAM pamětí

- SDRAM
  - Synchronní DRAM
  - SDRAM mají nízkou přístupovou dobu 8-15 ns a mohou fungovat synchronně se systémovou taktovací frekvencí -> 66MHz, 133MHz nebo více
  - EDO DRAM mají přístupovou dobu 50 až 60 ns -> tudíž je vidět jak velký rozdíl mezi EDO DRAM a SDRAM je
  - Důvod pro toto je hlavně L2 cache
  - Rozdíl je patrný hlavně od taktu 100MHz a výše
  - Pracuje v Burst mode a se synchronní taktovací frekvencí, né s RAS a CAS časováním jak je tomu u jiných RAM čipů
  - SDRAM také používají signály RAS, CAS, WE, CE, ale požívají je pro zápis, čtení a burst stop
  - RAS a CAS jsou kombinovány pro vytvoření příkazové sběrnice viz obr 7



  - SDRAM používá prokládání paměťových polí, jedno právě čte a druhé se připravuje
- DDR SDRAM
  - Double data rate
  - Zpětně kompatibilní typ paměti -> vedl k modulům PC-266 -> vývoj kračuje až do této doby



- Obrázek 8: Časové schéma čtení z DDR SDRAM

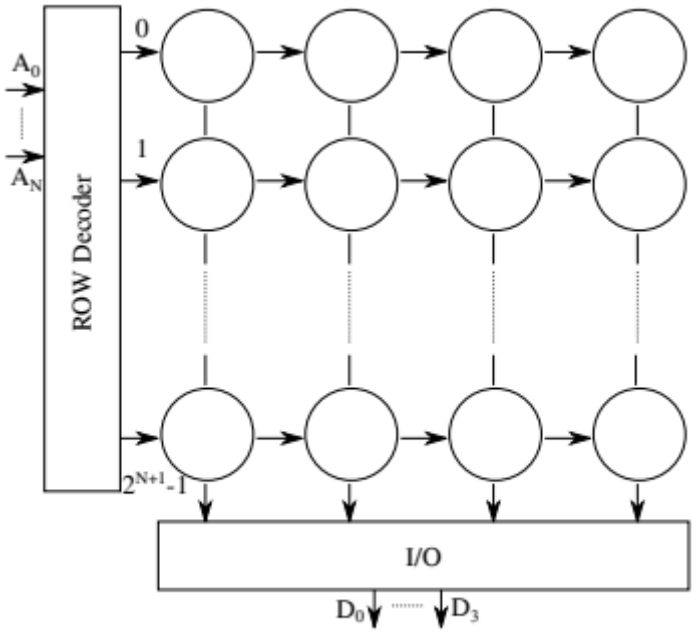
Občerstvování DRAM

- Proces čtení je destruktivní -> po něm občerstvení -> cca 10 ms
- V současnosti 3 metody refreshe
  - RAS-only
    - Nejjednodušší a nejvíce používaná metoda
    - Vykonání předstíraného cyklu čtení
    - Poskytne se pouze RAS signál a DRAM poskytne adresu řádku -> proběhne občerstvení celého řádku -> je nutné, aby DRAM poskytl adresy řádků přesně jak jdou po sobě
  - CAS before RAS refresh
    - Vlastní refresh logika s adresním počítadlem
    - CAS je udržován na nízké urovni po jistou dobu, než RAS klesna na nízkou uroven -> tím je pak refresh logika aktivovaná a vykoná automatický vnitřní refresh
  - Hidden refresh
    - Tento cyklus refreshe je schovaný za čtení, protože systém refreshe je kratší než čtení, takže proto to probíhá „skrytě“

Paměťové moduly

- Pamětové čipy -> DIP (Dual inline package) -> integrované obvody s vývody na obou stranách
- Pro snadnější instalaci jsou umístěné na modulu
- Dnes se používají kompaktní moduly SIMM a DIMM než jednotlivé čipy
- Pro dosažení požadované velikosti je potřeba instalovat potřebné množství čipů -> mohou být vloženy do soketů na motherboardě
- SIMM moduly
  - Single inline memory module
    - Na jedné nebo obou stranách čipy 30/72 piny -> normálně 72 pinů -> podpora 32bitový přesun mezi procesorem a pamětí
- DIMM moduly
  - Double inline memory module
    - 168 pinové DIMM -> vždy šířka 64 bitů
    - Jako DIMM se používají hlavně paměti SDRAM a DDR RAM
- RIMM moduly
  - Rambus inline memory module
    - 184 kontaktů

- Kapacity 64,128,546 Mb -> vede to k hodnotě 800MHz
- Podobně jako v případě DDR RAM probíhá přenos po obou stranách hodinového pulsu -> Intel zde používá jinou implementaci Direct RAMBus -> datová sběrnice 16bitů
- RAMBus byly instalovány na intel P4 kolem roku 2000



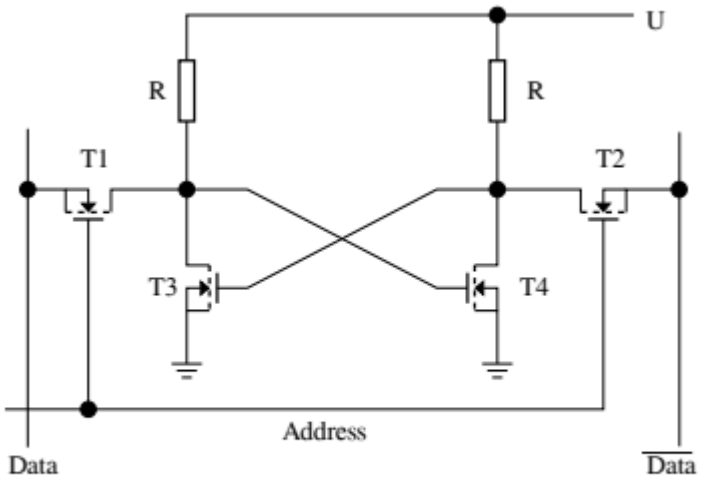
Obrázek 9: Organizace paměti SRAM

Statické paměti

- Ve statických pamětech jsou informace uložené stavem klopného obvodu -> realizace pomocí 4/6 tranztisorů -> obě varianty na obr 10 a 11
- Na obr 9 je vidět zjednodušená organizace paměti SRAM -> kolečko paměťová bunka -> organizace 2D mřížky -> jeden řádek tvoří slovo -> vodorovné čáry jsou pamětové vodiče, které vybírá signál dekodéru řádku (ROW) na základě vstupní adresy A0 až An -> svislé vodiče jsou datové voideče které přenáší uloženou informaci vybraného řádku do výstupního I/O bufferu
- Pamětový kontroler SRAM čipů jsou adresy řádků poskytovány jako jedna informace -> chybí zde adresní multiplexing -> nutno více pinů -> SRAM čipy jsou větší než DRAM čipy -> vnitřní adresování je jednodušší a SRAM jsou rychlejší než DRAM
- A to nejen při adresování, ale i čtení a zápisu -> není potřeba refresh -> klopný obvod může udržovat signál, jak dlouho chce
- SRAM jsou dražší a pojmou méně informací kvůli menší hustotě paměťových buněk (4x – 6x menší než u DRAM) -> díky tomu se SRAM používá pro rychlé cache paměti zatímco DRAM spíše pro relativně pomalé hlavní paměti RAMka

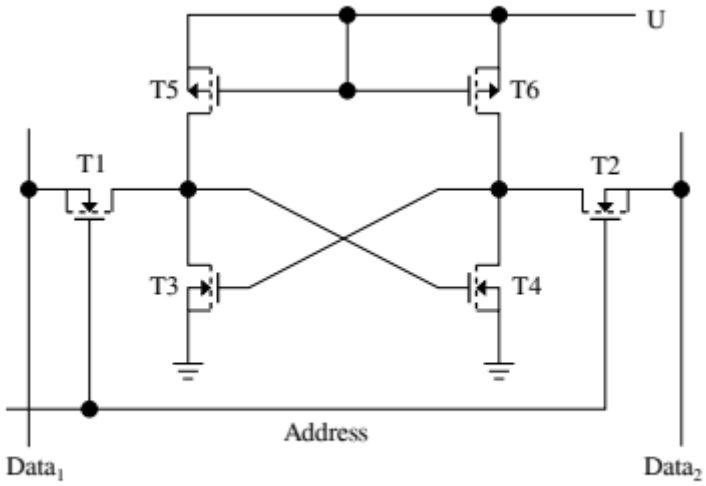
SRAM – static random access memory

Pamětová bunka SRAM



Obrázek 10: Paměťová buňka SRAM se 4 tranzistory

- V obou případech tranzistorů (4/6) se bunka SRAM skládá ze dvou NMOS přístupových tranzistorů T2 a T3 a klopného obvodu se dvěma NMOS paměťovými tranzistory T3 a T4 -> další dva elementy jsou odpory, nebo dva PMOS tranzistory T5 a T6, které vedle s tranzistory T3 a T4 tvoří CMOS dvojici
- V SRAM jsou tyto bunky uspořádané v mřížce (rows,cols) které se vybírají dekodéry řádku
- Báze (gate) přístupových tranzistorů T1 a T2 jsou připojeny na adresové vodiče a emitory (source) na páry datových vodičů DATA a negace DATA
- Paměti SRAM lze vyrábět pomocí technologie TTL, pak funguje tato bunka jako paměťová bunka technologie MOS, ale skládá se pouze ze dvou PNP tranzistorů a dvou odporů



Obrázek 11: Paměťová buňka SRAM se 6 tranzistory

Čtení a zápis dat

- SRAM nepoužívá adresní multiplexing -> adresa čipu je tím pádem přenesena jako jedna informace při zápisu i čtení z/do bunky se aktivuje dekodér řádku odpovídajícího adresového vodiče

- Při čtení se přístupové tranzistory T1 a T2 zapnou a propojí klopný obvod s datovými vodiči DATA a negace DATA -> pro stabilizaci vybere dekodér odpovídající sloupec (odpovídající datové vodiče DATA a negace DATA) a předá na výstupu data do I/O bufferu a tímto do vnějších obvodů
- Zápis probíhá opačným způsobem -> přes vstupní datový buffer -> zapasovaná data se zavedou na odpovídající čtecí zesilovač -> aktivuje dekodér řádku a zapne přístupový tranzistor T1 -> stejně jako se při čtení snaží předat na datový vodič DATA a negace DATA -> čtecí zesilovač je silnější než tranzistor T3 -> poskytne datovým vodičům signál, který odpovídá zapisovaným datům -> klopný obvod se přepne podle nově zapisovaných dat a nebo si udržuje svojí původní hodnotu v závislosti na to jestli jsou data stejné nebo se liší

Async Sync a PB SRAM

- Asynchronní SRAM
    - Od doby procesoru 386 a stále se nachází v některých L2 cache
    - Není synchronizovaná se systémovými hodinami -> proto musí CPU na data z paměti cache L2 čekat (né moc dlouho jako u DRAM)
- Synchronní burst SRAM
    - Podobná SDRAM -> synchronizovaná se systémovými hodinami
- Pipeline Burst SRAM
    - Zřetězená dávková SRAM
    - Díky dávkové technologii jde požadavky zřetězit a mít jejich výsledky téměř instantně
    - PB SRAM – Jedná se o zlepšení SRAM, je zde sice drobná desynchronizace -> je navržená pro komunikaci se sběrnicemi

Pamět s trvalým obsahem

- Schémata těchto technologií je ve studijním materiálu k technologii výroby číslicových systémů
- Nevýhoda předchozích pamětí je závislost na napájení
- SRAM a DRAM nejsou vhodné pro startovací sekvenci PC -> tady se používají ROM čipy -> data jsou uloženy energeticky nezávislým způsobem
- Ve vývoji se vystřídaly pasivní i aktivní elektro prvky
    - Byly použity odpory, indukčnost, feritová jádra, kondenzátory, diody, tranzistory unipolární i bipolární
- Dobré například pro uchování biosu

ROM

- Read only memory -> pouze pro čtení
- Bunka paměti je elektický odpor nebo pojistkou -> některé z nich jsou elektricky přepálí -> neporušené prvky pak vedou proud a je v nich minimální napětí -> logická 0
- Přepálené prvky proud nevedou a je v nich maximální napětí -> logická 1
- Informace do nich zapisuje výrobce -> doba pamatování není ohraničena

PROM

- U programovatelných ROM čipů vypaluje data do paměti uživatel pomocí programátoru -> pomocí elektrického pulzu
- Jedna z metod je přepálení pojistky mezi adresovým a datovým vodičem -> pojistky jsou z niklu a chromu nebo křemíku -> stejně jako u ROM není po naprogramování možný zápis

- Další realizace paměti PROM -> bipolární multiemitorový tranzistor na každý adresový vodič -> pokud se má z paměti číst -> do adresového vodiče se přivede 1 -> multiemitorový se otevře ve směru kolektor-emitor začne procházet proud -> pokud nebyla pojistka přepálena, tak se otevře transistor který je připojený jako invertor a výstup se přepíše na 0 -> pokud je pojistka přepálená, tak se tranzistor neotevře a zůstane logická 1

EPROM

- Erasable programmable read only memory
- Je možné opakovaně zapisovat -> informace se uchovává pomocí elektrického náboje který je kvalitně izolovaný, takže udrží svůj náboj i po odpojení od elektřiny
- K programování je potřeba pulz trvající 50ms o napětí 5V+ (u některých typů až 12V)
- Programuje se speciálním programátorem
- Lze vymazat ultrafialovým zářením a poté zapsat nová data
- EPROM lze poznat podle okénka na pouzdře kudy vstupuje vymazávací ultrafialové záření
- Doba pamatování dat je cca 10-20 let
- Používá se kapacita izolovaného hradla tranzistoru MOS

EEPROM

- Odstranění okénka pro UV světlo -> bylo to drahé a složíte -> chtěli udělat něco, co půjde vymazat stejně jako to bylo naprogramované
- To je přivedlo k elektricky vymazatelné PROM
- Programování úplně stejně jako u EPROM -> Pro vymazání se obrátí polarita pulzu -> počet zápisů a mazání je ohraničen
- Doba omezena zase na 10-20 let

