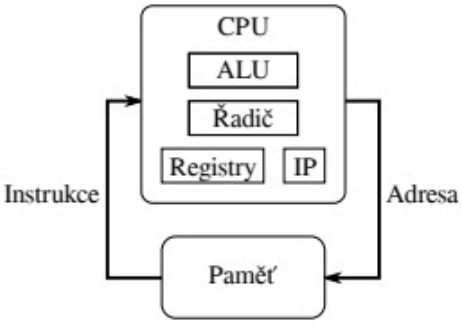
1. Architektura počítače (Architektura počítače – pohled na podstatné vlastnosti počítačů.)

Čím je tvořen počítač - CPU, paměť a periférie.

- **Operační paměť:** slouží k uchování zpracovávaného programu, zpracovávaných dat a výsledků výpočtu
- **ALU - Arithmetic-logic Unit (aritmetickologická jednotka):** jednotka provádějící veškeré aritmetické výpočty a logické operace. (Obsahuje sčítačky, násobičky (pro aritmetické výpočty) a komparátory (pro porovnávání))
- **Řadič:** řídící jednotka, která řídí činnost všech částí počítače. Toto řízení je prováděno pomocí řídících signálů, které jsou zasílány jednotlivým modulům. Reakce na řídící signály, stavy jednotlivých modulů jsou naopak zasílány zpět řadiči pomocí stavových hlášení
- **Vstupní zařízení:** zařízení určená pro vstup programu a dat.
- **Výstupní zařízení:** zařízení určená pro výstup výsledků, které program zpracoval
- **CPU - Central Processor Unit (centrální procesorová jednotka)/Procesor:** Řadič + ALU

Základní princip fungování počítače

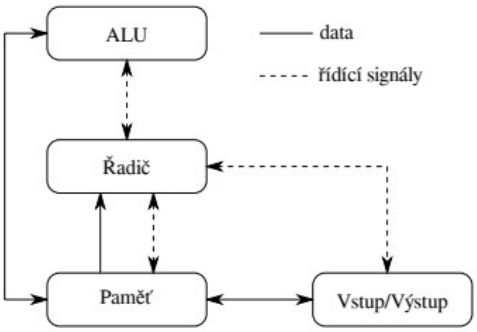
Pro vysvětlení základního principu fungování počítače bude stačit pouze procesor a paměť. Dále je potřeba mít na paměti informace: počítač je programován obsahem paměti, instrukce z paměti se vykonávají sekvenčně a každý následující krok závisí na kroku předchozím.
Z toho je zřejmé, že procesor je sekvenční obvod a vstupem tohoto obvodu budou strojové instrukce z paměti. Pro názornost a další vysvětlení fungování je procesor i s pamětí zobrazen na obrázku:

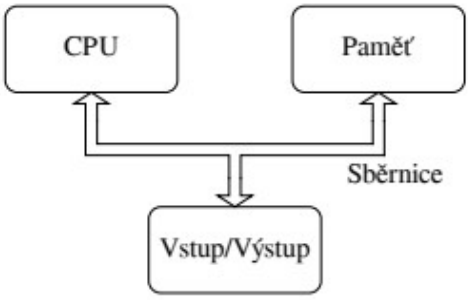


Obrázek 4: Zjednodušený princip fungování počítače

Procesor, jako sekvenční obvod, využívá paměť. Dosud byla paměť uváděna jako samostatná část počítače. Ve skutečnosti je velmi malá část paměti počítače umístěna i přímo v procesoru a to ve formě registrů. Jedním z těchto registrů je IP (Instruction Pointer), ukazatel do paměti na instrukci, která má být provedena, či je prováděna. U některých procesorů bývá tento registr označován jako PC (Program Counter – jeho význam je však stejný jako IP).
Má-li procesor vykonat strojovou instrukci, jeho řídící část využije aktuální hodnotu registru IP a prostřednictvím sběrnice si vyžádá (přečte) z paměti instrukci na adrese IP.
Poté co procesor získá z paměti strojovou instrukci, tak ji provede. Následně zvýší IP o délku právě provedené instrukce a celý cyklus čtení a provedení instrukce se opakuje.
Pokud právě prováděná instrukce potřebuje data z paměti, tak si je procesor vyžádá stejně, jako si načetl instrukci samotnou. (Adresa těchto dat je obvykle součástí prováděné strojové instrukce).
Uložení výsledku řeší procesor podle dokončené strojové instrukce. Výsledky se mohou ukládat do registrů a v tom případě zůstávají v procesoru. Do paměti se výsledky strojové instrukce uloží opět prostřednictvím sběrnice.

Koncepce počítače dle von Neumanna.





Vlastnosti:

- **počítač se skládá z CPU, paměti a periférií**
- počítač musí být univerzální pro jakýkoliv úkol
- počítač se bude programovat obsahem paměti pomocí strojových instrukcí
- paměť bude pro data i kód (společná)
- instrukce budou vykonávány sekvenčně (za sebou)
- sekvenční vykonávání může být změněno pomocí skokových instrukcí
- paměť bude složena z buněk stejné velikosti (1B = 8 bitů) a pořadové číslo buňky bude adresa
- vše v paměti bude ve dvojkové soustavě (data i instrukce)
- počítač bude řízen pomocí řadiče, instrukce budou prováděny v ALU jednotce
- řídící a výpočetní částí počítače je CPU
- pro vstup a výstup slouží V/V (I/O) zařízení
- vše je propojeno a komunikuje prostřednictvím sběrnice

Princip činnosti počítače podle von Neumannova schématu

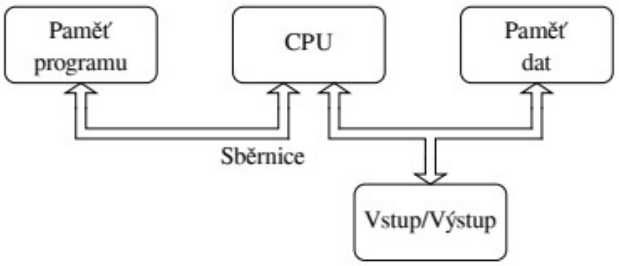
1. Do operační paměti se pomocí vstupních zařízení přes ALU umístí program, který bude provádět výpočet.
2. Stejným způsobem se do operační paměti umístí data, která bude program zpracovávat
3. Proběhne vlastní výpočet, jehož jednotlivé kroky provádí ALU. Tato jednotka je v průběhu výpočtu spolu s ostatními moduly řízena řadičem počítače. Mezivýsledky výpočtu jsou ukládány do operační paměti.
4. Po skončení výpočtu jsou výsledky poslány přes ALU na výstupní zařízení.

Výhody: rozdělení paměti pro kód a data určuje programátor, řídící jednotka procesoru přistupuje do paměti pro data i pro instrukce jednotným způsobem, jedna sběrnice – jednodušší výroba, počítač je univerzální (to co programátor naprogramuje tak mu to vypočítá)

Nevýhody: společné uložení dat a kódu může mít při chybě za následek přepsání vlastního programu, jediná sběrnice – malá propustnost, chybí paralelismus – může se vykonávat pouze jeden program a ne více současně (paralelismus je simulován operačním systémem, úlohy jsou přepínaný a vytváří se dojem paralelismu)

(Základní odlišnosti dnešních počítačů od von Neumannova schématu: dnešní počítač zpracovává programy paralelně (využívá multitaskingu), počítač může mít i více než jeden procesor, existují I/O zařízení, které umožňují jak vstup tak výstup dat, program se nezavádí celý do paměti, ale pouze jeho části, když jsou potřeba při konání programu)

Harvardská architektura



- instrukce (program) a data jsou v oddělené paměti
- základní koncepce a principy fungování jsou u harvardské i von Neumannovy architektury stejné
- v některých systémech se pro paměť programu používá typ paměti ROM (read only memory), přičemž paměť dat vyžaduje typ paměti RWM (Read-Write Memory)

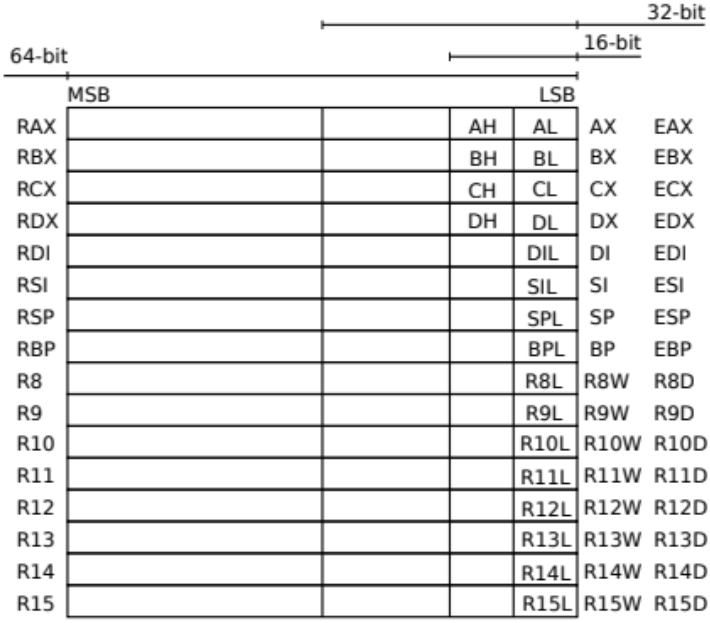
Výhody: oddělení paměti dat a programu přináší výhody: program nemůže přepsat sám sebe, paměti mohou být vyrobeny odlišnými technologiemi, každá paměť může mít jinou velikost nejmenší adresovací jednotky, dvě sběrnice umožňují souběžné přistupovat pro instrukce i data
Nevýhody: dvě sběrnice kladou vyšší nároky na vývoj řídící jednotky procesoru a zvyšují i náklady na výrobu výsledného počítače, nevyužitou část paměti dat nelze použít pro program a obráceně.

2. Jazyk symbolických instrukcí, strojové instrukce, Assembler

Při programování s využitím strojových instrukcí se v dnešní době nepíše program přímo v binárním strojovém kódu. Programy se píší s využitím jazyka symbolických instrukcí (JSI), pro který je často i v českém jazyce užíván název Assembler. Každá strojová instrukce má v JSI svou zkratku a strojový kód je z JSI vytvořen až překladačem. První 64 bitový procesor x86 byl navržen a realizován firmou AMD. Firma provedla rozšíření sady osmi 32 bitových registrů procesorů předchozích generací na sadu registrů 64 bitových. Při přechodu z 32 na 64 nebyly pouze rozšířené registry, ale došlo také k navýšení počtu pracovních registrů na 16, tedy dvojnásobek.