Flash paměti

- V posledních letech se používá typ EEPROM paměti, schopný uchovat si data i po odpojení elektřiny -> náhrada za diskety a pevné disky -> Flash paměť
- Výhoda rychlého programování přímo v PC -> délka uchované informace 10–100 let
- Struktura paměťových buněk stejná jako EEPROM
- Programovací a mazací puls 10 nanosekund -> i menší napětí -> vymazání je velmi lehké a rychlé
- Bez problémů 10 000 programovacích a mazacích cyklů -> dle typu paměti
- Adresová buffer přijímá adresy -> přesun do dekodéru řádku a sloupce
- Nemají stejně jako SRAM čipy multiplexing
- Dekodéry vyberou jeden adresní vodič a jeden nebo více datových vodičů tak jako v běžném čipu -> výsledná data jsou předávany na datový buffer a v případě zápisu jsou zapsané do adresované bunky tímto bufferem přes I/O bránu
- Proces zápisu je trochu složitější -> je možné zapsat 0, ale není možné zapsat 1 normálním způsobem, k tomu je potřeba zkopírovat celý sektor do RAM paměti a vymazat ho z flash paměti -> v RAM paměti se 1 zapíše do daného řádku a ten se potom celý zapíše zpět do flash paměti na jeho původní místo

Další typy pamětí

Video paměti

Video RAM (VRAM)

- Běžná DRAM nemá dostatečně široké přenosové pásmo pro udržení rozlišení a barevné hloubky -> kvůli tomu bylo vytvořeno VRAM
- Dvouportová pamět -> má dva přístupové porty pro paměťové buňky -> jeden pro refreshování obrazu a druhý pro změny dat, které se mají zobrazovat -> dva porty -> ztrojnásobené kapacity přenosového pásma a v důsledku toho vyšší grafický výkon

WRAM

- Také dvouportové -> širší přenosové pásmo (cca o 25% oproti VRAM) + několik grafických funkcí pro tvůrce
- Systém dvojitého vyrovnávání dat (double buffering) -> několika násobně rychlejší než u VRAM -> vyšší obnovovací frekvence zobrazení

SGRAM

- Synchronní grafická pamět -> něco jako SDRAM -> SGRAM je optimalizovaná pro maximální přenos dat

FIFO paměti

- Realizace přímo v procesoru nebo jako stavební členy s různou organizací
- Lze je rozdělit na typy:
    - Bez přesouvání obsahu
        - Zápis a čtení z fronty se řídí dvojic registru
        - Čte se podle obsahu registru začátku fronty -> konce registru fronty
        - Řídící obvody vytváří dva důležité stavové signály -> plná a prázdná fronta (přetečení podtečení)
    - S přesouváním obsahu
        - Čtení a zápis stejně složité -> mají jeden přídavný registr -> fronta s probubláváním asynchronně posouvá registr po zápisu až do posledního volného místa, při přečtení se z fronty odstraní
        - Princip vyžaduje, aby u každého paměťového místa existoval indikátor obsazenosti (klopný obvod)

Cache paměti

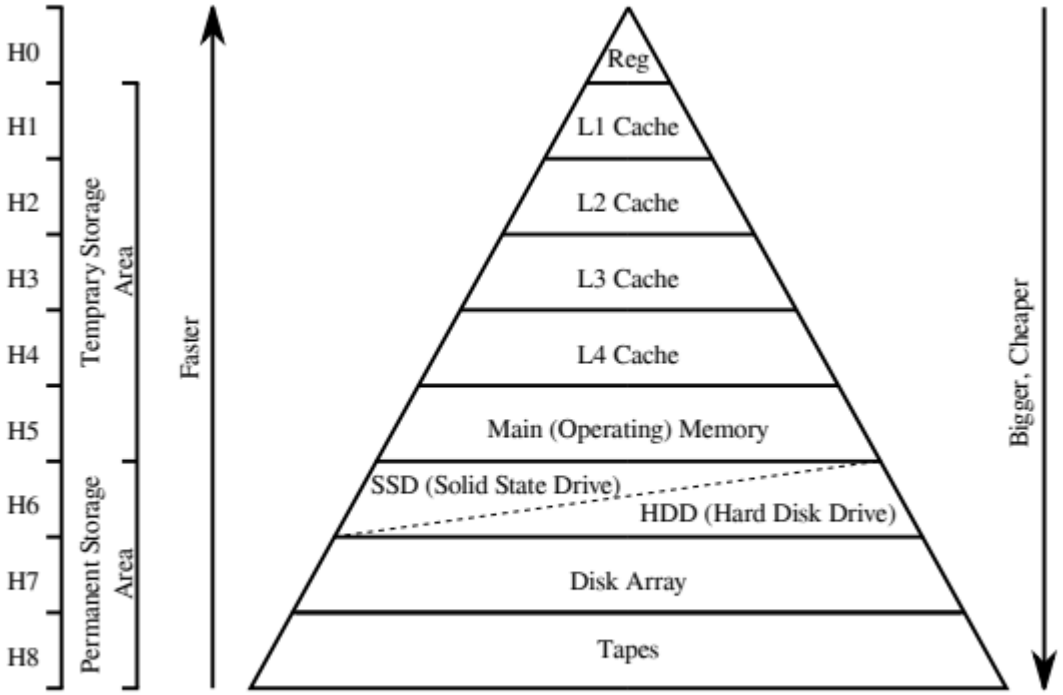
- Mezisklad pro informace
- Používají ji rychle komponenty, aby nemuseli čekat -> cache si už data nečetla

L1 cache

- Přímo na procesoru -> zásobování procesoru daty ze sběrnice
- Cache se načte předem data a čeká, dokud je procesor nevyužije
- Cache má svůj řadič, který předpovídá, jaké data bude procesor potřebovat a ty do sebe načte -> cache je rychlejší než sběrnice -> je to výhodné -> nemusí se čekat

L2 cache

- Umístěna mezi mikroprocesorem a operační pamětí -> všechny data co projdou, tak projdou i cache pamětí, pokud je bude procesor znovu potřebovat, může se dotázat cache paměti
- Zase se cache snaží pomocí řadiče předpovídat co bude procesor potřebovat
- Pracuje také jako víceportová pamět, která může číst i zapisovat
- Cache používají 3 režimy
    - Write through (zápis krze cache, přímý zápis)
        - Nejstarší a nejpomalejší -> typický pro procesory řad 486
        - Data jsou zapisovány do cache a zároveň do operační paměti
        - Při čtení porovná řadič cache pořadované data v daty na požadované adrese, pokud je najde, tak je přečte
    - Write back (opožděný zápis)
        - Novější a rychlejší způsob pro Pentia a rychlejší 486
        - Data jsou zapisovány pouze do cache a pouze při odstranění se zapíšou do operační paměti -> než se tyto data do operační paměti dostanou, mohou několikrát změnit svoji hodnotu -> šetří čas
        -
    - Pipeline burst
        - Nejnovější a nejrychlejší systém práce -> dnes běžně používaný
        - Provede více operací zřetězeně -> pokud čte z nějaké adresy, přečte zároveň i informace následujících adres (což by stejně asi dělal za chvíli)
        - Přistupová doba mezi 9-15 nanosekund



Obrázek 12: Paměťová hierarchie

Hierarchie pamětí v počítači

- Každá technologie má svůj důvod a praktické použití
- Většinou jde o kompromis ceny a rychlosti -> kdyby existovala univerzální pamět co by byla ve všem nejlepší -> používala by se jen ona
- Příklad pamětové hierarchie na obr 12
- Ukázka je pouze demonstrativní, vždycky se může lišit podle typu počítače
- Obr 12 by reprezentoval něco jako výkonný server

- Paměti jsou v počítači uspořádány podle rychlosti, ceny a kapacity. V obrázku 12 je to znázorněno šipkami na levé a pravé straně pyramidy. Nejrychlejší a nejdražší jsou paměti v CPU, tedy registry na vrcholu pyramidy.
- Každá vyšší úroveň paměti (v pyramidě) tvoří vyrovnávací paměť (cache) pro úroveň pod sebou.

  Např. hlavní paměť tvoří vyrovnávací paměť pro pevný disk a data hlavní paměti jsou částečně uložena v Cache L4, atd.
- Paměti se dělí na oblast pro ukládání dočasných dat a dat trvalých.
- Dříve se používalo dělení pamětí na vnitřní a vnější paměti a dodnes je možno v některých textech tuto informaci nalézt. Za dělící rovinu mezi těmito dvěma typy pamětí se považoval přechod mezi H6 a H5.

  Dnes je vhodnější dělení na paměti čistě polovodičové a paměti s mechanickými součástkami. Toto dělení je naznačeno čárkovanou čarou v úrovni H6. Všechny paměti nad touto čarou jsou čistě polovodičové a paměti pod čarou obsahují nějaké mechanické díly. Toto dělení je velmi důležité z hlediska rychlosti paměti. Všechny paměti s mechanickými díly jsou i o několik řádů pomalejší, než paměti polovodičové.

Dále je možno charakterizovat jednotlivé použité technologie, jejich parametry a kapacity v paměťové hierarchii:

H0 - registry procesoru. Rychlost odpovídá rychlosti CPU a pohybuje se v jednotkách ns, nebo desetinách ns. Kapacita se může pohybovat ve stovkách bytů na jádro, takže celkově i v jednotkách kB.

H1 - Cache L1, SRAM, rychlost odpovídá vnitřním sběrnicím CPU, obvykle v nižších jednotkách ns. Kapacita desítky až stovky kB na jádro.

H2 - Cache L2, SRAM, rychlost obvykle od jednotek do 10 ns. Kapacita stovky kB na jádro.

H3 - Cache L3, SRAM, rychlost v nižších desítkách ns, kapacita v jednotkách až desítkách MB.

H4 - Cache L4, SRAM, rychlost v desítkách ns, kapacita desítky až stovky MB.

H5 - Hlavní paměť, někdy nazývaná operační paměť, DRAM, rychlost v desítkách ns, kapacita jednotky až stovky GB,

H6 - SSD disky, Flash, rychlost v desetinách či jednotkách ms, kapacita stovky GB až jednotky TB. Pevné disky, rychlost v jednotkách až desítkách ms, kapacita v jednotkách TB.

H7 - Disková pole, parametry odvozeny od pevných disků, kapacita od desítek TB až po jednotky PB.

H8 - Páskové jednotky, rychlosti dle použitého řešeni od desítek minut až po hodiny.

Chyby pamětí, jejich detekce a oprava

Chyby pamětí

- Elektrické zařízení -> může vracet nesprávné hodnoty -> DRAM jsou díky své charakteristice schopné často vracet špatné hodnoty -> DRAM je kvůli refreshování méně spolehlivá než SRAM
- Tvrdé (opakující) chyby -> část hardware je rozbitá a bude neustále vracet špatnou hodnotu -> lehké takové chyby rozpoznat a opravit, protože jsou konzistentní a opakují se
- Přechodové (měkké) chyby -> Nastávají, když se přečte bit se špatnou hodnotou, ale poté funguje správně -> tyto problémy se špatně lokalizují, chápou a i opravují + jsou časté -> Jediná oprava nebo něco takového jsou funkce pro jejich detekci a opravné metody -> mohou detekovat chyby ve více než jednom bitu a opravit je

Parita

- Dělí se na fyzickou a logickou
  - Fyzická
    - Paritní bity jsou posunuty paměťovému kontroléru během procesu zápisu dat jsou uloženy v paměťovém modulu -> během čtení paměťový modul předá na výstup informaci o uložené paritě
  - Logická
    - Kontroler generuje paritní bity, ty jsou předány během zápisu, pamětový modul je však neuloží -> tím šetří bity cca 10% (4 až 36) -> je levnější
- Pamětové moduly mohou anebo nemusí obsahovat paritu -> většinou neobsahují -> obsahuje jeden bit paměti pro každý jeden bit uložených dat
- Paměti s paritou navíc přidávají jeden bit navíc k 8bitům v bytu -> používá se pro detekci chyby
- Paměti s paritou mohou použít kontrolu parity -> jednoduchá forma detekce chyb
- Paměti bez parity nic takového nemají

Kontrola parity

- Detekce jednobitových chyb v systémové paměti
- Když se byte zapíše do paměti -> logický obvod (generátor/kontrolér) vyhodnotí zda je byte sudý nebo lichý podle jedniček
- Pokud je sudý -> 9. paritní bit bude nastaven na 1 jinak bude 0 -> výsledek je ten, že v jakém koliv případě bude byte lichý, protože jej ze sudého přesuneme na lichý -> lichá parita

- Během čtení dat se paritní obvod kouká, jestli není nějaký byte sudý, jestli ano, musela tam být chyba -> obvod vygeneruje nemaskovatelné přerušení (NMI) -> okamžité zastavení procesoru
- Tato ochrana je pouze na jeden bit, když by se nám poškodily dva bity a prohodily svoje hodnoty tak to nezjistíme, protože si nepamatujeme jednotlivé bity, jenom celkovou hodnotu jedniček v bytu

ECC paměti

- Kontrola parity kontroluje jenom jeden bit a navíc nic neopravuje -> vynalezen protokol pro detekci a opravu chyb ECC (Error correction code)
- Detekuje jedno i více bitové chyby + je schopný jednobitové chyby při čtení opravit
- ECC používá speciální algoritmus pro kódování informací bloků -> obsahuje detaily pro obnovení chyby jednoho bitu
- Na rozdíl od parity, kde jeden bit kontroluje 8bitů ECC používá skupiny bitů -> 7 bitů pro ochranu 32 bitů anebo 8 pro 64bitů
- ECC může detekovat chyby až 4 bitů, ale neumí je opravit -> tyto chyby řeší stejně jako parita, prostě pošle NMI signál na přerušení, tyto chyby na více bitech jsou však vzácné
- Na rozdíl od parity však mírně zpomalí systémové operace -> je komplikovanější a trvá to tím pádem déle, protože se musí daná chyba opravit -> kvůli tomuto čekání je potřeba vložit wait state během čtení paměti -> snížení výkonu o 2-3%