Registry procesoru

Registr – malé a velmi rychlé úložiště dat, které využívá procesor při své činnosti, jsou součástí procesoru
– procesor přesouvá data z operační paměti do registru, aby je mohl zpracovat (např. aritmetickými strojovými instrukcemi), protože je omezený počet registrů v procesoru, jsou data z registru zapisována zpět do operační paměti



Obrázek 1: Přehled registrů 64bitového procesoru x86

Registry 64bitové pro všeobecné použití: RAX, RBX, RCX, RDX, RDI, RSI, RBP, RSP, R8 až R15

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

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

Registry 8 bitové: první čtyři 16bitové registry jsou rozděleny na horní a dolní 8 bitové části: AH (high), AL (low), BH, BL, CH, CL, DH, DL
další 8bitové registry jsou:
DIL, SIL, SPL, BPL, R8L až R15L

Další registry:
Čítač instrukcí RIP (IP) – ukazuje na aktuální vykonávanou instrukci, jeho změny se provádí skokovými instrukcemi, nikdy ne přímo
Stavový registr FLAGS – obsahuje stavové bity, které programátor využívá prostřednictvím podmíněných instrukcí.

Carry Flag (CF) – Tento příznak indikuje přetečení aritmetiky bezznaménkových celých čísel, nastaven na 1 když vznikne přenos největšího bitu výsledku
Overflow Flag (OF) – Nastavuje se pokud je celočíselný výsledek příliš velké kladné či záporné číslo (bez znaménkového bitu), které není možno uložit do cílového operandu. Jinak je vynulován. Tento příznak indikuje přetečení aritmetiky znaménkových celých čísel.
Sign Flag (SF) – Nastavuje se na hodnotu nejvyššího bitu výsledku, který obsahuje znaménko celých znaménkových čísel. 0 indikuje číslo kladné a 1 číslo záporné.
Zero Flag (ZF) – Nastaven v případě kdy výsledek je nula, jinak zůstává nulový.

Registry jsou pouze v některých případech vázány účelově. Pro úplnost některé příklady:
- RAX, EAX, AX, AL je akumulátor. Používá se při násobením dělení a v řetězcových instrukcích.
- RCX, ECX je používán jako čítač
- CL je použit jako čítač při bitových rotacích a posunech
- RDX, EDX, DX je horní část argumentu při násobení a dělení
- RSI, RDI je využíván jako indexový registr v řetězcových instrukcích

Adresování

Adresování rozdělujeme na adresování přímé a nepřímé. Při přímém adresování uvádíme vždy konkrétní pevnou adresu. Programátor ovšem v praxi potřebuje přímou adresu jen výjimečně, nejčastěji jsou za přímou adresu považovány adresy globálních proměnných. Pokud nelze adresu určit přímo, lze použít adresování nepřímé přes registry.
V 64bitovém režimu je adresování možné pomocí konstanty a dvou registrů:

[bázový + indexový * měřítko + konstanta]

Na místo bázového i indexového registru je možno použít kterýkoliv ze šestnácti 64bitových registrů. Měřítko je jedno ze čtyř čísel: 1, 2, 4 a 8. Usnadňuje manipulace s polem, jehož položky jsou jiné velikosti než 1 bajt.
Datové typy – specifikují jen velikost vyhrazené paměti, nikde se neříká nic o obsahu, zda jde o znak, celé číslo se znaménkem či bez, nebo číslo reálné. Za obsah a typovou kontrolu proměnné si odpovídá programátor.

- DB – data BYTE (8 bitů)
- DW – data WORD (16 bitů)
- DD – data DWORD (32 bitů)
- DQ – data QWORD (64 bitů)

Spojování programů v JSI a v jazyce C

V dnešní době není zvykem psát v JSI celé programy, ale pouze některé jeho části. Pro hlavní část se využívá např. Jazyk C. Tento jazyk se využívá k přípravě dat a následně pro výpisy výsledků, tedy pro základní vstupní a výstupní operace.

Uvedený kód v JSI musí být vždy rozdělen na část datovou a kód. V datové části je ukázáno, jak s použitím datových typů deklarovat proměnné *g_module_counter* a *g_module_sum.* V JSI platí pravidlo, že všechny symboly jsou lokální. Proto pro zveřejnění symbolu *g_module_counter* je nutno použít klíčové slovo global.

```
    ; 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
```

section .data – proměnné – default lokální, global – udělám globální, extern – proměnná je v C a chci k ní přistupovat i v assembleru
section .text – funkce (global hp_bar, další řádek: hp_bar:, další: enter 0,0 … leave, další: ret)

Základní instrukce přesunové

Přesun dat pomocí instrukce MOV – Instrukci MOV je možno použit několika způsoby:

```
MOV cíl, zdroj ; cíl = zdroj (přesune zdroj do cíle)
```

Jako operandy instrukce lze použít registry (R), proměnné v paměti (M) a konstanty (K).

```
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
```

Ve všech těchto pěti případech platí několik zásad, které platí i pro naprostou většinu dalších instrukcí:

- velikost operandů musí být stejná
- nikdy nelze použít dva paměťové operandy
- v kombinaci R, M a M, R a R, K určuje velikost operandu vždy použitý registr
- pokud není ani jeden operand registr, musí velikost operandu určit programátor pomocí typu (byte, word, dword, qword)

Přesun dat s rozšířenými instrukcemi MOVZX a MOVSX

```
MOVZX cíl, zdroj    ; cíl = zdroj (přesune zdroj do cíle)
MOVSX cíl, zdroj    ; cíl = zdroj (přesune zdroj do cíle)
```

Jako parametry instrukce lze použít registry (R) a proměnné v paměti (M) a to jen ve dvou kombinacích:

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

Pro oba parametry musí platit, že velikost operandu cíl musí být větší, než velikost operandu zdroj. Instrukce MOVZX provede přesun operandu s rozšířením o nuly do vyšším bitů. Instrukce MOVSX rozšíří operand znaménkově.

MOV cíl, zdroj

Instrukce provede přesun obsahu operandu zdroj do operandu cíl. Velikost obou operandů musí být stejná. Přesouvat lze obsah paměti, registru a konstantu.

CMOVcc cíl, zdroj

Instrukce provede přesun obsahu operandu zdroj do operandu cíl, pokud bude splněná podmínka cc. Velikost obou operandů musí být stejná. Přesouvat lze obsah paměti nebo registru do registrů.

MOVZX cíl, zdroj

Instrukce se používá v případě, že velikost operandu zdroj je menší, než velikost operandu cíl a provádí se rozšíření nulami.

MOVSX cíl, zdroj

Stejné jako MOVZX, ale provede se znaménkové rozšíření.

XCHG cíl, zdroj

Vymění se obsah obou operandů.

BSWAP cíl

Provede změnu pořadí bytů. Jedná se o konverzi little a big endian formátu čísla.

PUSH zdroj

Uloží na vrchol zásobníku obsah operandu zdroj a posune vrchol zásobníku.

POP cíl

Z vrcholu zásobníku se přesune hodnota do operandu cíl a sníží vrchol zásobníku.

Logické a bitové instrukce

Ovlivňují obsah příznakového registru procesoru. Logické instrukce mění SF a ZF, bitové posuny i CF.

AND cíl, zdroj

Instrukce provede bitově cíl = cíl and zdroj.

TEST cíl, zdroj

Instrukce provede bitově cíl and zdroj. Jde o operaci AND bez uložení výsledku.

OR cíl, zdroj

Instrukce provede bitově cíl = cíl or zdroj.

XOR cíl, zdroj

Instrukce provede bitově cíl = cíl xor zdroj.

NOT cíl

Instrukce provede negaci všech bitů operandu.

SHL/SAL cíl, kolik

Bitový i aritmetický posun doleva operandu cíl o požadovaný počet bitů. Operand kolik je konstanta nebo registr CL.

SHR cíl, kolik

Bitový posun doprava operandu cíl o požadovaný počet bitů. Operand kolik je konstanta nebo registr CL.

SAR cíl, kolik

Aritmetický posun doprava operandu cíl o požadovaný počet bitů. Operand kolik je konstanta nebo registr CL.

ROL cíl, kolik ROR cíl, kolik

Bitová rotace doleva/doprava operandu cíl o požadovaný počet bitů. Operand kolik je konstanta nebo registr CL.

RCL cíl, kolik RCR cíl, kolik

Bitová rotace doleva/doprava přes CF operandu cíl o požadovaný počet bitů. Operand kolik je konstanta nebo registr CL.

BT cíl, číslo

Zkopíruje do CF hodnotu bitu daného operandem číslo z operandu cíl.

BTR cíl, číslo

Zkopíruje do CF hodnotu bitu daného operandem číslo z operandu cíl a nastaví jej na nulu.

BTS cíl, číslo

Zkopíruje do CF hodnotu bitu daného operandem číslo z operandu cíl a nastaví jej na jedničku.

BTC cíl, číslo

Zkopíruje do CF hodnotu bitu daného operandem číslo z operandu cíl a provede jeho negaci.

SETcc cíl, číslo

Nastaví cíl na hodnotu 0/1 podle toho, zda je splněna požadovaná podmínka.

SHRD/SHLD cíl, zdroj, kolik

Provede nasunutí kolik bitů ze zdroje do cíle. Zdroj se nemění.

Aritmetické

Aritmetické instrukce nastavují cílový operand a nastavují i všechny příznakové bity v registru FLAGS.

ADD cíl, zdroj

Instrukce provede aritmetické sčítání cíl = cíl + zdroj

ADC cíl, zdroj

Instrukce provede aritmetické sčítání včetně CF cíl = cíl + zdroj + CF

SUB cíl, zdroj

Instrukce provede aritmetické odčítání cíl = cíl - zdroj

CMP cíl, zdroj

Instrukce provede aritmetické odčítání cíl = cíl - zdroj , ale neuloží výsledek

SBB cíl, zdroj

Instrukce provede aritmetické odčítání s výpůjčkou cíl = cíl – zdroj – CF

INC cíl

Instrukce provede zvýšení operandu o jedničku. Nemění CF.