Externí paměti PC – disky

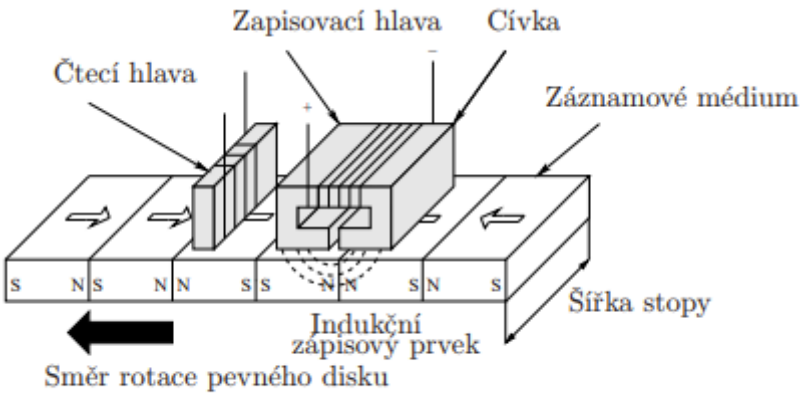
Historie

- Pevné disky
  - 1953 – první pevný disk – RAMAC
  - 1973 – první uzavřená disková jednotka, přímo integrovaná i čtecí hlava – disky Winchester
  - 1990 – Magnetorezistivní hlava
  - 1997 Giant Magnetorezistiv hlava
- Disková mechanika
  - 1971 první pružný disk
  - 1976 první disketová mechanika pro diskety o velikosti 5,25“
  - 1983 První 3,5“ disketa s kapacitou 875KB
- CD-ROM
  - 1982 specifikace standardu Red Book
  - 1988 první CD-R mechanika
  - 1994 standard yellow book (možnost ukládat počítačová data)
  - První CD-RW mechanika
- DVD
  - 1997 první DVD přehrávače a disky

Typy pamětí

- Magnetické paměti
  - Pevný disk
  - Disketová mechanika
- Optické paměti
  - CD
  - DVD
- Metrooptické paměti

Paměti magnetické



Obrázek 1: Princip magnetického zápisu

- Externí paměti nejčastěji pracují s magnetickým záznamem

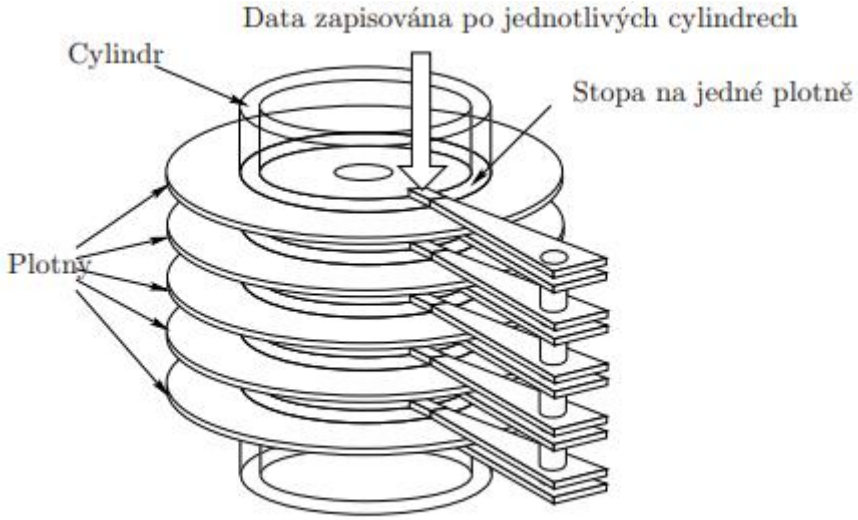
- Záznamové médium má tvar kruhové desky (disk, disketa) nebo dlouhé pásky (magnetická páska) -> je pokryto aktivní magnetickou vrstvou a pohybuje se konstantní rychlostí -> v bodu dotyku média je štěrbina magnetického obvodu -> jádro přerušené štěrbinou na němž je cívka -> magnetický tok který prochází jádne při průchodu elektrického proudu cívkou se v místě štěrbiny rozptyluje do nejbližšího okolí a uzavírá se přes vzduchovou mezeru -> toto zasahuje i aktivní vrstvy záznamového materiálu, kterou magnetizují
- Pokud se mění směr elektrického proudu v cívce -> mění se i směr magnetického toku jádrem i štěrbinou -> mění se smysl magnetizace vrstvy
- Magnetická reverzace -> místa mezi různými směry magnetického zápisu -> jejich přítomnost na médiu ukazuje zapsanou informaci
- Megnetické reverze způsobují na cívce napětové impulsy -> toto zpracovávají elektronické zesilovače

Pevný disk

- Uzavřená jednotka v počítači -> trvalé uskladnění dat -> pouzdro chrání před nečistotami atd…
- Obsahuje -> pevné plotny diskového tvaru -> slitina hliníku nebo skla -> plotny nejsou ohebné -> proto pevný disk -> plotny nejde vyjmout
- Části pevného disku
    - Plotny disku
    - Hlavy pro čtení a zápis
    - Pohon hlav
    - Vzduchové filtry
    - Pohon ploten disku
    - Řídící deska
    - Kabely a konektory

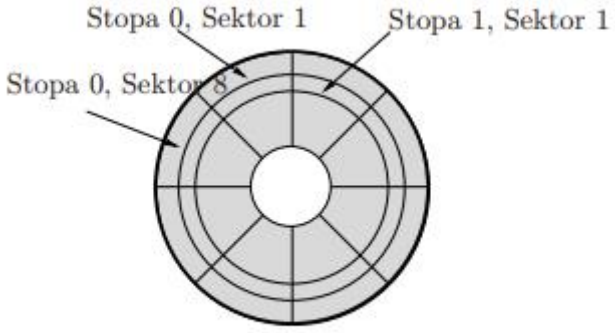
Geometrie disku

- Uspořádání prostoru na disku -> počet hlav, cylindrů a stop
- Data jsou po bajtech -> skupiny po 512 (sektory) -> sektor je nejmenší jednotka dat na kterou jde zapsat nebo číst -> sektory jsou seskupeny do stop -> stopy jsou ve skupinách neboli cylindrech nebo válce
- Předpoklad je, že disk má nejméně dva povrchy -> systém adresuje sektory na pevném disku pomocí prostorové matice cylindrů, hlav a sektorů
    - Stopy – Každá strana každé plotny je rozdělená na soustředné stopy (kružnice) -> Jejich počet neustále narůstá (více než 16 000 stop) -> povrchů i hlav je několik, takže stačí při jedné poloze hlavy mít jednu stopu, takže stačí přepínat hlavy
    - Cylindry – Více ploten (disků) umístěných nad sebou otáčející se stejnou rychlostí -> každý plotna má dvě strany kam jde ukládat -> Souhrn stop v jedné poloze hlav se nazývá cylindr – obr 2



Obrázek 2: Cylindry

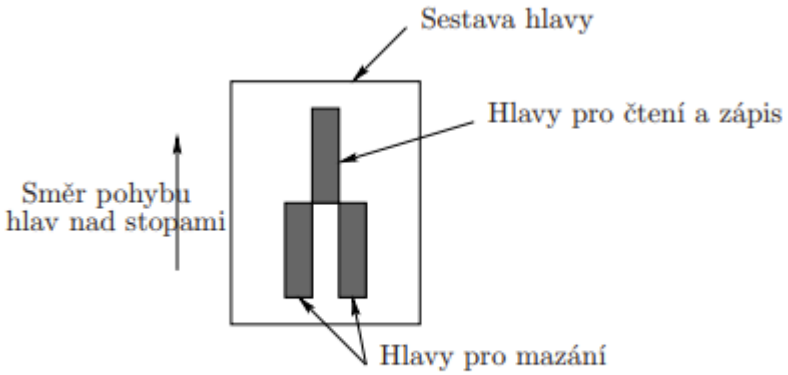
- Počet stop na jednom povrchu je totožný s počtem cylindrů -> neuvádí se počet stop, ale cylindrů
- Sektory
    - Stopa je příliš velká pro efektivní ukládání dat (min. 100kB) -> rozděluje se na několik očíslovaných částí -> sektory -> výseče na plotně – obr 3
    - U pevných disků bývá 900+ sektorů -> nejmenší adresovatelná jednotka -> sektory se číslují na rozdíl od hlav a cylindrů od 1
    - Na začátku sektoru je hlavička -> identifikace začátku sektoru -> číslo
    - Zakončení sektoru -> ukládání kontrolního součtu -> kontrola integrity uložených dat
    - Při formátování se vytváří před a za sektorem identifikační část -> řadič používá pro číslování sektorů -> určení začátku a konce -> sektory jsou oddělené mezisektorovými mezerami (není možné zde uložit data)
    - Čtení ze sektoru -> hlava na požadovanou stopu -> čekání až se disk natočí tak, že je sektor pod hlavou -> probíhá čtení -> nejdelší je přemístění hlavy -> nejrychleji se čtou soubory co jsou na jedné stopě a jsou nad sebou v jednom cylindru



Obrázek 3: Stopy a sektory

Disketové mechaniky

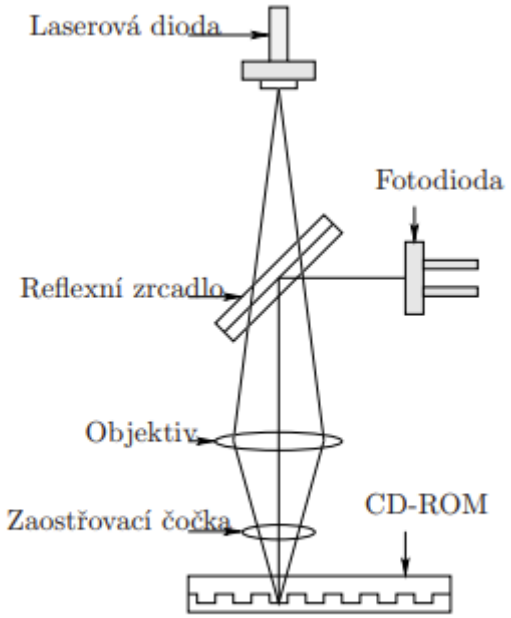
- Disketa se otáčí 300 ot/min -> nad disketou jsou hlavy v max dráze 2,54cm -> směr dovnitř ke středu nebo od středu ven -> během tohoto můžou hlavy zapsat 80 stop na obě strany diskety -> z tohoto důvodu jsou občas stopy nazývané cylindry -> cylindr je stopa co se zrovna nachází pod hlavami na obou stranách
- Záznam dat je prováden metodou tunelového mazání -> zapsání určité šířky stopy s následným mazáním okraje (té samé stopy) z důvodu problémů interference sousedními stopami, je šířka stopy 115 nanometru



Obrázek 4: Sestava hlavy disketové mechaniky

-
- Rozhraní používané všemi typy mechanik se nazývá Shugart Associates SA-400 založené na čipu NEC 765 nebo na kompatibilních čipech
- Části disketové mechaniky
    - Hlavy pro čtení a zápis
    - Pohon hlav a mechaniky
    - Řídíc deska
    - Řadič
    - Konektory
- Typy disketových mechanik
    - Mechanika 3,5“ o kapacitě 720KB (DD – double density)
    - Mechanika 3,5“ o kapacitě 1,44MB (HD – high density)
    - Mechanika 3,5“ o kapacitě 2,88MB (ED – extra high density)

Optické paměti



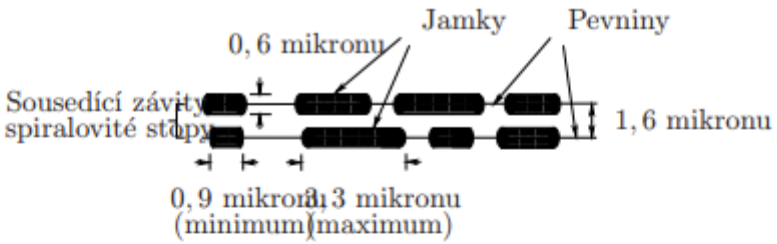
Obrázek 5: Princip optického záznamu

Princip ukládání dat

- Optické paměti CD-ROM -> rozšířené -> vytvořené z polykarbonátu
- Strana kde je něco uložené je reflexivní a má ochrannou vstvu na které je popis CD
- Záznam je pomocí pitů a polí -> jednička je přechod mezi pitem a polem -> něco jako magnetická reverzace
- Při čtení projde celé medium dvakrát nejdříve ze strany kde nejsou pity
- Vypálení pitů do vrstvy -> při zápisu silnější dioda -> nevratná změna -> mění se koeficient reflexe

Jamky a pevniny

- Čtení z CDčka je založeno na rozdílném odrazu laseru od pitů a pevnin -> jedno CD má cca bilion pitů -> jamky na CD mají 0,125 x 0,6 nanometru a jsou proměnné v délce cca 0,9-3,3 nanometru



Obrázek 6: Jamky a pevniny u CD médií

- **DVD** - záznam je stejně jako u CD disků tvořen stopou s jamkami a pevninami. Liší se pouze v rozměrech, které jsou menší z důvodu dosažení větší kapacity (tab. 1).

	DVD	**CD**
Délka jamky	0,4 - 1,9μm	0,9 - 3,3μm
Hloubka jamky	0,105μm	0,125μm
Šířka jamky	0,4μm	0,6μm
Vzdálenost sousedních stop	0,74μm	1,6μm
Plocha média pro záznam dat	8,759mm^2	8,605mm^2
Kapacita	4,7 - 17GB	650 - 700MB

Tabulka 1: Rozdíly mezi CD a DVD

- Stopy a sektory
  - CD-ROM
    - Data jsou zaznaménány do stopy -> spirála -> jednotlivé závity spirály jsou od sebe 1,6 nanometu
    - Jedno CD s kapacitou 74 minut (700MB) obsahuje 22188 závitů a celková délka stopy je 5,77km
    - Stopa je rozdělená na sektory -> 75 sektorů za sekundu -> sektory se dělí na 98 rámců -> každý z nich obsahuje 33 bytů (24data, 1 subkod, 8 kontrolní kod a oprava chyb)
    - Kapacita jednoho sektoru je 3234 bytů
  - DVD
    - **DVD** - stopa ve formě spirály, má celkovou délku 11,84km. Závity jsou od sebe vzdáleny 0,74μm, výsledná hustota stop je 1351 závitů na milimetr. Celkem je v jedné záznamové vrstvě 49324 závitů.

      Stopa je tvořena sektory, z nichž každý obsahuje 2048 bytů dat. Při zápisu jsou nejprve data zformátována do datových rámců o velikosti 2064 bytů. Z toho je 2048 bytů pro data, 4 byty obsahují identifikační informace, 2 byty obsahují informace pro detekci chyb v identifikačních informacích (*ID Error Detection - IED*), 6 bytů dat pro ochranu autorských práv, a 4 byty obsahují kód pro detekci a opravu chyb pro celý rámec.

CD ROM

- Compact Disc Read Only Memory
- Medium pro čtení, ale i samotná čtečka k práci s tímto médiem
- Další formáty jsou CD-R (recordable) a DC RW (rewritable)
- Lepší než diskety, horší než dnešní nové disky
- Laser snímá vzor z povrchu -> umístěn rovnoběžně s povrchem disku -> paprsek je na disk odražen zrcadlem přes dvě čočky -> fotodetektor měří intenzitu odraženého světla
- Mechanismus je od disku asi 1 mm, laser nemůže disk poškodit, pokud se něco neposere
- Rychlost otáření 540ot/min ve středu a 212 ot/min u krajů -> díky tomuto dosáhneme konstantní rychlosti čtění
- Čtení je pomocí infračervené diody, která přes pohyblivé zrcátko svítí na stopu kde se čtou data
- Světlo se láme přes zdrcátka a čočky, poslední část je fotosenzitivní a dovede říct co se stalo a převede tento světelný impuls na elektrický -> elektrické impulsy jsou dekodovány procesorem a předány do PC ve formě dat

Části mechaniky CD ROM

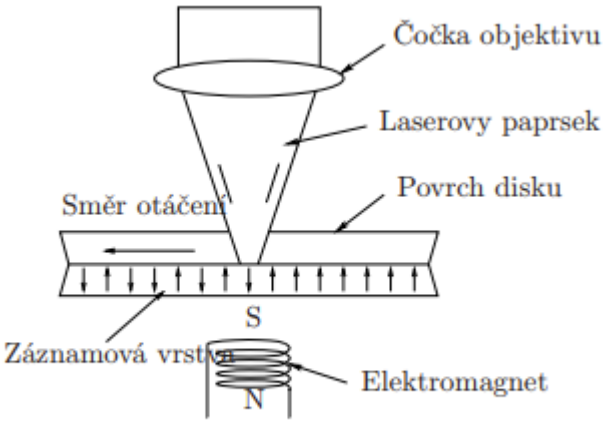
- Laserová (optická) hlava
- Fotodetektor
- Servomechanismus
- Ovládací čip
- Vyrovnávací pamět
- Elektronika pro dekodování signálů a řízení procesů

DVD

- Digital versatile disc
- V podstatě jde o vysokokapacitní CDčko -> DVD je kompatibilní s CD -> v DVD mechanice můžeme pustit CDčka
- Zvýšená kapacita, frekvence světelného paprsku, jiné ošetření dat a kodování dat
- Používá se laser ve viditelném spektru -> červený o vlnové délce 650nm
- DVD může mít problémy s kompatibilitou při čtení médií CD-R CD-RW -> příčinou je barvivo na záznamové vrstvě -> Odrazivost barviv závisí na vlnové délce atd…
- Kapacita DVD je od 4,7GB u jednostranných jednovrstvých médií a do 17,1 GB u dvouvrstvých oboustranných médií
- Typy mechanik umožnující zápis
    - Mechaniky DVD-RAM
    - Mechaniky DVD-R
    - Mechaniky DVD-RW
    - Mechaniky DVD+RW

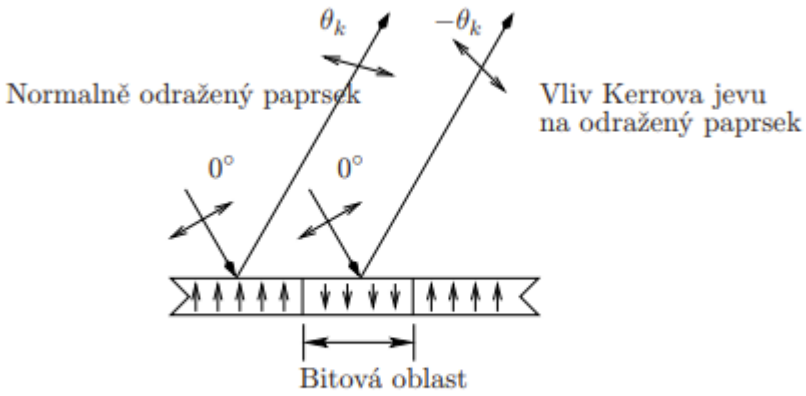
Paměti magneto-optické

Princip ukládání dat



Obrázek 8: Záznam dat u magnetooptických disků

- Kombinace optického a magnetického principu
- Termomagnetooptický disk
- Využívá změny magnetické orientace -> působení tepla a elektromagnetického pole -> proces je vratný
- Ozáření místa výkonným laserovým svazkem -> Základ MO disku je magnetická vrstva, Curieův bod leží v okolí 180 stupnů -> toto je bod, kdy se látka mění z chování feromagnetického na paramagnetické -> faktu že paramagnetické látce můžeme změnit stav jednodušeji než feromagnetické tady využíváme pro zápis ->Laserový paprsek se zaostří do stopy o průměru menší než jeden mikrometr a dioda se přepne do výkonného režimu -> vlivem teploty dosáhne bodu Curierova, tímto přemagnetuje část do požadovaného směru -> pokud dojde k rychlému ochlazení, stav magnetické vrstvy se vrátí zpátky do feromagnetické -> tuto situaci ukazuje obr 8
- Čtení dat je založeno na Kerrově jevu -> rovina polarizovaného světla se otočí po směru hodinových ručiček nebo proti smeru v závislosti na magnetické orientaci povrchu -> rychlost čtení je podobná jako u magnetických disků



Obrázek 9: Kerrův efekt

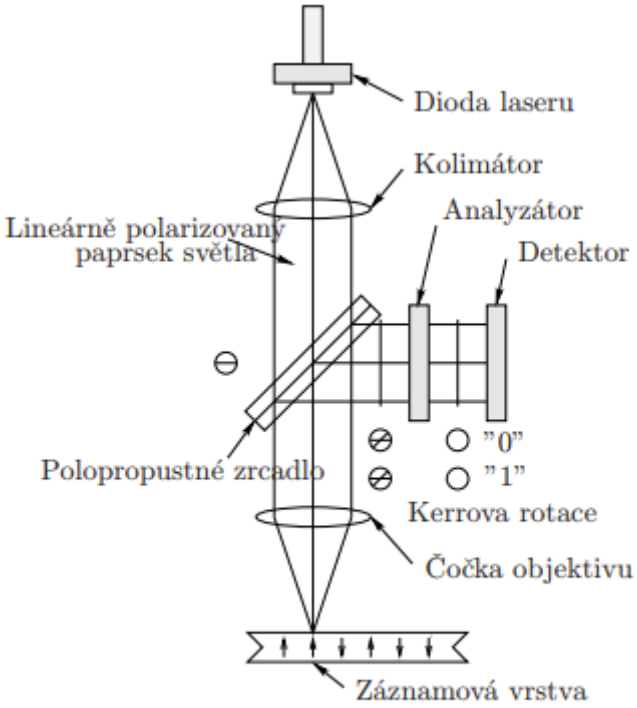
Magnetooptické disky

- 2 standartní velikosti -> 3,5“ a 5,25“ -> 1,3GB a 5,2GB

- Kdysi šlo ukládat pouze jednorázově -> nešlo přemazat (WORM)
- MO mechaniky nemají pevně zabudované médium -> vkládá se podobně jako disketa
- Magnetické prvky jsou na jedné straně disku a laserové na druhé straně
- Lze zapisovat jen na jednu stranu, pokud chceme použít obě strany, musíme médium vyndat a dát jej obráceně
- Jsou v plastovém krytu, aby se nepoškodily -> ta normálních teplot jsou stabilní a archivace je možná cca po dobu 30 let

Geometrie Disku

- Točí se konstantně -> spirála -> stopa = 25 sektorů po 512B nebo 1024B



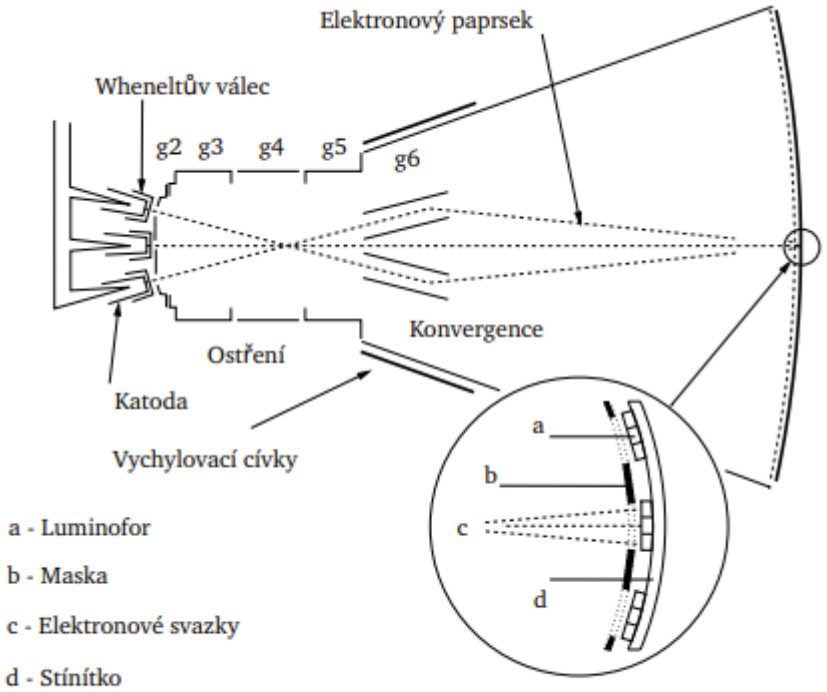
Obrázek 10: Čtení dat u magnetooptických disků

ZOBRAZOVACÍ JEDNOTKY

- Typy monitorů
  - CRT
  - LCD
  - Plazmové
  - OLED
  - E-Ink
-

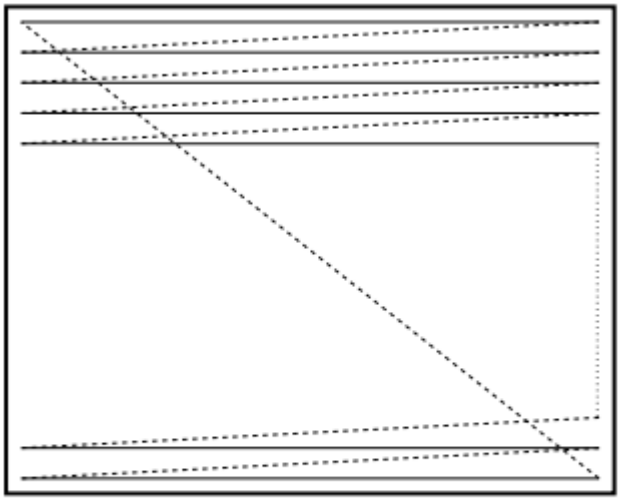
CRT

- Vzduchoprázdná skleněná banka -> přední část je luminiscenční látka
- Proces:
  - Grafická karta -> VGA port -> DAC převodník -> DAC je na specializovaném čipu kde je i RGB -> převedení do analogové tabulky kde máme napětí potřebné pro každou složku (RGB)
  - DAC přesněji obsahuje tři převodníky na každou jednu ze tří barev
  - Elektronové dělo, které je na konci katodové trubice -> elektronové svazky jsou emitovány z nepřímo žhavé katody, který má na svém povrchu nanesenou emisní vrstvu -> To po zahřátí vystřeluje vysokou rychlostí proudy elektronů pro jednu ze tří barev -> základní vlastnost je záporný náboj -> tato vlastnost je využívána k nasměrování částic -> toto projde mřížkou která pustí jenom požadované množství elektronů a tím řídí jejich intenzitu



Obrázek 1: CRT displej

-



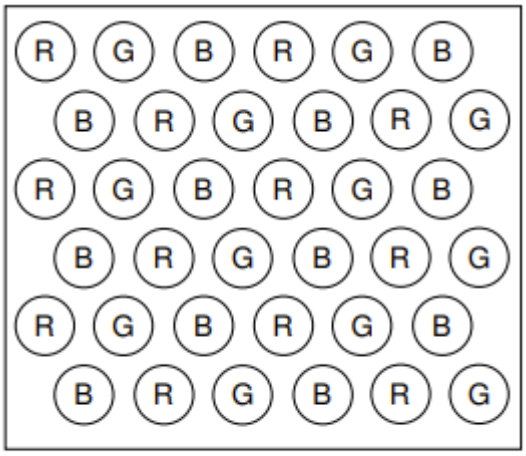
Obrázek 2: Princip řádkování

-
- Elektronové svazky procházejí Wheneltovým válcem, který má oproti katodě záporný potenciál -> podle napětí řídí intenzitu svazků
- Anoda s vysokým napětím je na konci horním okraji trubice -> kladně nabitá anoda vytahuje elektrony z elektronového děla -> tyto elektrony se zde však nikdy nedostanou kvůli odchylovacím cívkám
- Po průchodu wheneltovým válcem procházejí elektronové svazky přes jednotlivé mřížky g2 – g6, které mají vzhledem ke katodě kladný potenciál -> díky tomu jsou elektrony přitahovány -> potenciál je nejmenší v g2 a největší v g6 -> toto má dotáhnout svazky až na stínítko obrazovky
- Speciální funkci má mřížka g3 (ostření) která má za úkol zaostřovat svazky a mřížka g6 (konvergence) od které se svazky postupně sbíhají -> k jejich setkání dojde u masky obrazovky kde se překříží a dopadnou na své luminofory
- Svazky prochází kolem vychylovacích cívek, aby ohnul paprsek ve vertikálním a horizontálním směru a tím jej zaměřil na určitý bod na obrazovce
- Paprsek elektronů začne vlevo nahoře a skončí vpravo dole -> rastrování/řádkování (obr2)
- Až dokončí tento cyklus, plynule se vrátí doleva nahoru a začne nový cyklus, celou cestu paprsku po obrazovce určujeme termínem pole
- Signály zasílané do vychylovací cívky určují? Rozlišení, počet barevných bodů svisle a vodorovně a refresh rate
- Další problém je, že elektrony se odpuzují, proto by mohlo dojít k rozmazání obrazu, proto je zde mřížka/stínítko do které je kyselinou vypáleny dírky, tento materiál se nesmí roztahovat kvůli teplotě, protože vevnitř je to docela heat jak hovado -> toto by mohlo způsobit, že svazky nedopadnou, kam mají a zkreslily by se barvy
- Maska je mírně zakulacená, aby předvídala pohyb a roztahování -> kvůli tomuto zakulacení je zakulacené i sklo
- Barva záleží na tom, jak uděláme kombinaci intenzit