DEC cíl

Instrukce provede snížení operandu o jedničku. Nemění CF.

NEG cíl

Instrukce provede změnu znaménka operandu.

MUL zdroj IMUL zdroj

Instrukce pro násobení dvou bezznaménkových/znaménkových čísel. Operandem zdroj se podle jeho velikosti (8, 16, 32) bitů násobí akumulátor (AL, AX, EAX) a výsledek se uloží do (AX, AX-DX, EAX-EDX)

DIV zdroj IDIV zdroj

Instrukce pro dělení dvou bezznaménkových/znaménkových čísel. Operandem zdroj se podle jeho velikosti (8, 16, 32) bitů dělí připravená hodnota v (AX, AX-DX, EAX-EDX) a výsledek se uloží do (AL, AX, EAX) a zbytek po dělení bude v (AH, DX, EDX)

CBW - Instrukce pro znaménkové rozšíření registru AL do AX. Používá se před znaménkovým dělením

CWD - Instrukce pro znaménkové rozšíření registru AX do AX-DX. Používá se před znaménkovým dělením

CDQ - Instrukce pro znaménkové rozšíření registru EAX do EAX-EDX. Používá se před znaménkovým dělením

CQO - Instrukce pro znaménkové rozšíření registru RAX do RAX-RDX. Používá se před znaménkovým dělením

Skokové instrukce nepodmíněné a podmíněné

JMP cíl

Provádění programu se přenese na adresu danou operandem cíl. Většinou jde o jméno návěští, kam se řízení přesouvá, ale může jít i o cíl daný adresou v registru nebo v paměti.

CALL cíl

Volání podprogramu. Stejně jako JMP, ale na vrchol zásobníku se uloží adresa instrukce následující za CALL.

RET N

Z vrcholu zásobníku se odebere adresa na kterou se následně předá řízení. Jde tedy o návrat z podprogramu. Volitelný operand N odstraní z vrcholu zásobníku dalších N bytů.

LOOP cíl

Vyjádřeno jazykem C se provede if (--ECX) goto cíl;. Jde o řízení cyklu, kde počet opakování je dán registrem ECX. Registr ECX se dekrementuje před vyhodnoceních podmínky.

LOOP/Z cíl

Vyjádřeno jazykem C se provede if (--ECX && ZF) goto cíl;.

LOOPNE/NZ cíl

Vyjádřeno jazykem C se provede if (--ECX && !ZF) goto cíl;.

JCXZ cíl

Provede skok na požadované místo jen pokud je registr ECX (CX) nulový.

Jcc cíl

Skupina podmíněných skoků. První podskupina řeší elementární podmíněné skoky:

- Instrukce JZ/E, JNZ/NE, JS, JNS, JC, JNC, JO, JNO testují přímo jednotlivé bity ve stavovém registru procesoru.

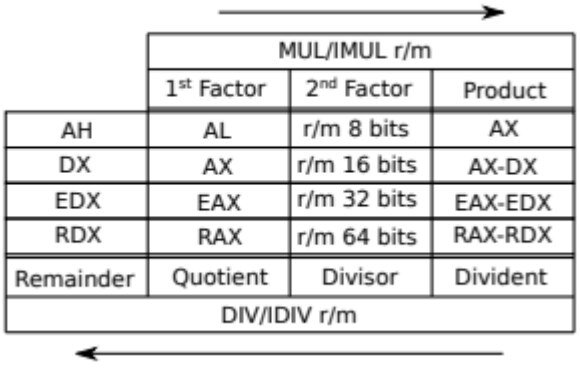
Druhá podskupina řeší porovnávání čísel:

- 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

Ve výše uvedených instrukcích mají písmena A-B-L-G-N-E svůj pevně daný význam.
Pro operace s bezznaménkovými operandy se používá A (above) a B (below). U operandů znaménkových se používá G (greater) a L (less). Pro negaci N (not) a pro rovnost E (equal).

(Pomocné a řídicí funkce: CLD – DF nastaví na nulu. STD – DF nastaví na jedničku. CLC – CF nastaví na nulu. STC – CF nastaví na jedničku. CMC – provede negaci (complement) CF.
NOP – prázdná instrukce)

Pomůcka při násobení a dělení:

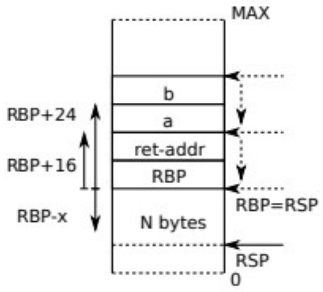


Volání funkcí s parametry

Pro předávání parametrů do funkcí se v 64bitovém režimu používá kombinace předávání parametrů přes registry i přes zásobník. Pro předávání celočíselných parametrů a ukazatelů se využívá pro prvních 6 parametrů zleva šestice registrů:

- RDI, RSI, RDX, RCX, R8 a R9

Prvních 8 parametrů zleva typu float/double se předává přes registry SSE: XMM0 až XMM7
Pro funkce s proměnným počtem parametrů se vyžaduje, aby byl v registru AL uveden počet parametrů předávaných přes registry XMMx.
Všechny další parametry se předávají přes zásobník. Všechny ukládaní hodnoty jsou 64bitové.



Obrázek 3: Zásobník v 64bitovém režimu po instrukci `enter N,0`

Na obrázku je ukázáno, jak by vypadal zásobník v případě, kdyby parametry *a* a *b*, byly sedmým a osmým celočíselným parametrem funkce. Přístup k lokálním proměnným zůstává nezměněn.

Návratové hodnoty funkcí

Způsoby předávání návratových hodnot jsou shodné v 64bitovém režimu, jako v režimu 32bitovém. Došlo jen k malému rozšíření a změnám při předávání hodnot float a double. V 64bitovém režimu se již nevyužívá pro výpočty jednotka FPU, ale výhradně SSE.

- 8 bitů – registr AL
- 16 bitů – registr AX
- 32 bitů – registr EAX
- 64 bitů – registr RAX
- 128 bitů – registry RAX-RDX
- float/double – registr XMM0

3. Technologie výroby číslicových obvodů.

(Znalosti toho tématu jsou důležité pro pochopení dalších témat, jako jsou např. mikropočítače a polovodičové paměti počítačů. V tomto tématu je důležitý hlavně rozdíl mezi bipolárním a unipolárním tranzistorem. Značky tranzistorů. Typy unipolárních tranzistorů.)
Hybridní – hybridní IO se skládají z tenké keramické destičky, na kterou jsou naneseny vodivé spoje, rezistory a přilepeny křemíkové destičky s diskrétními polovodičovými součástkami
Výhody: korundová podložka je výborný izolant, ztrátové teplo je účinně odváděno substrátem
Monolitické – základem pro výrobu je monokrystal z velmi čistého polovodiče, dokonale čistý monokrystal se poté nařeže na velmi malé plátky, na ty se pak difuzí přidávají různé příměsi, které v daných místech přetvářejí polovodičový materiál na materiály typu P nebo N (vznikají PN přechody). Ke kontaktům jsou poté přivařeny měděné drátky, které jsou vyvedeny na vývody (nožičky) IO. Celý obvod je pak zapozdřen do pouzdra.

Podle technologie výroby rozlišujeme číslicové obvody na hybridní a monolitické. Hybridní integrované obvody obsahují pasivní a aktivní součástky, které se připevní na jednu nosnou destičku, vzájemně propojí a zapouzdří. V monolitických integrovaných obvodech jsou všechny potřebné prvky soustředěny na jedné destičce polovodiče, nejběžněji křemíku. Technologickým postupem jsou na této destičce vytvořeny jak aktivní, tak pasivní prvky i vzájemné propoje součástek. Naprostá většina elektronických číslicových systému je vyráběna monolitickou technologií. Hybridní obvody se používají většinou jen u převodníku číslo-analog a analog-číslo. Dále se popisují monolitické integrované logické systémy a jejich vlastnosti.
Podle stupně integrace se rozlišuje:

- SSI (Small Scalle Integration) – malá integrace do 30 prvků
- MSI (Middle Scalle Integration) – střední integrace do 1000 prvků
- LSI (Large Scalle Integration) – velka integrace do 100 tisíc prvků
- VLSI (Very Large Scalle Integration) – do 10 miliónů prvků v pouzdře
- ULSI (Ultra Large Scalle Integration) – do 1 miliardy prvků v pouzdře
- GSI (Gigantic Scalle Integration) – nad 1 miliardu prvků v pouzdře

Tranzistor - třívrstvá polovodičová součástka, kterou tvoří dvojice přechodů PN. Tranzistory jsou základní aktivní součástky, které se používají jako zesilovače, spínače a invertory. Jsou základem všech dnešních integrovaných obvodů, jako např. procesorů, pamětí. Základní vlastností tranzistoru je schopnost zesilovat – malé změny napětí nebo proudu na vstupu mohou vyvolat velké změny napětí nebo proudu na výstupu.

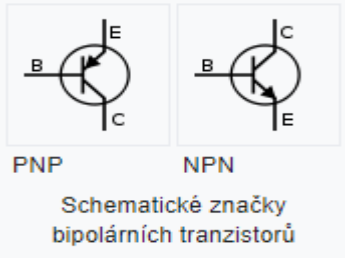
Bipolární technologie (přenosu náboje se účastní elektrony i díra)

Pro svou činnost používají bipolární tranzistory. Oproti unipolárním technologiím se jedná, až na některé výjimky, o rychlejší obvody, nedosahující tak velkého stupně integrace. Bipolární obvody mají také větší spotřebu a jsou levnější než unipolární technologie.

Technologie TTL (transistor-transistor-logic) (vycházející z použití bipolárních křemíkových tranzistorů)
- technologie pro vyrábění číslicových obvodů, používá napájení 5V
- tranzistor s vícenásobným emitorem tvoří logické funkce
- Napětí 0 V až 0,8 V se interpretuje jako logická 0, napětí 2 V až 5 V se interpretuje jako logická 1.