Invar

- Jednotlivé otvory v masce jsou kruhové a jsou uspořádány do trojúhelníku (stejně tak i luminofory na stínítku) viz obrázek 3
- Nevýhodou je velká plocha kovové masky -> větší roztažení

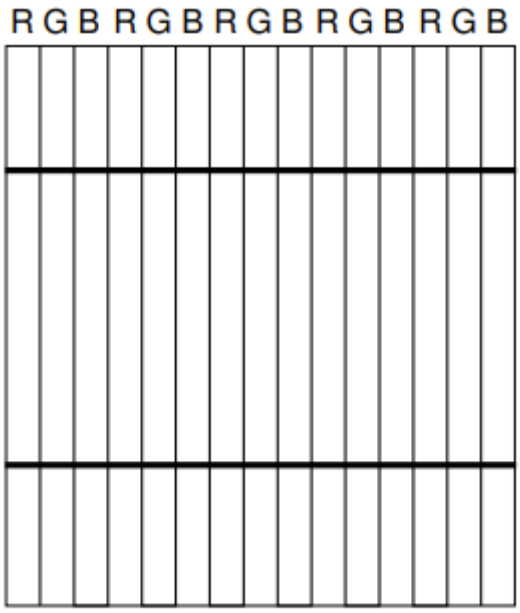
- Vypouklost masky se postupně zmenšovala -> obrazovky typu Delta poměrně nekvalitní obraz



- Obrázek 3: Maska Invarové obrazovky

Trinitron

- Alternativa Inveru od Sony
- Pevně natažené dráty -> horizontálně -> fosforu proniká více elektronů -> větší svítivost
- Další dva dráty vertikálně -> kvůli upevnění horizontálních
- Výška menší než šířka -> snižuje zkreslení -> obrazovka působí válcovitě s výsečí válce o poloměru 2m
- Také má vyhody a nevýhody
    - Ostrost a kontrastnost v rozích, kde delta vykazovala zhoršení ostrosti
    - Náchylnost k interferancím elektromagnetického pole



Obrázek 4: Maska Trinitronové obrazovky

Parametry CRT monitorů

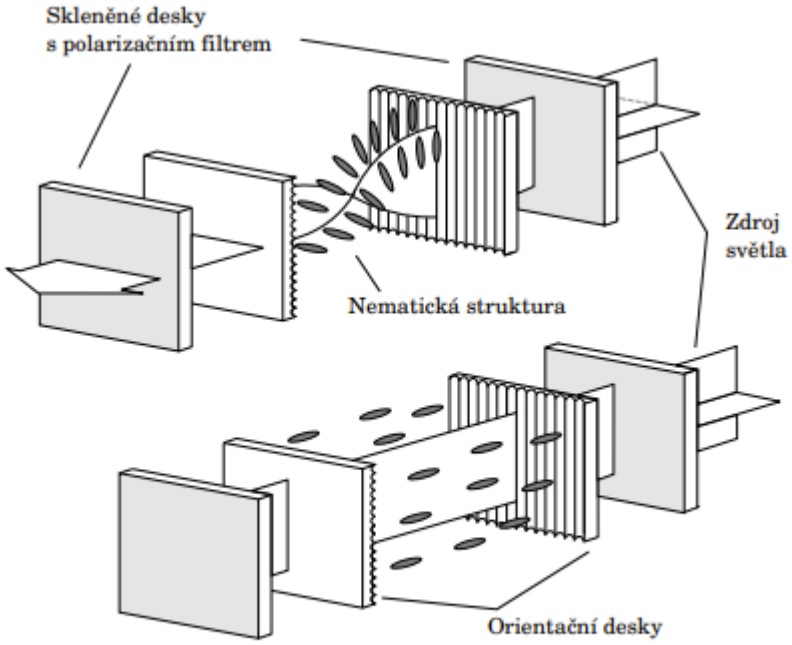
- Uhlopříčka, rozlišení, horizontální frekvence, vertikální frekvence šířka pásma, bodová rozteč, refresh rate

Výhody a nevýhody CRT displayů

- Ostrost, věrohodné barvy, odezva, pozorovací uhly
- Velikost, Spotřeba, Vyzařování

LCD monitory

- Liquid crystal display

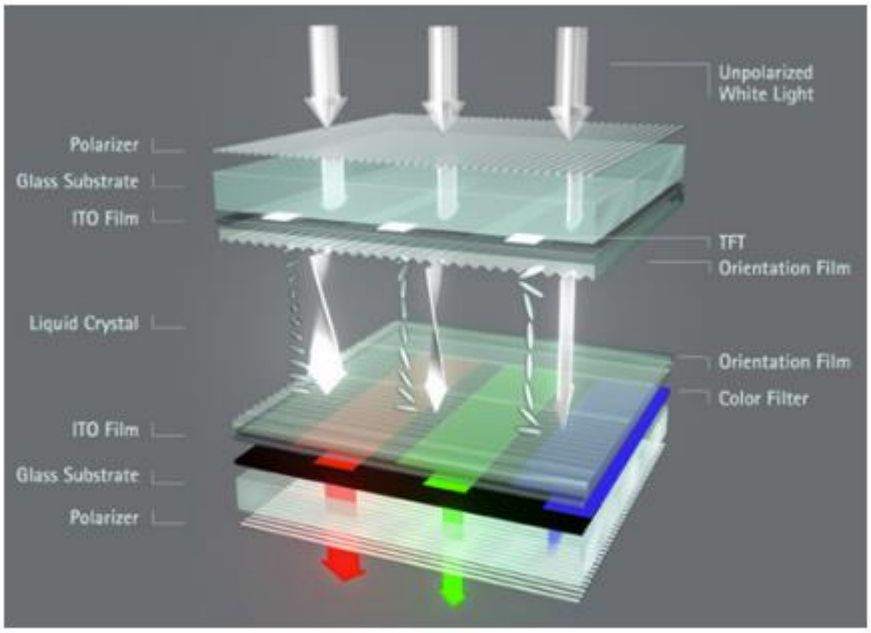


Obrázek 5: Princip činnosti LCD displeje

-
- Jádnem LCD je TN (twisted nematic) struktura která je z obou stran obklopena polarizačními prvky -> ty jsou natočené stejně jako drážkové orientační desky
- Nepolarizované světlo projde prvním polarizačním sklem -> polarizuje se -> prochází první vrstvou pootočených krystalů, které světlo otočí o 90 stupňů, a to projde dalším polarizačním sklem, které je otočené o 90 stupňů oproti prvnímu -> klidové chování TN-LCD bez přivedeného napětí -> propouští světlo
- Jakmile začne krystaly protékat malý proud -> krystaly se otáčí po směru toku proudu -> všechny zrnka se tedy točí stejně a přestane docházet k otáčení světla
- První polarizační vrstva tedy světlo polarizuje -> skrze krystaly projde světlo nezměněně a druhá polarizační vrstva světlo kompletně zablokuje, neboť je jeho polarita o 90 stupňů odlišná
- Světlo, které prochází vrstvou tekutých krystalů může tedy otáčet svou polarizaci -> intenzitou řídícího střídavého proudu určíme kolik světla krystaly propustí
- Vrstva krystalů je rozdělená na stejně velké malé buňky, které tvoří jednotlivé body display
- Panel musí být podsvícen bílým světlem -> elektroluminiscenční výbojky

Barevné LCD

- Konstrukce je téměř stejná jako u jednobarevných z obr 6
- Jeden bod se však skládá ze 3 menších a podle míry propuštěného světla do daného bodu určujeme barvu



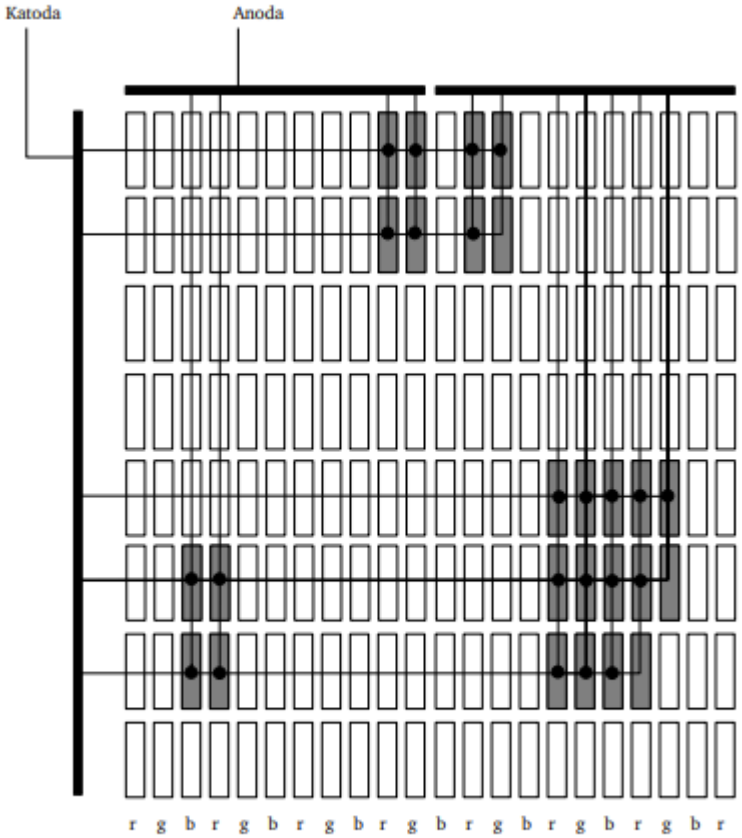
- Obrázek 6: Princip činnosti barevného LCD displeje

Pasivní display

- Mřížka vodičů nacházející se na průsečíku v přížce
- Dovede se na daný bod elektřina -> pokud je do řádku dovedena elektřina a druhý je uzemněný -> obvod se uzavře a stav krystalu v bodu se změní z bílého na černý -> opakováním tohohle vznikne obraz
- Problém je při velkém počtu řádků a sloupců, elektrod je moc a obraz pak není ostrý -> aktivní bod je černý taky částečně aktivní jsou šedé
- Taky není moc rychlý odezva dat z GPU je cca 200ms což jsou cca 3 snímky -> pak vzniká čára za kurzorem například

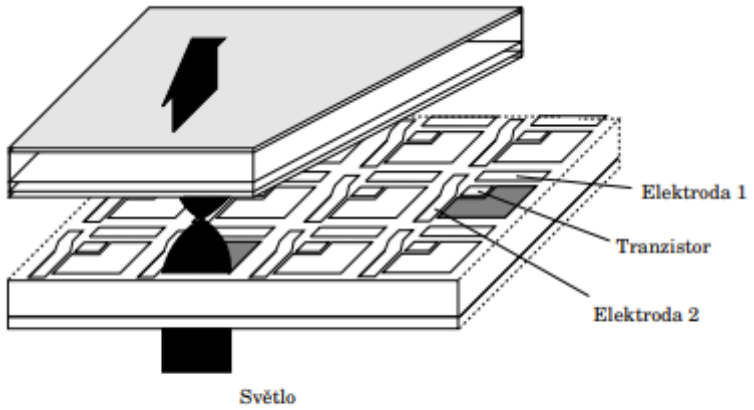
Aktivní display

- Vyvinuto, aby eliminovalo problémy pasivního displaye
- TFT display (Thin film tranzistor) -> prvně vyrobeno roku 1972
- Je vyrobený z amorfního křemíku a později z polykrystalického křemíku
- Zase každý bod v mřížce jsou kontakty (dioda/tranzistor) -> menší proud na ovládání svítivosti bodu -> díky tomuto mohl být v maticovém zobrazovači vypínán a zapínán proud daleko častěji -> vyšší refresh rate
- Dal se přesně regulovat proud v bodu a tím přesně i jeho svítivost
- TF tranzistory kompletně izolují body, tímto se vyřešil problém z pasivních displayů
- Nativní rozlišení bývá 1600x1200 -> 4800 sub pixelů -> šířka bodu je 0,24-0,29 mm



Obrázek 7: Struktura pasivního displeje

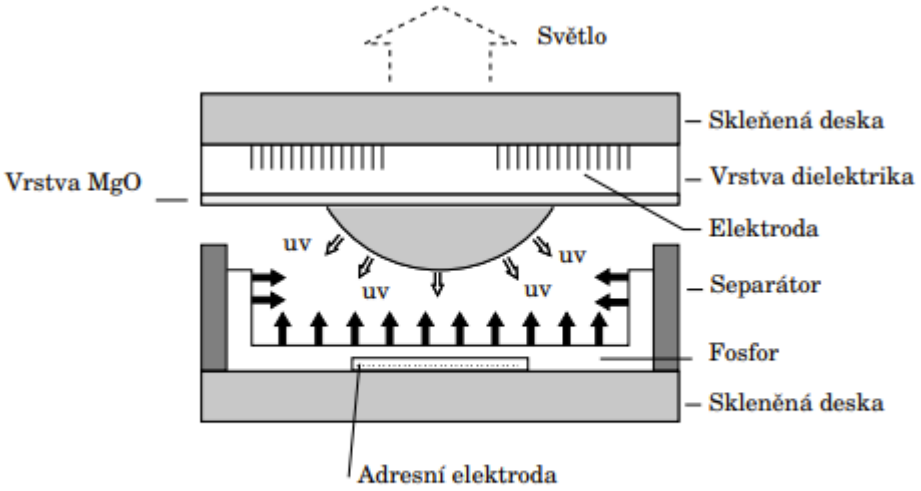
-
- Parametry LCD monitorů
    - Uhel pohledu, doba odezvy, kontrast
- Výhody nevýhody
    - Kvalita obrazu, životnost, spotřeba, odrazivost a oslnivost, bez emisí
    - Citlivost na teplotu, pevné rozlišení, vadné pixely, doba odezvy