S-TTL (Schottky TTL) - zvyšuje rychlost přepínání stavů **L-TTL (Lower-power TTL)** – pomalejší se sníženým výkonem **H-TTL (High-power TTL)** – rychlejší se zvýšeným výkonem

Bipolární tranzistor (BJT – Bipolar Junction Transistor - řízeny proudem tekoucím do báze)



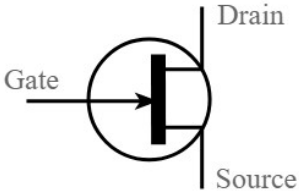
Schematické značky bipolárních tranzistorů

Součástka složená ze třech vrstev polovodičového materiálu. Prostřední oblast se nazývá báze (B), krajní emitor (E) a kolektor (C). Ke každé z oblastí je zapojen vývod. Při vhodném zapojení je velikost elektrického proudu tekoucího mezi emitorem a kolektorem řízena malými změnami proudu tekoucího mezi bází a emitorem. Bipolární tranzistory se používají jako zesilovače, spínače a invertory. Vyrábějí se jako samostatné součástky nebo jako prvky integrovaných obvodů. Ve složitých integrovaných obvodech však převládá používání unipolárních tranzistorů. (špatně odvádí teplo)

Unipolární technologie

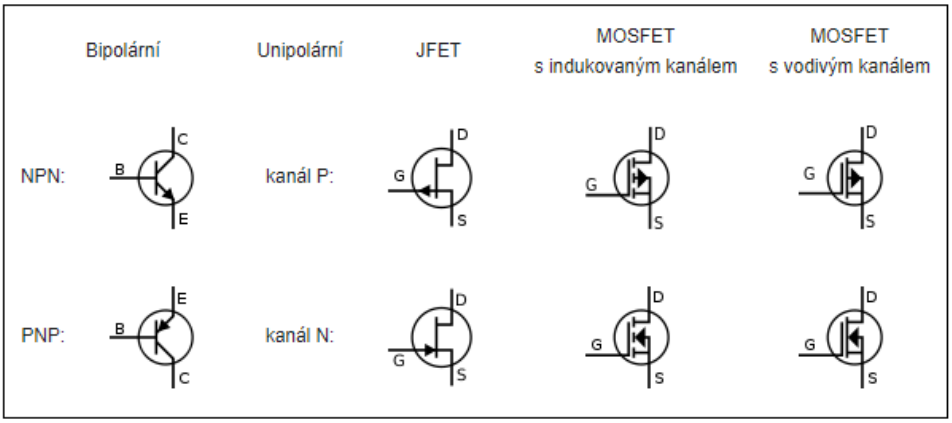
U těchto technologií se přenosu náboje účastní (na rozdíl od bipolárních technologií) pouze jeden druh nosičů náboje, a to buď elektrony nebo díry.

Unipolární tranzistor (FET – Field Effect Transistor - řízeny napětím na řídící elektrodě (gate) – tranzistor řízený elektrickým polem (napětím))



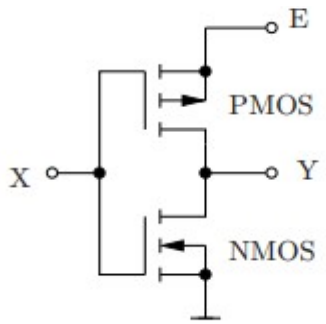
Princip spočívá v řízení proudu mezi dvěma elektrodami pomocí elektrody třetí. Vedení proudu se účastní pouze většinové nosiče náboje (elektrony v polovodiči N a díry v polovodiči P), proto se mu říká unipolární.
Pro velký vstupní odpor se těmto tranzistorům také říká tranzistory řízené elektrickým polem (FET, Field-Effect Transistors). Velký vstupní odpor je velkou výhodou unipolárních tranzistorů oproti bipolárním tranzistorům, jejichž malý vstupní odpor se nepříznivě projevuje při zesilování signálů ze zdrojů s velkým vnitřním odporem. Vstupním obvodem unipolárního tranzistoru tak neteče proud a je řízen napětím. Řídící elektrodou teče buď jen malý proud ekvivalentní proudu diody v závěrném směru nebo jí neteče prakticky žádný proud.

JFET (Junction FET) – nejjednodušší typ unipolárního tranzistoru, Jedná se o polovodičovou součástku se třemi elektrodami označovanými G (Gate), S (Source), a D (Drain), kterou lze používat jako elektronicky řízený spínač, zesilovač nebo napětím řízený rezistor. Bipolární tranzistory jsou řízené proudem bází, a pokud bází neteče žádný proud, jsou zavřené. Naproti tomu JFET jsou řízené napětím na elektrodě G, a není-li mezi elektrodami G a S žádný rozdíl potenciálů, je JFET otevřený, takže polovodičovým kanálem mezi elektrodami S a D může téct elektrický náboj. Přivedením předpětí vhodné polarity na elektrodu G dochází k přivírání kanálu – JFET bude klást průtoku proudu určitý odpor, takže proud kanálem mezi elektrodami S a D bude menší; při dosažení závěrného napětí poklesne proud protékající kanálem na nulu.
MESFET – (Metal Semiconductor FET) Řídící elektroda je tvořena závěrně polarizovaným přechodem kov–polokov.
MOSFET – (Metal Oxide Semiconductor FET) Řídící elektroda je izolována od zbytku tranzistoru oxidem. Jejich výkonnostní varianty mají mezi Drain a Body takzvanou body diodu, která jím pomáhá zvládat napěťové špičky opačného napětí způsobené rychlým rozpojováním induktoru např. motoru (pro jehož řízení se často používají)
MISFET – (Metal Insulated Semiconductor FET) Obecný název pro tranzistor s izolovanou řídící elektrodou. Izolantem nemusí být jen oxid (např. nitrid…).



MOSFET s obohacováním kanálu
Princip činnosti spočívá ve vytváření a rozšiřování vodivého kanálu mezi emitorem S a kolektorem D vlivem příčného elektrického pole vyvolaného přivedením napětí na hradlo G. Tento typ MOSFETu, ať už s kanálem P nebo N, díky vysokému výstupnímu odporu, malé spotřebě, značné odolnosti proti rušivým signálům a výboným spínacím vlastnostem, hraje primární úlohu v technice číslicových integrovaných obvodů.

CMOS



Obrázek 9: Zapojení invertoru technologie CMOS

Při signálu logické 1 na vstupu X (kladná logika) je vodivý (zapnutý) tranzistor NMOS a tranzistor PMOS je vypnutý. Na výstupu Y je logická 0. Při úrovni logické 0 na vstupu X je tranzistor PMOS zapnutý (vodivý) a tranzistor NMOS je vypnutý. Na výstupu Y je úroveň logické 1.
Tedy tranzistory fungují jako spínače, které přepínají výstup buď na napájecí napětí E nebo k zemi. Takže pokud nezatěžujeme výstup takového obvodu, je jeho spotřeba v klidovém stavu prakticky nulová. Výstup obvodu má relativně malou impedanci v obou stavech (řádově stovky ohmů).
Obvody CMOS mohou mít napájecí napětí v rozmezí 3 až 16 V .
Díky extrémně nízkému příkonu, dobré šumové imunitě (45% napájecího napětí), slučitelnosti s obvody TTL, širokému rozmezí napájecího napětí, velkému rozsahu pracovních teplot a velkému logickému zisku, došlo k obrovskému rozšíření obvodů CMOS a k jejich převládnutí na trhu. Tato technologie je dosud nejpoužívanější technologií ze všech.
Tyto obvody se používají pro výrobu monolitických mikroprocesorů, pamětí a dalších prvků obvodů LSI, VLSI a ULSI, ale i pro výrobu logických členů obvodů SSI a MSI.

Technologie výroby pro paměti s pevným obsahem.

Technologie SOS (Silicon On Sapphire)

- označení skupiny, které základem čipu je destička syntetického safíru.
- Výhody safírové podložky: zmenšení parazitních kapacit až 3x
- Mikroprocesory a paměti RWM využívají tuto technologii
- drahé na výrobu (vysoká cena safíru 5x větší než křemík, proto se začala vyrábět SOI technologie)

Technologie SOI (Silicon On Insulator)

- izolantem je křemíková destička pokrytá oxidem křemičitým (SiO_2), na němž jsou vytvořeny ostrůvky polovodičových struktur, které jsou od sebe dokonale izolované
- oxidem křemičitý je dobrý izolant a jeho výroba je poměrně jednoduchá (zahřívání křemíku v oxidační atmosféře), proto je levnější než SOS technologie

Technologie FAMOS
Technika plovoucího hradla (gate) s lavinovou injekcí nosičů - (Floating– gate Avalanche–injection MOS)
- vznikla u firmy Intel
- nejrozšířenější technologie pro výrobu elektricky programovatelným pamětí EPROM
- základem paměťové buňky je tranzistor MOS s řídicí elektrodou, která není k ničemu připojena, protože je izolovaná ze všech stran oxidem křemičitým
- Pro mazání informace z paměti se používá ultrafialové ionizující záření (o vlnové délce 253μm) Tím se překonají bariéry v opačném směru. Potenciály hradla a emitoru se tak vyrovnávají, zruší (vymaže) se obsah paměti FAMOS a tranzistor FAMOS se uvede do původního vypnutého (nevodivého) stavu. Tím je paměť opět připravena k dalšímu programování.
- Při každém mazání informace dochází k mírné degradaci parametrů paměťové buňky FAMOS
- (Pokud však je mazání šetrné (např. studeným utrafialovým zářením), nevybočí parametry pamětí EPROM z tolerancí ani po několika desítkách cyklů mazaní–programování.)