Obrázek 8: Struktura TFT displeje

Plazmové monitory

- Plazma – superskupenství složeným z iontů a elementárních částic -> čtvrté skupenství
- Klidový stav -> směs vzácných plynů -> argon, neon, xenon -> jelikož jsou to elektro neutrální atomy, musíme najít způsob, jak z nich udělat plazmu -> zavedení elektrického proudu do plynu -> vznikne mnoho volných elektronů -> srážky elektronů s částicemi plynu vyvolají ztrátu elektronů některých atomů a vzniknou ionty -> spolu s elektrony získáváme plazmu



Obrázek 9: Princip činnosti jedné buňky plazma displeje

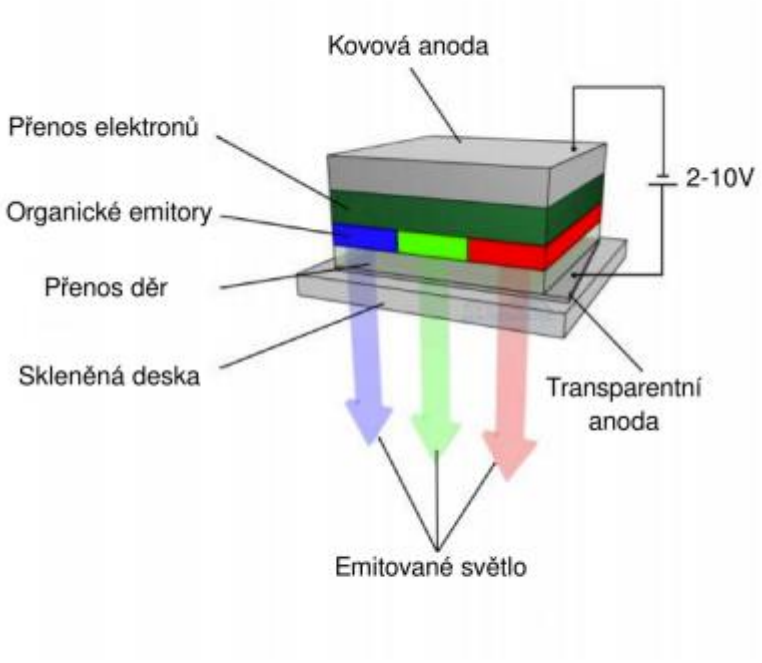
-
- Jednotlivé nabité částice se vlivem elektrického pole začnou pohybovat ke svým opačným pólům -> ionty plynů k záporně a elektrony ke kladně nabitému pólu
- V tomto zmatku do sebe částice naráží a předávají si energii, pak jí uvolní v podobě fotonu
- Celý plazma display je tvořen maticí miniaturních fluorescentních buněk (bodů) ovládaných sítí elektrod
- Horizontální řádky tvoří adresovací elektrody -> vertikální tvoří zobrazovací (výbojové) elektrody -> každou buňku lze adresovat samotnou -> buňky jsou mezi dvěma tenkými skleněnými tabulkami, každá obsahuje kondenzátor a tři elektrody
- Adresovací elektroda je na zadní straně buňky a dvě zobrazovací elektrody na přední straně -> tyto dvě diody jsou izolovány diaelektrikem a chráněny vrstvou oxidu hořečnatého
- Zase 3 barevné sub pixely RGB
- Do obou zobrazovacích elektrod je pouštěno střídavé napětí -> po zavedení se provede výboj, který začne ionizovat a vytvářet plazmu -> diaelektrikum a oxid hořečnatý výboj sice zastaví, ale pak proběhne změna polarity, takto dostaneme stálý výboj -> napětí se drží těsně pod hladinou -> pak při mírném zvýšení se zaktivuje proces vytváření plazmy na adresovací diodě
- Poté díky kinetické energii začne xenon a neon vytvářet UV světlo, a to se přes luminofory stane výslednou barvou
- Ovládání intenzity jednotlivých buněk funguje na principu pulzního kodu PWM (Puls wide modulation) -> tato modulace slouží k převedení analogového signálu s nekonečným rozsahem na binární slovo -> intenzita subpixelů je dána počtem a šířkou napěťových pulzů -> rozdělení snímků na podsnímky
- V této době jsou pixely které mají svítit nabuzeny pomocí robrazovacích elektrod na určité napětí -> rozbrazovací fáze je udělané pomocí zobrazovacích elektrod na celý display tím se rozsvítí pouze nabuzené pixely s danou urovni nabití
-

-
- Výhody a nevýhody
    - Kvalitní kontrastní obraz, není nutné podsvícení, velké pozorovací uhly, minimální hloubka a hmotnost
    - Pamětová efekt, levné displaye mají problémy s kontrastem, cena, energetická náročnost

OLED

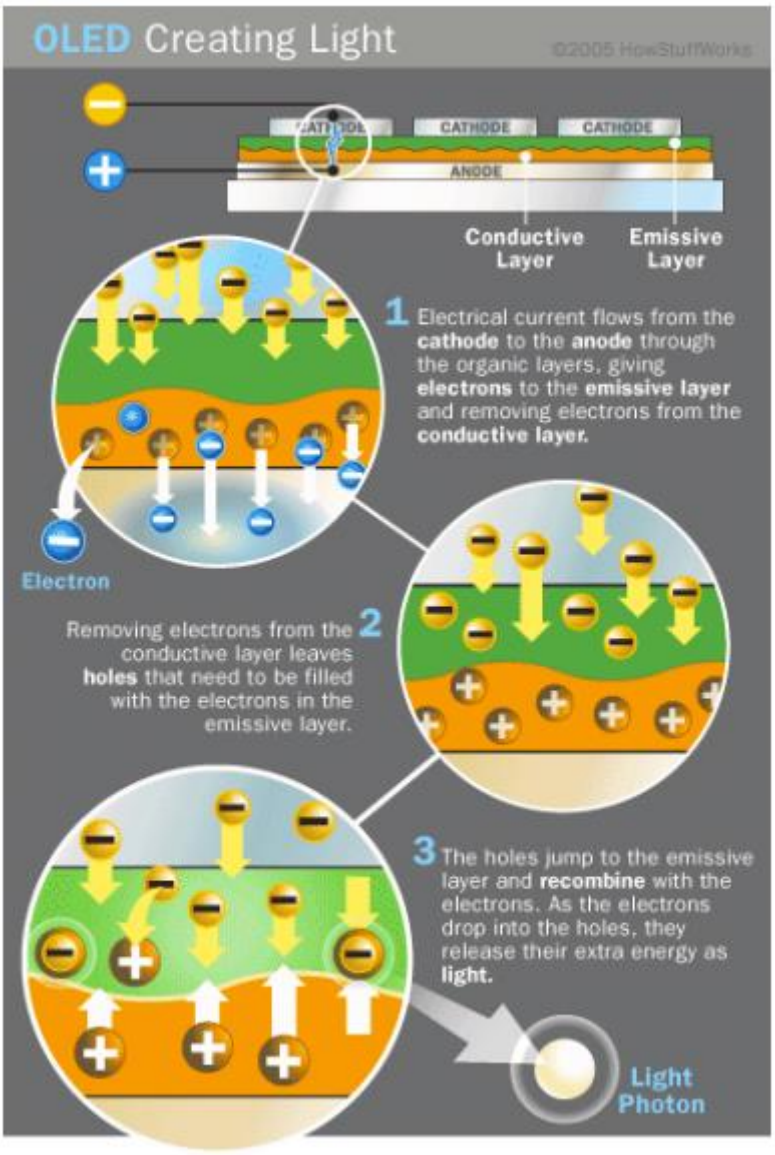
- Organic light emmiting diode/display -> organická dioda emitující světlo

Zjednodušenou strukturu OLED displeje můžeme znázornit obrázkem 10.



Obrázek 10: Základní struktura OLED displeje

-
- Na kovovou katodu jsou naneseny další vrstvy -> vodivá vrstva pro elektrony -> dále samotné organické světlo emitující vrstva -> vrstva pro přenos děr a průhledná katoda -> ochranná skleněná vrstva
- Po přivedení napětí se na obě elektrody začnou hromadit elektrony na straně organické vrstvy blíže k anodě -> díry představují kladné částice se hromadí na opačné straně blíže ke katodě -> částice se začnou srážet a postupně propadávat sítem -> opět energie vyzářena jako fotony -> rekombinace

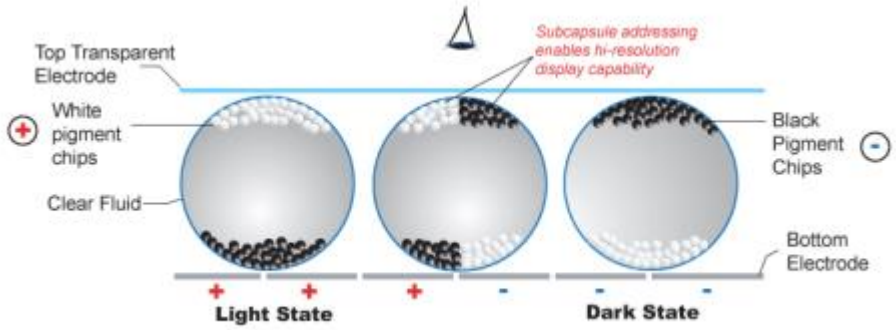


Obrázek 11: Princip činnosti OLED

- AMOLED A PMOLED
  - Passive a Active matrix OLED
  - Stejný princip jako u LCD aktivní a pasivní
  - Zase je zde matice -> každá OLED je aktivována u pasivních displayů dvěma na sebe kolmými elektrodami, procházejícími celou šířku a výšku displaye
  - U aktivního je každá vlastní část monitorovaná vlastním tranzistorem
- Vlastnosti OLED, výhody, nevýhody
  - Energeticky málo náročné, protože jsou zdrojem vlastního světla, dobrý kontrast, výkon OLED dosahuje 20-30 lm/W (žárovka má 10-15), výjimečně může až 50 lm/W u bílých OLED
  - Výhody
    - Velmi tenké, dobrý kontrast, plně barevné, nízká spotřeba, dobré pozorovací uhly, prakticky bez zpoždění, snadná výroba, možné dát na pružný podklad
  - Nevýhody
    - Moc ne

E-Ink

- EPD – electic paper device -> nepotřebuje elektřinu pro zobrazení statické grafiky



-
- Jednotlivé body tvořeny kapslemi o velikost desítek či stovek nanometru -> tyto kapsle jsou elektroforetické (elektricky separovatelný) roztok -> v tomto roztoku jsou záporně nabité černé částice (inkoust) a kladné bílé částice -> velikost řádově v jednotkách nanometru
- Kapsle jsou mezi dvěma elektrodama, když je přiveden elektrický proud, oddělí se nám barvy za opačně nabitou elektrodou -> pomocí dělených elektrod můžeme dělat kombinaci těchto dvou stavů (tmavý světlý)
- Roztok musí být chemicky stabilní -> průsvitný roztok je hydrokarbonový olej -> taky díky své viskozitě udržuje částice tam, kde mají být i po odpojení elektřiny -> černé částice jsou uhlík -> bílé mají jádro z oxidu titaničitého a obal je oxid křemíku
- Proud na rozdělení stačí velmi malý, třeba i pár nanoAmpér při napětí 5-15V
- Po odpojení napájení už není potřeba nic na udržení částic v dané poloze
- Velmi kontrastní obraz a dobrý pozorovací uhel (180)
- Velké zpoždění na požadavek o změnu obsahu stránky -> řádově stovky ms
- Není pro videosekvence
- Umí vytvořit pouze 16 odstínů černobílé
- Existuje i RGB verze -> 4000 barev
- Výhody a nevýhody E-Ink
- Výhody
    - Vysoké rozlišení, dobrý kontrast, čitelnost na slunci, není nutné podsvětlení, velmi tenké, možnost pružného podkladu, nulová spotřeba proudu při zobrazení statické informace, minimální spotřeba při kreslení
- Nevýhody
    - Málo odstínů šedi, špatné barevné rozlišení, zpoždění při překreslování

CUDA, a tak no :)

Historie GPU

- 1970 - ANTIC in 8-bit Atari
- 1980 - IBM 8514
- 1993 - Nvidia Co. založeno
- 1994 - 3dfx Interactive založeno
- 1995 - chip NV1 od Nvidia
- 1996 - 3dfx vydalo Voodoo Graphics
- 1999 - GeForce 256 by Nvidia - podpora geometrických transformací
- 2000 - Nvidia kupuje 3dfx Interactive
- 2002 - GeForce 4 vybaveno pixel a vertex shadery
- 2006 - GeForce 8 - unifikovaná architektura (nerozlišuje pixel a vertex shader) (Nvidia CUDA)
- 2008 - GeForce 280 - podpora dvojité přesnosti
- 2010 - GeForce 480 (Fermi) - první GPU postavené pro obecné výpočty - GPGPU

- Výhody GPU
  - Hodně vláken – 512+
  - Vlákna jsou nezávislá a pracují jak jen zvládnou
  - Neměli by být využity podmíněné skoky
  - Optimalizováno pro přístup do paměti cca 200gb/s

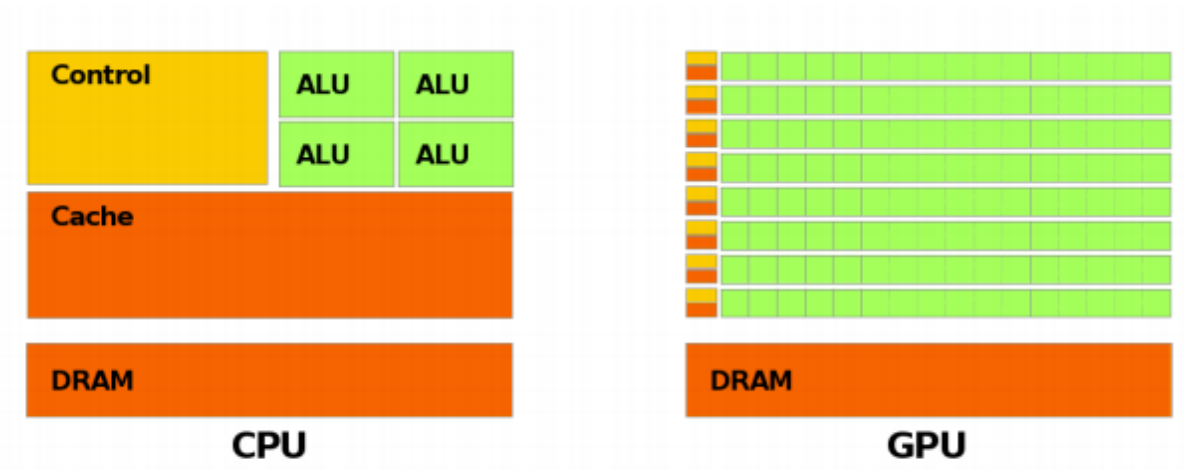
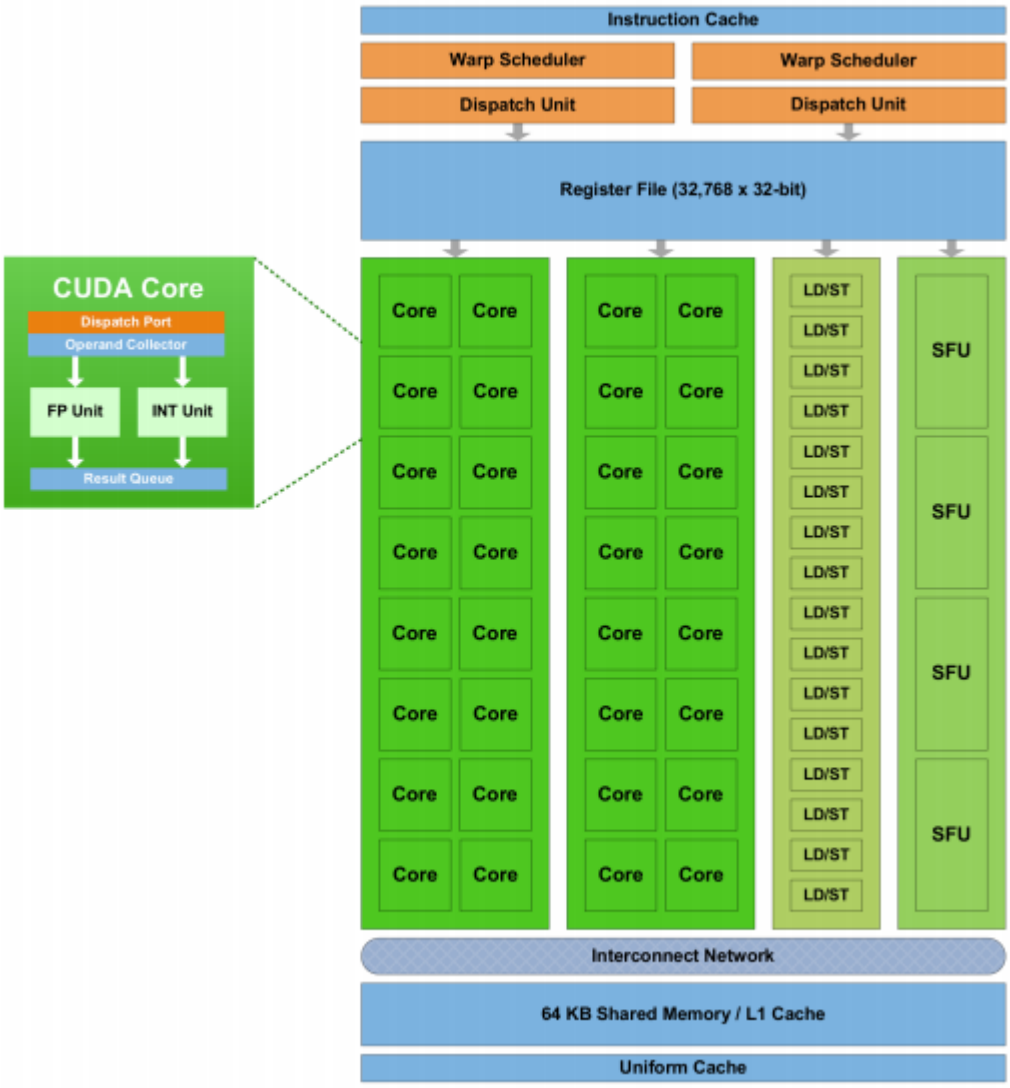


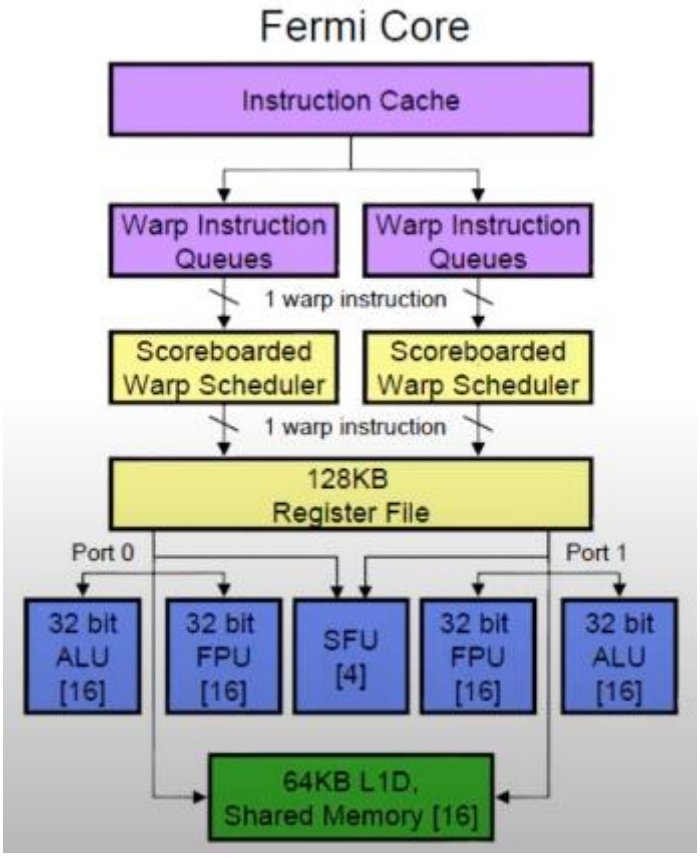
FIGURE: Zdroj Nvidia Programming Guide

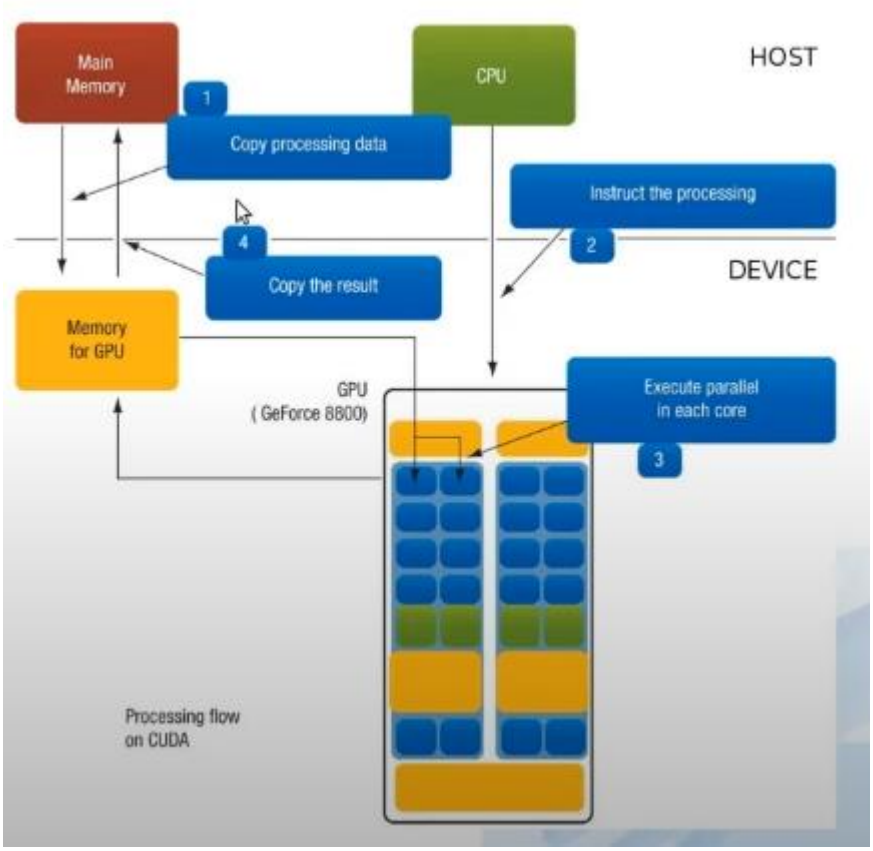
GPGPU

- General purpose GPU
- Na začátku bylo pro GPGPU nutné využívat rozhraní OpenGL
- Formulováno pomocí textur s operací s pixely
- Kvůli vývojářům vznikly shadery
    - Jednoduchý procesor pro práci s pixelama
    - Podpora floating výpočtu
    - Velikost kodu byla omezena na pár instrukcí
- CUDA roku 2007
    - Výrazně zlepšuje GPGPU
    - Nevyžaduje openGL
    - Rozšíření C/C++
    - Funguje jen na Nvidii
- Cuda Architecture 1
    - GeForce 580
        - 16 multiprocesorů kde každý má
            - 32 jader pro jednotlivé vlákna
            - 64kB rychlé paměti -> něco jako cache
            - Pamět se dělí na 16 modulů -> jeden pro každé vlákno
        - 6x 64bit pamětové moduly pro 384bitový přístup a až 6gb RAM
        - 768 kb L2 cache
- CUDA architecture 2

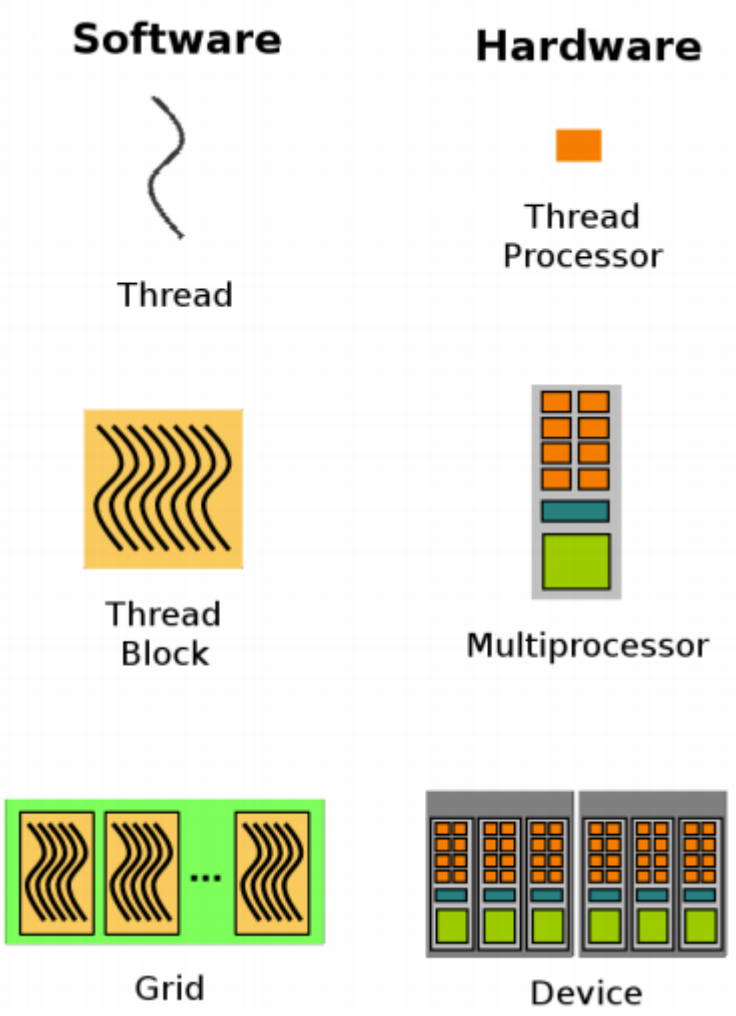


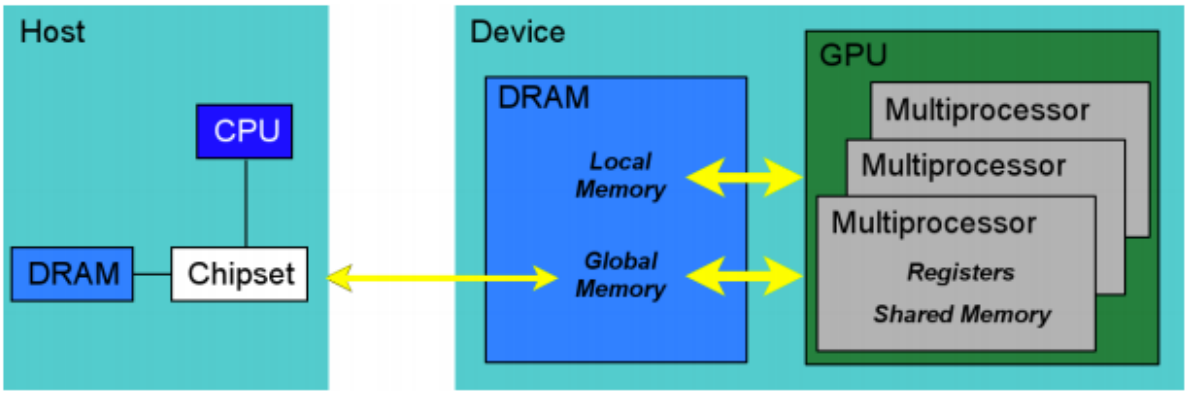
-
- Schéma jednoho multiprocesoru
- Core se dělí na warpy -> ten dvojsloupec Core, tady jsou dva warpy
- Nahoře je vidět pro každý warp jeho dekodér instrukcí -> jedna dekodovaná instrukce jde do všech jader
- 32x LD/ST je load a store -> pomocné jednotky
- 4x SFU -> pomocné transcendentní funkce (sinus cosinus logaritmus atd…)
- 32x 2x BTW každé jádro má část pro počítání s int a float čísly