Technologie FLOTOX
Paměťová polovodičová struktura FLOTOX (FLOating–gate Tunnel OXide cell)
- modifikace technologie FAMOS
- používána pro tvorbu paměťových buněk mikroelektronických vymazatelných a programovatelných pevných pamětí EEPROM (Electrically Erasable and Programmable ROM)

Technologie CCD
Pro součástky vyrobené technologií CCD (Charge Coupled Devices) není typická zesilovací činnost základních obvodových členů, ale přenos náboje na parazitních kapacitách soustavou elektrod vytvořených na strukturách MOS.
- Na tomto principu se vytváří posuvné registry, ale sekvenční paměti z nich vyrobené nejsou energeticky nezávislé, proto se v mikropočítačích neuplatňují
- rozsáhlejší použití mají v analogové technice jako paměti ve snímačích obrazu pro televizi a v monolitických plochých displejích
- Výhody: malá spotřeba energie a malé rozměry

Technologie BiCMOS
- rychlejší struktura než CMOS
- využití pro paměti a hradlová pole
- umožňuje zvýšeni rychlosti
- využívána firmou Intel pro výrobu mikroprocesorů řady Pentium

4. Komunikace s periférierni
Sběrnice
Svazek vodičů, který slouží pro přenos dat mezi dvěma nebo více zařízeními (např. mezi mikroprocesorem a ostatními částmi počítače). **Sběrnice** má za účel zajistit přenos dat a řídicích povelů mezi dvěma a více elektronickými zařízeními. Přenos dat na sběrnici se řídí stanoveným protokolem. V případě modulární architektury elektronického zařízení nebo počítače je sběrnice po mechanické stránce vybavena konektory uzpůsobenými pro připojení modulů. Sběrnice dělíme na adresovou, řídící a datovou.
Mezi nejčastější účastníky každého přenosu, uskutečněném na sběrnici, bývá téměř vždy mikroprocesor, který v těchto případech rovněž určuje další údaje přenosu (např. Směr přenosu, druhého účastníka, kdy má být informace na sběrnici platná a další nutné údaje).
(způsob přenosu dat: paralelní – přenáší se více bitů najednou, sériový – přenáší se po jednom vodiči bit po bitu)

Adresní sběrnice
Pokud chce mikroprocesor data číst nebo je zapisovat, musí nějakým způsobem sdělit místo čtení i zápisu. Toto místo je identifikováno adresou, která je přenášena po adresové sběrnici. Zdrojem této informace je mikroprocesor.
Počet bitů adresové sběrnice, neboli počet vodičů, odpovídá počtu bitů adresy. Mikroprocesor vytváří tuto adresu a určuje tak využitelný adresový prostor. Jako příklad nám poslouží např. šestnáctibitová adresová sběrnice, která maximálně adresuje $2^{16} = 65536$ adres.
Univerzální mikroprocesory mívají dva adresové prostory. Jeden adresový prostor pro adresování paměti a druhý pro adresování vstupů a výstupů. Každý blok komunikující s mikroprocesorem musí být umístěn v některém z těchto dvou prostorů. Tyto prostory nejsou totožné a rozlišení těchto adresovatelných prostorů zajišťuje řídící sběrnice.

Řídicí sběrnice

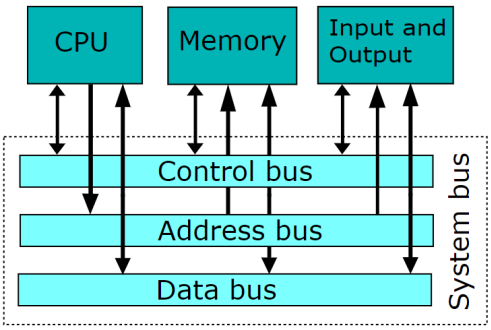
Část počítačové sběrnice, která slouží pro přenos řídicích signálů. Pracuje v součinnosti s adresní a datovou sběrnicí. (Počet vodičů řídicí sběrnice je různý.)

Řídící sběrnice je souhrn jednotlivých signálů aktivních v různých časových okamžicích s různým významem, které mívají různé zdroje. Některé signály jsou generovány mikroprocesorem, některé mohou být ovlivňovány jinými bloky. K jednotlivým blokům jsou pak přivedeny pouze signály jim patřící. Mezi nejčastější signály řídící sběrnice patří například:

- Signál **RESET**, jež je vybaven každý mikroprocesor. Aktivován uživatelem nebo jiným přídavným obvodem. Tento signál uvede mikroprocesor do jeho výchozího stavu.
- Signál **Memory Read (MR)**, zabezpečuje časování čtení z pamětí či jiných bloků do mikroprocesoru nebo počítače
- Signál **Memory Write (MW)**, zabezpečuje časování zápisu do pamětí či jiných bloků do mikroprocesoru nebo počítače
- Signály I**nput/Output Read/Write (IOR/IOW)**, slouží pro čtení z, nebo zápis do zařízení
- Signál **READY**, určuje připravenost obvodu

Datová sběrnice

Slouží pro přenos všech dat v počítači. Data jsou vždy přenášena mezi dvěma bloky počítače. Typickým příkladem je přenos dat z paměti do mikroprocesoru. Mikroprocesor se účastní, až na výjimku jako přijímač a vysílač, všech přenosů v počítači. Je nutné, aby v jakémkoliv okamžiku byl aktivní pouze jeden vysílač, (budič sběrnice). Při nedodržení této podmínky by na datové sběrnici došlo k neurčitosti signálu, v horším případě pak ke zničení jednoho nebo obou vysílacích obvodů (pro odstranění rizika se používají třístavové budiče sběrnice – oddělují od sběrnice). Mezi nejdůležitější parametry datové sběrnice patří její šířka nebo počet bitů a časování. Šířka datové sběrnice udává, kolik bitů je schopno se přenést najednou a patří mezi parametry výrazně ovlivňující rychlost komunikace. Ovšem nemá význam, aby bitová šířka datové sběrnice byla větší než bitová šířka mikroprocesoru.



Sdílený a oddělený adresní prostor pro paměť a periférie.

Přenosy dat na sběrnicích se mohou uskutečnit řízením technickými prostředky nebo programovým řízením.

Řízení technickými prostředky

- na řízení se podílí mikroprocesor jen částečně nebo je úplně vyřazen
- přenos je řízen řadičem přímého přístupu do paměti (DMA)
- využívá se pro přenosy velkých bloků dat (např. přenos dat z přídavných zařízení do paměti RAM)

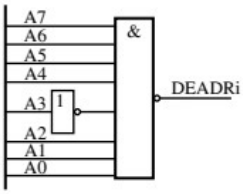
Programové řízení

- na přenosu se podílí výhradně mikroprocesor (postupně provádí instrukce účelně seřazené do segmentu řídícího programu)
- výhody: levná realizace, snadná změna algoritmu přenosu změnou příslušného segmentu řídícího programu
- nevýhody: vlastnost pomalejších mikroprocesorů => celkově pomalý přenos

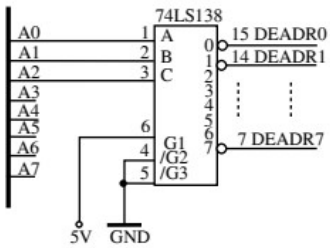
Přenos dat s programovým řízením je obecně přenosem mezi dvěma registry. U mikroprocesoru probíhá velmi často mezi některými vnitřními registry, případně mezi vnitřním registrem a buňkou vnější paměti. Přenosy mezi vnitřními a vnějšími registry lze rozdělit na přenosy paměťové a vstupně/výstupní. To znamená, že umožňuje-li architektura mikroprocesoru oba tyto přenosy, mívají pak tyto dva adresové prostory (prostor pro adresování paměti a vstupů/výstupů). Výběr prostoru je pak určen příslušnými signály pro určitý prostor.

Umístění periferních zařízení do některého z těchto prostorů nazýváme mapováním. Mezi výhody mapování patří mapování periferií do V/V prostoru a spočívá v rychlejším přístupu do tohoto prostoru, zpravidla instrukcemi vstupu (IN) a výstupu (OUT). Výhoda mapování do paměťového prostoru spočívá ve větším množství použitelných instrukcí pro přenos dat (k dispozici jsou všechny instrukce používané pro práci s pamětmi). Paměťový prostor bývá obsazen více jak jednou fyzickou pamětí nebo periferním zařízením, a proto je nutné při přenosech dat s mikroprocesorem rozhodnout, které zařízení je ke komunikaci určeno. Tímto úkolem je právě pověřen adresový dekodér.

Adresový dekodér



Obrázek 2: Princip úplného dekodéru adresy



Obrázek 3: Princip neúplného dekodéru adresy

- rozhoduje, které zařízení bude s pamětí komunikovat
- jeho výstupy jsou signály CS (Chip Select) pro jednotlivé obvody
- signál CS připojuje daný obvod k datové sběrnici tak, že jeho sběrnice přepne ze stavu vysoké impedance do aktivního stavu
- Může být stavěn jako dekodér pro:
  - úplné dekódování adresy
  - neúplné dekódování adresy
  - lineární přiřazování adresy
  - univerzální přiřazení dekódovaných adres

wiki: vstupem jsou dva nebo více bitů adresní sběrnice, a výstupem je několik vodičů výběr zařízení (anglicky Device Selector), které aktivují vybraný paměťový nebo vstupně/výstupní obvod[1].

Adresový dekodér s úplným dekódováním adresy
Při úplném dekódování je jisté unikátní adrese ADRi přiřazen pouze jeden signál DEADRi a obráceně, kde jistému signálu DEADRi odpovídá pouze jedna adresa ADRi. Realizace je pomocí hradla AND.

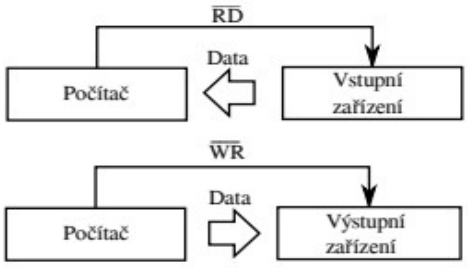
Adresový dekodér s neúplným dekódováním adresy
Pro adresování dekodéru s neúplným dekódováním adresy nejsou uvažovány některé řády adresy s tím, že jistému signálu DEADRi přísluší adresový prostor ADRi, kde i=1,2,3... Takto postavený adresový dekodér je sice levnější, ale vyžaduje pozorné přiřazování adres k jednotlivým adresovaným obvodům. S těmito dekodéry se můžete velmi často potkat při přidělování paměťových prostorů k jednotlivým typům pamětí nebo též při přidělování k přídavným periferiím v počítačovém systému.

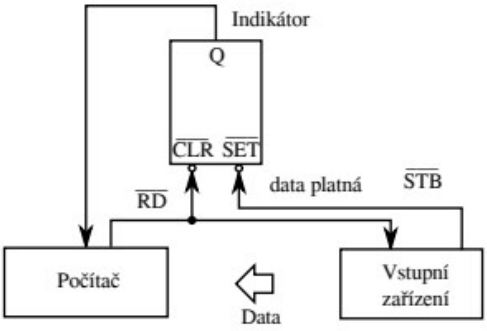
Adresový dekodér s lineárním přiřazením adresy
Jednotlivé řády adresy těchto dekodérů jsou pokládány přímo za výstupní výběrové signály dekodéru DEADRi = ADRi. Je to tedy nejlevnější adresový dekodér. Jeho nevýhodou je možnost připojit pouze *m* přídavných zařízení u adresy s *n* řády.

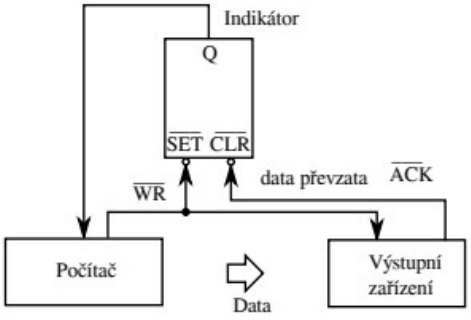
Komunikace metodou vstupně výstupních bran (I/O portů)

Obvod, který zprostředkovává předávání dat mezi sběrnicí počítače a periferním zařízením počítače. Dvě možnosti použití brány s/bez pamětí. Oba typy bran jsou výkonové zesilovače (budiče) jednosměrně nebo obousměrně řízené. Základem bran bývá *záchytný registr (latch)* s třístavovým vstupem.

- **Technika nepodmíněného vstupu a výstupu dat**
  - Při vstupu vyšle počítač signál $\overline{RD}$, čímž přikáže vstupnímu zařízení předat data do vstupní brány počítače. Při výstupu počítač vyšle současně data i signál $\overline{WR}$ a výstupní zařízení převezme data. Tento způsob je mimořádně jednoduchý a předpokládá stálou připravenost periferního zařízení komunikovat.
- **Technika podmíněného vstupu**
  - Jsou-li poskytována platná data ze vstupního zařízení, pak se za pomocí snímacího impulsu STB (strobe) nastaví hodnota $Q = 1$. Tento stav Q je označován jako indikátor (angl. flag). Pokud je tento stav, kdy je $Q = 1$ platný, jsou předány data počítačem za pomocí impulsu $\overline{RD}$ a po uskutečnění tohoto přenosu je tento indikátor vynulován.
- **Technika podmíněného výstupu**
  - Počítač vyšle signál $\overline{WR}$ pro přepis dat do výstupního zařízení a následnému nastavení indikátoru. Výstupní zařízení po převzetí dat impulsem ACK indikátor nastaví na nulovou hodnotu. Touto hodnotou je počítači sděleno, že může vyslat další data.

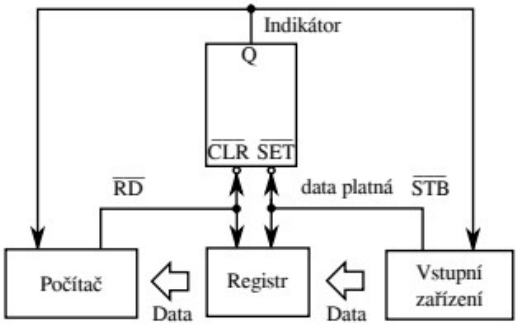




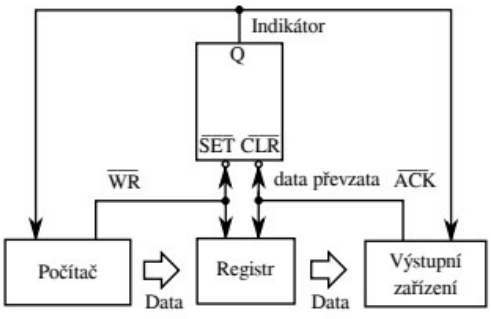


V obou případech jde o **neúplný (jednosměrný korespondenční) režim**, kdy pomocí indikátoru jsou sdělovány informace pouze počítači o zahájení nebo ukončení přenosu a kdy vysílač dat je povinen data udržovat.

Technika úplného (obousměrného korespondenčního) režimu
Využívá vyrovnávací paměti (registru) a klopný obvod pracující jako semafor (jeho stav je je testován vysílačem i přijímačem dat). Přichází zde i v úvahu možnost vzájemného blokování (interlock).
U vstupu dat semafor informuje procesor o připravenosti dat ve vyrovnávací paměti. Pro periférii tento semafor představuje informaci, zda je možno do vyrovnávací paměti zaslat další data, či zda obsah paměti nebyl ještě přečten. U výstupu dat je význam signálu semaforu pro procesor a periférii opačný (než pro vstup).



Obrázek 8: Technika vstupu dat s vyrovnávací pamětí



Obrázek 9: Technika výstupu dat s vyrovnávací pamětí

Handshake - automatizované vyjednávání, jehož úkolem je nastavit parametry komunikačního kanálu mezi dvěma subjekty před zahájením vlastní komunikace.

Řízení komunikace

Existují dva případy zahájení komunikace počítače nebo mikroprocesoru s periferiemi:

- **Zahájení komunikace z iniciativy programu** – Jde o případ, kdy počátečním příkazem k zahájení komunikace je instrukce probíhajícího programu. Příkladem může být například algoritmus programu, který potřebuje přijímat z, resp. vysílat do okolí informace. Nutnou podmínkou je schopnost dané periferie přijmout resp. vyslat informaci v daný okamžik nebo musí informaci v okamžik určený počítačem vystavit. Příkladem těchto periferií mohou být stavové spínače nebo sdělovače.
- **Zahájení komunikace z iniciativy periferie** – Zahájení komunikace z iniciativy periferie nastává v případě, že jistá periferie chce poslat resp. přijmout svou informaci z resp. do počítače. Zde však nastává problém v navázání komunikace, neboť se počítač může nacházet v činnosti, kdy ihned nemůže s periferií komunikovat. Počítač se však nějakým způsobem musí dozvědět, že je vyžadována komunikace a která periferie je iniciátorem výzvy.

V případě, že je komunikace **aktivovaná periferií**, ne počítačem, lze postupovat několika způsoby:

- Obvodovým řešením daného periferního zařízení. Určitou operaci může uskutečnit tak, aniž by o tom musel vlastní počítač vědět. Toto bývá řešeno pomocí obvodů nízké a střední integrace.
- Pro inicializaci určité operace zařízením použijeme příznakový bit (Flag) nebo skupinu příznakových bitů. Tato hodnota je čtena počítačem. Pokud je hodnota rovna 1, může iniciovat určitou operaci. Pomocí příznakových bitů oznamuje periferní zařízení svůj stav (připraven, zaneprázdněn, porucha atd.). Skupina příznakových bitů vytváří stavové slovo (Status Word). Počítač tedy nejprve přečte stavové slovo dodané periferií a na základě analýzy tohoto slova rozhodne o další činnosti. Tomuto zařízení budeme říkat **programové řízení**.
- Zařízení, které chce inicializovat nějakou operaci, pomocí speciálního signálu procesoru přeruší právě probíhající program. Procesor přeruší svou činnost a vykoná požadovanou akci komunikace. Poté se vrátí tam, kde byl procesor přerušen a pokračuje v původní činnosti. Pro obsluhu komunikace mezi více zařízeními tímto způsobem, musí mít mikroprocesor nebo počítač k dispozici obsáhlejší **systém přerušení**.
- Pokud operace, kterou chce periferní zařízení inicializovat, spočívá v přenosu bloku dat z periferního zařízení do paměti počítače nebo naopak, může se tato operace uskutečnit pomocí **přímého přístupu do paměti (DMA).**

Programové řízení komunikace

Využívají se zde instrukce pro vstup a výstup ve spojení s instrukcemi umožňující testování logických proměnných a instrukcí skoků.
Použití principu programového řízení je velmi jednoduché u těch počítačů, které mají přímo vnější vstup příznakového (stavového) bitu a instrukce podmíněných skoků, umožňující větvení programu podle hodnoty příznakového bitu.
Organizaci programového vybavení obsluhy komunikace musíme přizpůsobit vzhledem k našemu požadavku na včasné zjištění hodnot stavových bitů. Typickým případem jsou data z klávesnice, kdy informace nevyžadují příliš častou pozornost mikroprocesoru.
Program určitých zařízení lze sestavit tak, že vykonává opakovaně určitou posloupnost instrukcí. Tato posloupnost je vzhledem k rychlosti dnešních mikroprocesorů a počítačů natolik krátká, že by nemělo činit problémy snímat stavový bit mnohem častěji např. každých 10 ms. Pokud tato doba trvání je delší, můžeme toto testování zařadit vícekrát (např. voláním podprogramu) do posloupnosti instrukcí algoritmu programu.
Jisté případy mohou vzniknout, kdy mikroprocesor při určité fázi komunikace s periferním zařízením nastaví stavový bit a bude v určité krátké době vyžadovat odezvu na tento stav, jsou řešeny čekací smyčkou. Tím je míněno, že počítač testuje daný stavový bit, dokud nebude nastaven.
Výhody: úspora obvodu
Nevýhody: pomalé, počítač je zatěžován periodickým testováním stavů, i v případech kdy nedochází k přenosu

Řízení komunikace pomocí přerušení

Systém přerušení mikroprocesoru usnadňuje programové řízení spolupráce s periferními zařízeními především ve fázi zjišťování žádosti o obsluhu. Jakmile periferní zařízení požaduje od počítače obsluhu (např. přenos dat, různá hlášení atd.), aktivuje přerušovací signál vstupující do mikroprocesoru. Po zjištění žádosti o obsluhu procesor přeruší právě probíhající program a přejde do obslužného programu. Po jeho dokončení se procesor vrátí do původního místa v programu, kde byl přerušen a pokračuje v dalším provádění hlavního programu, v kterém byl přerušen.
U tohoto řízení však nastává problém při obsluze více zařízení a je tak potřeba rozsáhlejšího přerušovacího systému. Informace, které zařízení žádalo o obsluhu je nejčastěji poskytnuta ve formě *vektoru přerušení*, což je adresa příslušného obslužného programu.