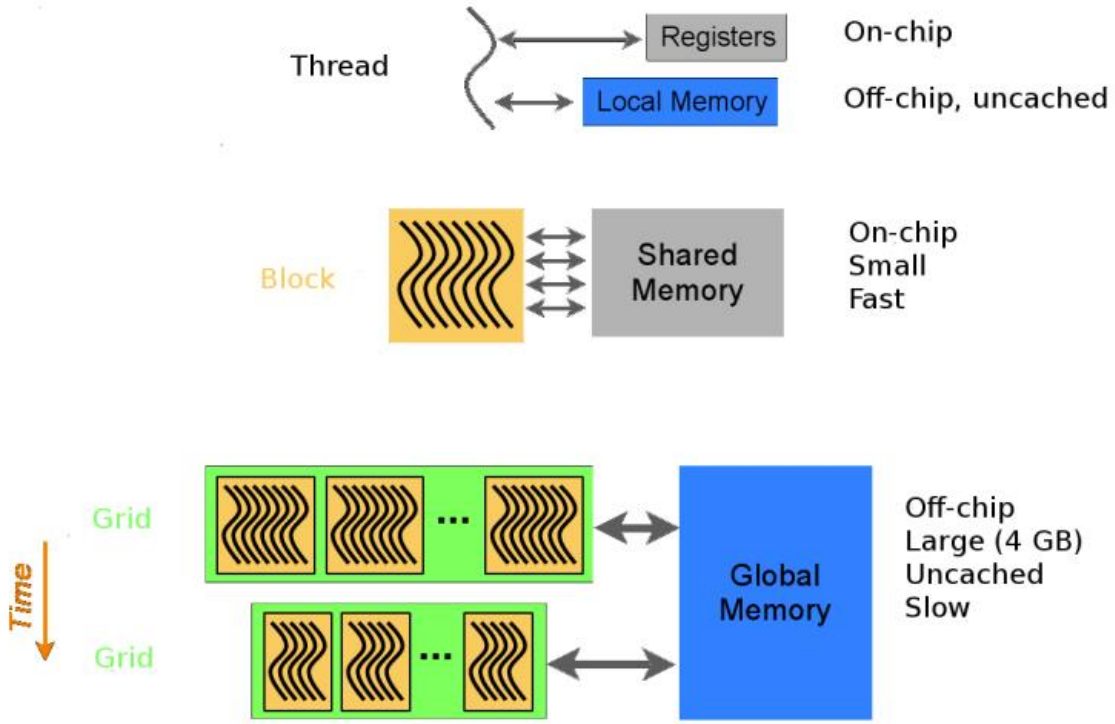




-
- Musíme mít nějaký obrázek nebo cokoliv -> připravíme data do GPU memory -> CPU zadá příkaz, příkaz se provede -> jde do GPU memory a zpátky do main memory
- CUDA host je procesor a pamět

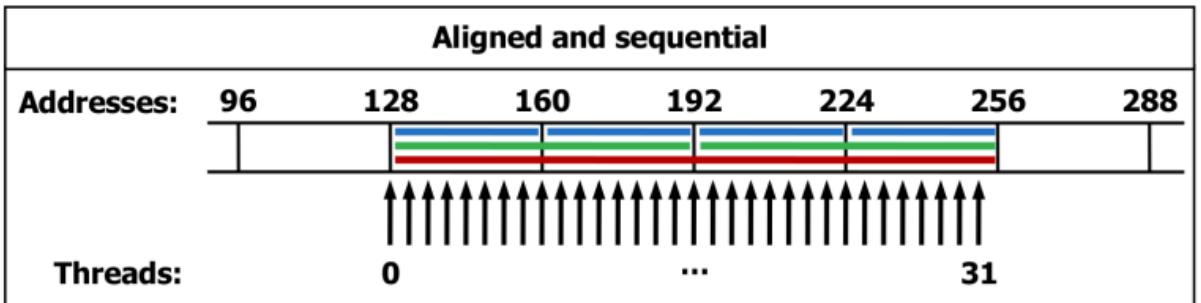
- CUDA device je GPU
- CUDA thread struktura která se přepíná při zpracování
- CUDA bloky -> jeden blok je zpracovávaný na jednom vláknu -> v jednom bloku už zvládneme synchronizovat -> max size bloku je 1024
    - Procesor přepíná mezi vlánky a tím zakrývá dropnou latenci pomalé globální paměti
    - Pracují vždy vlákna co mají načtená potřebná data, ostatní načítají
- Bloky vláken se pak seskupují do Gridu
- 

- 



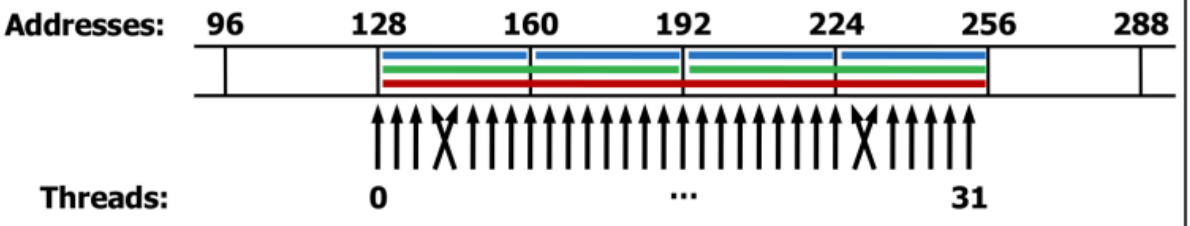
-
- Efektivita programování
  - Redukovat přenos dat mezi CPU a GPU
  - Optimalizovat přístup do globální paměti
  - Omezit divergentní vlákna
  - Správná velikost bloků
- Komunikace přes PCI express je pomalá -> minimalizovat ji
- Pokud je nutná častá komunikace -> just use pipelining
- Najednou je možné dělat
  - Výpočet na GPU
  - Výpočet na CPU
  - Kopírování dat z CPU na GPU
  - Kopírování dat z GPU na CPU
- Většina přístupů GPU do globální paměti tvoří načítání textur
- GPU je silně optimalizováno pro sekvenční přístup do globální paměti
- Musíme se tedy vyhnout náhodným přístupům do globální paměti
- Ideálka je
  - Načíst data do sdílené paměti multiprocesoru
  - Provést výpočty
  - Zapsat výsledek do globální paměti
- Sloučený přístup může snížit (až 32x) počet paměťových transakcí

- **Aligned and sequential**



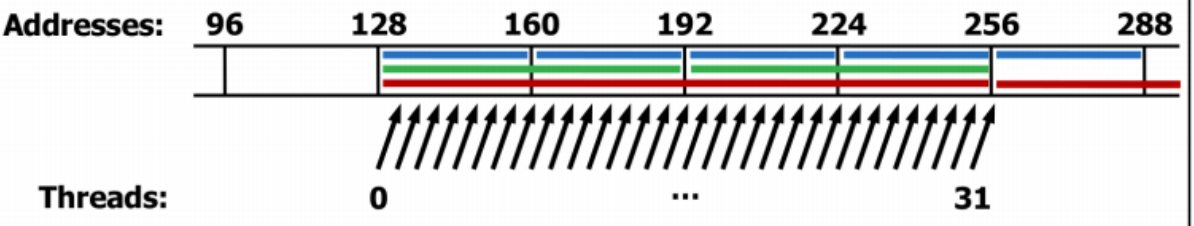
Compute capability:	1.0 and 1.1	1.2 and 1.3	2.0
Memory transactions:	Uncached		Cached
	1 x 64B at 128 1 x 64B at 192	1 x 64B at 128 1 x 64B at 192	1 x 128B at 128

- **Aligned and non-sequential**



Compute capability:	1.0 and 1.1	1.2 and 1.3	2.0
Memory transactions:	Uncached		Cached
	8 x 32B at 128 8 x 32B at 160 8 x 32B at 192 8 x 32B at 224	1 x 64B at 128 1 x 64B at 192	1 x 128B at 128

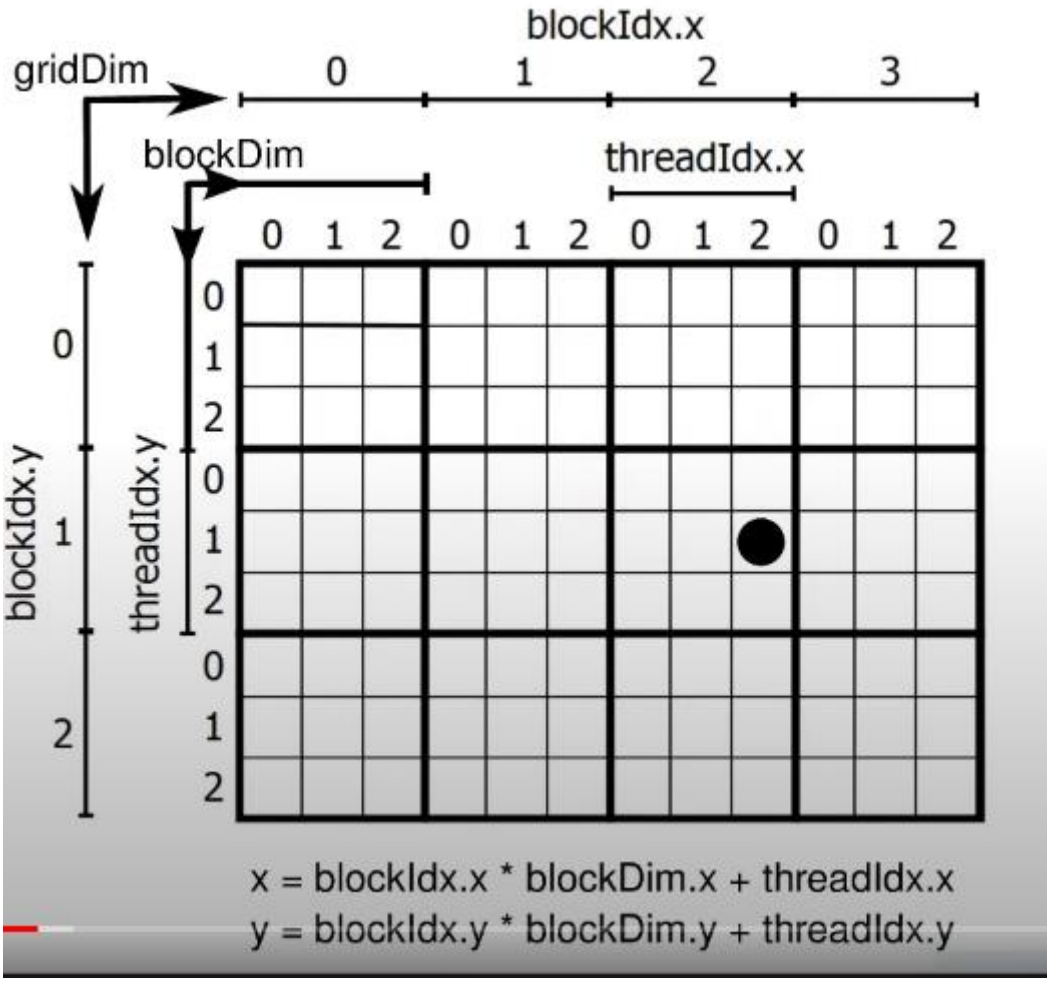
- **Misaligned and sequential**



Compute capability:	1.0 and 1.1	1.2 and 1.3	2.0
Memory transactions:	Uncached		Cached
	7 x 32B at 128 8 x 32B at 160 8 x 32B at 192 8 x 32B at 224 1 x 32B at 256	1 x 128B at 128 1 x 64B at 192 1 x 32B at 256	1 x 128B at 128 1 x 128B at 256

- Architektura FErmi zavádí plně funkční L1 a L2 cache
- L1 cache je na každém multiprocesoru
    - Tu lze nastavit pomocí tohohle
    - `cudaFuncSetCacheConfig( MyKernel, cudaFuncCachePreferShared )`
        - `cudaFuncCachePreferShared` - shared memory is 48 KB
        - `cudaFuncCachePreferL1` - shared memory is 16 KB
        - `cudaFuncCachePreferNone` - no preference
- L2 cache je společná pro všechny multiprocesory a má 768Kb
- Sdílená pamět multiprocesoru je rozdělená na 16 pamětových bank
- Data se do bank ukládají po 4 bytech
- Je nutné se vyhnout situaci kdy dvě vlákna ze stejné skupiny čtou z různých adres v jedné bance
- Nevadí, že čte více vláken ze stejné adresy -> broadcast
- Divergentní vlákna
    - CUDA umí zpracovávat více kernelů najednou, ale jenom na jiných multiprocesorech
- Na multiprocesoru běží více bloků vláken -> scheduler mezi nimi přepíná a pouští ty bloky, které mají data ready -> je potřeba aby jeden blok nevypotřeboval všechny registry a sdílenou pamět -> pokud nejsou registry -> uloží se do local memory -> pomalá pamět
    - Je potřeba zvolit dobrou velikost bloku -> násobek 3
    - Minimalizovat počet proměnných a množství sdílené paměti
    - Minimalizovat kod kernelu
- Efektivnost procesoru udává occupancy (maximum je 1.0)
    - Za účelem optimalizace lze použít
        - CUDA occupancy calculator
        - CUDA profiler
        - Výpisy nvcc -ptxas-options=-v
- GPU je v mnoha ohledech lepší než CPU, ale je potřeba mu rozumět -> vyvíjí se velmi rychle a člověk potřebuje stíhat tento pokrok
- CUDA3 a Fermi
    - Podpora cache na multiprocesoru – 64kB
    - Podpora ECC RAM
    - Uplná podpora C++
    - Printf jako funkce kernelu
- CUDA4
    - Lepší podpora pro počítání na více GPU
        - Lze obsluhovat více GPU z jednoho vlákna
    - Unifikovaný adresový prostor
- Zbytek světa
    - ATI/AMD – Radeon GPUs
        - Podporuje OpenCL (nvidia taky)
            - OpenCL nepodporuje C++ a Fortran
    - AMD Fusion – GPU implementované na základní desce a sdílející pamět s CPU
        - Odstranuje nutnost přenosu CPU -> GPU

        - Pracuje však s pomalou DRAM
    - Intel
        - Nové CPU od intelu obsahují integrovanou GPU
        - Larabee architektura
- Budoucnost CUDA
    - Nvidia vede v tomto závodě karet díky GPGPU díky CUDA
    - Brzy má podporovat vícejádrové systémy = běh kernelů na x86
    - Nvidia nemá vlastní CPU -> investuje do ARM architektury Nvidia Tegra
        - Microsoft oznámil podporu win8 na ARM CPUs
        - AMD ohlásilo vývoj ARM CPU
        - Nvidia plánuje vytvořit ARM CPU pro HPC



x = blockIdx.x * blockDim.x + threadIdx.x
y = blockIdx.y * blockDim.y + threadIdx.y