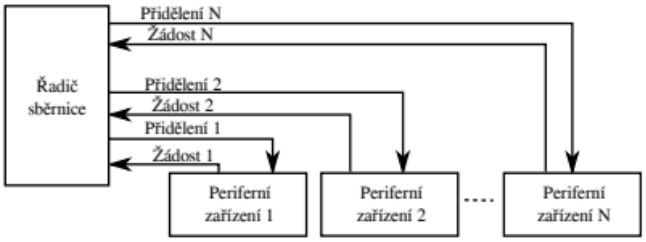
Některé možnosti identifikace zařízení žádajícího o obsluhu:
1. Požadavky na obsluhu jednotlivých zařízení jsou logicky sečteny a procesor přečte stavové slovo obsahující identifikační znak zařízení žádající o obsluhu. Pokud o obsluhu žádá více jak jedno zařízení najednou, musíme toto přijetí požadavků programově ošetřit.
2. Požadavky na obsluhu zařízení jsou opět logicky sečteny a signál potvrzení žádosti od mikroprocesoru je sériově veden přes všechna periferní zařízení. Priorita je tedy dána zařazením jednotlivých zařízení vůči tomuto signálu. Pokud nastane žádost o přerušení, procesor vyšle potvrzující signál nejprve prvnímu zařízení (zařízení s nejvyšší prioritou). Pokud toto zařízení nežádalo o přerušení, je tento signál propuštěn dál. Jakmile signál dojde k zařízení, jež požadavek o přerušení vydalo, je nutné zabránit dalšímu průchodu tohoto signálu k dalším zařízením. Procesoru předáme *vektor přerušení*, tj. adresu, na které se nachází obslužný podprogram. Tento způsob již vyžaduje přídavné řešení technickými prostředky
3. Další možností je použití *řadiče přerušení*, jehož princip spočívá ve vyslání identifikačního znaku daného zařízení až po přijetí žádosti o přerušení. Dále pak v přiřazování priority a její dynamické změny jednotlivým zařízením při více žádostech najednou a také zajišťuje maskování zdrojů.

Žádosti o přerušení jsou nejprve porovnány s registrem masky, který rozhoduje o jejich maskování. Tyto žádosti jsou nejprve uloženy v registru žádosti o přerušení. Poté jsou přijaty tyto požadavky na přerušení blokem priorit, rozřazeny podle jejich priority a vybrán jeden požadavek s nejvyšší prioritou. Tento požadavek s nejvyšší prioritou, který pošle procesoru kompletní informace o adrese, na které je program obsluhy přerušení. Tento program je potřeba před jeho činností naprogramovat. Naprogramování spočívá v nastavení priority jednotlivých zdrojů, masky a v neposlední řadě i adresy obsluhy přerušení příslušným zdrojům.

Priority

- umožňují procesoru určit pořadí jednotlivých žádostí
- mezi nejznámější metody vyhodnocování priority patří:
  - **Samostatné žádosti**

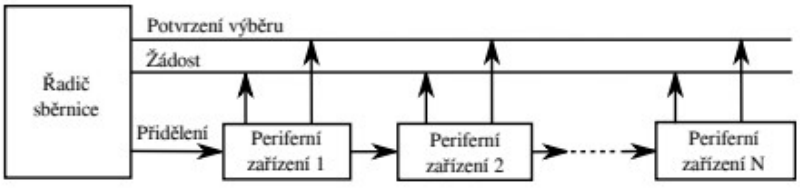
    Žádosti se posílají přímo po vodičích do procesoru. Ke každému vodiči „žádost“ je přiřazen jeden vodič s označením „přidělení“. Tímto jednoznačným přiřazením jednoho vodiče každé jednotce je dána informace procesoru, která jednotka žádala o sběrnici. Vyhodnocení priority se může provést jednoduchým prioritním dekodérem. Jednotlivé priority jsou přiděleny pevně, což umožňuje kombinačnímu obvodu zpracování přicházejících žádostí s minimálním zpožděním. Další možností je vyhodnotit přijaté žádosti programem, který cyklicky prohledává registr, a v kterém jsou tyto žádosti zapsány. Výhodou použití programu je možnost jeho snadné změny.

  - **Zřetězení**

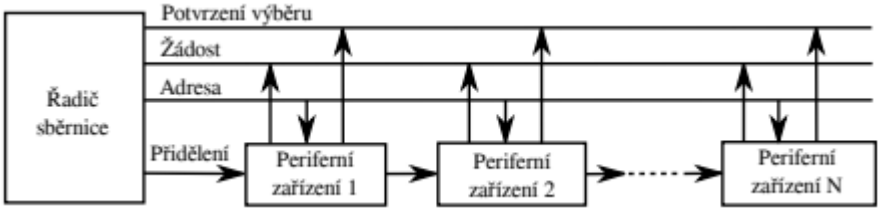
    Požadavky na obsluhu zařízení vysílají všechny jednotky po jediném vodiči, takže procesor dostává žádost anonymně a sám ji nemůže vyhodnotit. Tyto požadavky jsou opět logicky sečteny a signál potvrzení žádosti od mikroprocesoru je sériově veden přes všechna periferní zařízení. Pořadí, v němž jsou jednotky tímto vodičem propojeny, odpovídá pořadí priorit (první jednotka, následující přímo za procesorem, má nejvyšší prioritu, poslední jednotka má nejnižší prioritu).

    Po příchodu přidělovacího signálu záleží, zda jednotka žádala o přidělovací signál. Pokud žádala, zablokuje přidělovací signál a nepustí jej dál a současně potvrdí výběr. Potvrzení výběru učiní vygenerováním své adresy, kterou vyšle do procesoru. Druhou možností je vyslání jednobitového signálu po zvláštním vodiči, kterým je každá jednotka spojena s procesorem. Pokud je sběrnice přidělena určité jednotce, pak všechny jednotky za ní následující musí čekat. Jednotka, která nežádala o přidělení, propustí přidělovací signál beze změny a nijak na něj nereaguje.

  - **Cyklické výzvy**

    Třetí metodou vyhodnocování priorit jsou cyklické výzvy (polling). Jednotky vysílají své žádosti opět po jednom vodiči sběrnicového typu, takže procesor přijme žádost jako anonymní. Namísto jednoho vodiče, který signalizuje přidělení sběrnice, je však použita skupina vodičů „adresa jednotky“, schopná přenášet adresu kterékoliv z připojených jednotek. Když procesor přijme žádost o přidělení sběrnice (a je-li připraven této žádosti vyhovět), začne po adresových vodičích vysílat postupně adresy všech jednotek v předem stanoveném pořadí. Jakmile jednotka, která žádala o přidělení, rozpozná na adresových vodičích svou adresu, reaguje na ni jako signál „přiděleno“, vyšle do procesoru signál „potvrzení výběru“ a stane se řídící jednotkou.

Řízení přenosu dat pomocí DMA řadiče (přímý přístup do paměti)
V počítači existují události, na které nelze reagovat jinak než přerušením. Princip operace spočívá v prostém přesunu informací z periferního zařízení do hlavní paměti nebo naopak. To lze realizovat výhodněji formou **přímého přístupu do paměti** označované zkratkou **DMA** (z anglického výrazu Direct memory access).
Princip DMA: Spočívá v přímém přesunu informací bez účasti procesoru mezi vyrovnávacím registrem periferního zařízení a hlavní pamětí. Tedy procesor se vůbec neúčastní tohoto přesunu a tudíž nemusí ukončovat své aktuálně běžící programy, aniž by je přerušoval. Jedinou nutnou podmínkou je „uvolnění“ sběrnice pro procesor, to znamená, že procesor na dobu přesunu přepne všechny budiče sběrnic do třetího (vysokoimpedančního) stavu. Sběrnice nemohou zůstat po dobu přesunu bez řízení. A právě tomu zabraňuje existující další blok, který je schopný generovat adresu a určovat okamžiky přesunu dat po datové sběrnici. Tento blok se označuje jako blok DMA, někdy též označovaný kanál DMA.
V tomto bloku pro přímý přístup do paměti jsou určeny tři registry pro styk se sběrnicemi označené jako:

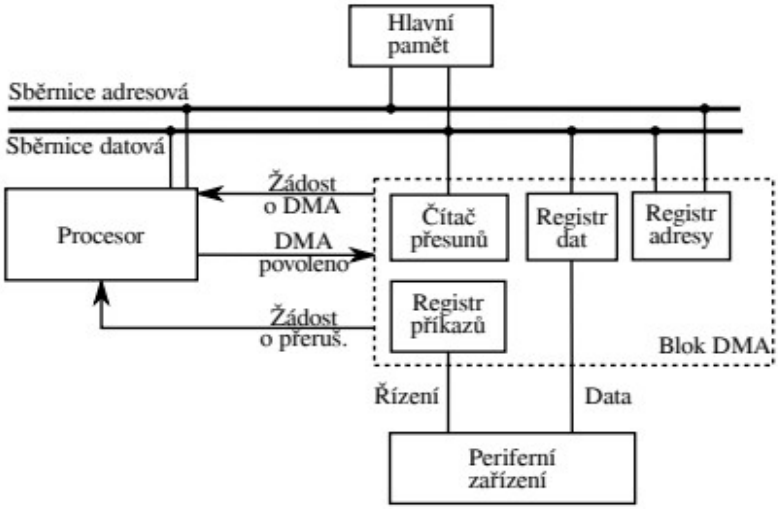
- **registr dat** – obsahuje slovo, jež má být přesunuto z periferního zařízení do hlavní paměti nebo naopak
- **registr adresy** – slouží pro uchování adresy hlavní paměti, což je adresa, na kterou bude zapsáno slovo nebo ze které bude dané slovo přečteno
- **čítač přesunů** – požadovaný počet slov, které mají být ještě přesunuty v rámci jednoho spojení mezi periferním zařízením a hlavní pamětí

Blok DMA může pracovat ve dvou režimech:

- přesouvání dat mezi periferním zařízením a hlavní pamětí jednotlivě
- přesouvání dat v blocích

Celá operace přímého přístupu do paměti obvykle probíhá v několika krocích:
1. Naprogramování procesorem bloku DMA.
2. Blok DMA spustí periferní zařízení a čeká, až bude toto periferní zařízení připraveno buď data přijmout nebo vyslat, jakmile periferní zařízení oznámí bloku DMA, že je připraveno žádat blok DMA o přímý přístup do paměti.
3. Procesor nejprve dokončí strojový cyklus a poté reaguje na žádost o DMA. Přímý přístup k paměti se pak uskutečňuje průběžně během normální činnosti procesoru tak, že blok DMA vysílá data synchronně (s fází hodin Φ_1, zatímco procesor používá paměť vždy ve fázi Φ_2).
4. K přímému přístupu do paměti může dojít v okamžiku, kdy vyšle procesor vybrané jednotce signál ACK a uvolní sběrnice. Vybraná jednotka pak vyšle na adresovou sběrnici obsah svého registru adresy, na datovou sběrnici obsah svého registru dat a čeká na provedení jednoho cyklu paměti. Pak zvětší obsah registru adresy o jedničku a zmenší obsah čítače přesunů. Není-li obsah čítače přesunů dosud nulový, testuje, zda periferní zařízení již přesunulo nové slovo do registru dat. Pokud nebylo přesunuto nové slovo, ukončí se dočasně přesun dat a přestane vysílat žádost o DMA a tím předá řízení procesoru.
5. Procesor dále pokračuje v provádění svého programu až do okamžiku, kdy některý blok DMA vyšle novou žádost o DMA, signalizující připravenost dat z periferních zařízení v registru dat.
6. Pokud je obsah čítače přesunů nulový, blok DMA končí celý přesun a uvolní sběrnice. Navíc může vyslat žádost o přerušení, čímž si vyžádá programový zásah procesoru, který spočívá v novém naprogramování jeho vnitřních registrů.

Shrnutí: Řadič DMA adresuje hlavní paměť a synchronizuje činnost mezi sběrnicí a periferním zařízením, zatímco data jdou mimo něj.



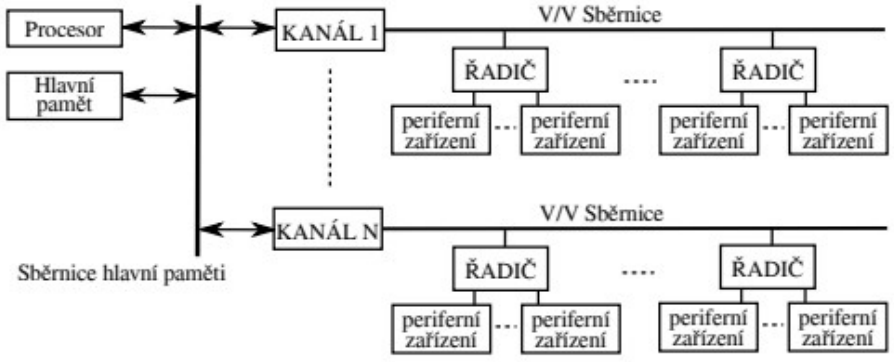
Kanálová architektura

Kanál – speciální procesor určený ke spojení hlavního procesoru s periferními zařízeními, řídí se kanálovým programem, mohou být multiplexní a selektorové

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

Mezi obvyklé způsoby propojení patří jeho využití v architektuře počítače (obrázek 11). Jeden systém může obsahovat i několik kanálů, které jsou připojeny na sběrnici hlavní paměti a procesoru. Hlavní paměť řídí „organizátor“ nebo „přidělovač“ hlavní paměti, jemuž se kanál musí podřídit jako každý jiný účastník. Každý kanál řídí periferní sběrnici, k níž mohou být připojena jednotlivá periferní zařízení. (lze připojit na jeden kanál více počítačů současně => lze vytvářet multipočítačové systémy propojením kanálů dvou různých počítačů pomocí tzv. Adaptéru kanál-kanál, jež se připojuje na V/V sběrnici)

Vlastní činnost kanálu během V/V operace je pak řízena *kanálovým programem*, prováděným přímo v kanálu. Kanálový program je uložen v hlavní paměti a čte se po *povelových slovech* (CCW), která se jednotlivě přenášejí do registru CCW v kanálu, kde je řízena jeho činnost. Prováděné operace jsou použity především pro řízení přenosu dat mezi registrem periferního zařízení a hlavní pamětí.



Obrázek 11: Kanálová architektura počítače

Struktura a funkce kanálů

Kanál má též možnost přímého přístupu do hlavní paměti a v mnohém se neliší od DMA. Jednou z částí, kterou se odlišuje kanál od DMA, je přítomnost vlastního řadiče, náležící kanálu, který mu umožňuje provádět kanálový program.

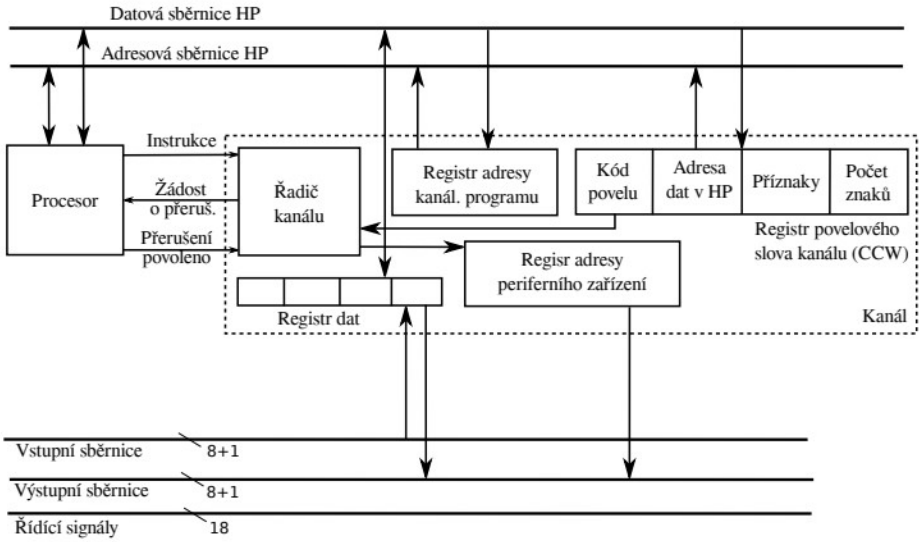
Struktura kanálu je závislá na tom, zda se jedná o selektorový nebo o multiplexní kanál. Hlavním rozdílem mezi selektorovým a multiplexním kanálem je to, že multiplexní kanál může obsluhovat více periferních zařízení současně (několik kanálových programů), kdežto selektorový pouze jen jedno zařízení. Selektorový kanál se používá k připojení velmi rychlých periferních zařízení, především disků.

Průběh přesunu dat řízený selektorovým kanálem probíhá tak, že nejprve kanál naváže spojení s určitým periferním zařízením, potom vydá příkaz ke spuštění V/V operace, provede se přesun bloku dat z hlavní paměti do periferního zařízení nebo naopak. Poté se spojení ukončí. Po ukončení spojení se může navázat spojení s dalším periferním zařízením, které je na něj připojeno. Během tohoto spojení se přesune opět celý blok dat.

Selektorové kanály se přestaly používat a nahradily je blokové multiplexní kanály stejných přenosových rychlostí a mají navíc některé další výhody.

Aby byl schopen provádět několik kanálových programů současně, je rozdělen kanál na podkanály, které jsou přiděleny jednotlivým periferním zařízením. Podkanálů bývá až 256 (maximální počet připojitelných periferních zařízení adresovatelných ve standardním styku). Multiplexní kanál se označuje jako *slabikový* nebo *blokový* podle toho, zda-li přesouvá jednotlivé slabiky nebo celé bloky mezi hlavní pamětí a periferním zařízením.

Slabikový multiplexní kanál se používá pro obsluhu pomalých zařízení jako je klávesnice, tiskárna nebo komunikační linky či spuštění určitého zařízení. Přesun několika slabik po sobě jdoucích z jednoho periferního zařízení se označuje jako *dávkový* (burst mode). Mezi jeho výhody patří zvýšená přenosová rychlost téměř o řád až několika stovek tisíc slabik za sekundu.



Obrázek 12: Struktura kanálu

5. Procesory CISC a RISC. Procesory RISC.

V dnešní době se ustálilo dělení počítačů do dvou základních kategorií podle typu použitého procesoru:

- **CISC** - počítač se složitým souborem instrukcí (Complex Instruction Set Computer)
- **RISC** - počítač s redukovaným souborem instrukcí (Reduced Instruction Set Computer)

V dnešní době totiž prakticky neexistují procesory, které by nesly „čisté“ rysy RISC a CISC, vždy jde o kompromis mezi oběma směry. Vývoj procesorů, které zpětně dostaly označení CISC, směřoval na konci 70. let k nezadržitelnému růstu jejich složitosti a tak se objevily první pokusy o celkové zjednodušení struktury procesorů. Vznikla tak zcela nová kategorie procesorů, dnes označovaná jako RISC.

Cisc (Complex Instruction Set Computing)

- označení pro skupiny procesorů vyznačující se sadou instrukcí, kterých je velké množství, mají nejednotnou délku a obsahující instrukce, které řeší několik operací naráz, obsahují relativně malý počet registrů
- toto pojmenování bylo zavedeno až zpětně po představení myšlenky RISC procesorů, těžko se hledá čistě CISC procesor, CISC procesory dnes mají prvky jak z CISC, tak z RISC

Výhody: specializované instrukce pro provedení složitých úkolů

Nevýhody: velká složitost provedení procesoru (instrukce musejí být naprogramovány mikroprogramově) i samotné instrukční sady (problémy s překladem z vyšších jazyků)

Typický zástupce: procesory řady Intel x86 (od Pentium Pro ve skutečnosti vnitřní implementace procesoru používá architekturu RISC, nicméně instrukční sada x86 je zpětně kompatibilní a uvnitř procesoru probíhá překlad z x86 CISC na RISC instrukce)

Problémy vývoje CISC

Složitost CISC procesorů vede k problémům při výrobě (velká spotřeba materiálu, větší pravděpodobnost vady, komplikovaný návrh, problémy s vysokými frekvencemi, pipelining, cache atd).

Základní konstrukční vlastnosti procesorů RISC.

RISC procesoru je třeba přenechat jen tu činnost, která je nezbytně nutná a přenést další potřebné funkce do architektury počítače, programového vybavení a kompilátoru.

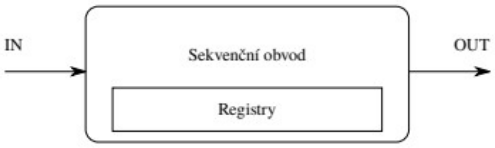
Výsledkem návrhu konstrukce procesoru RISC jsou zejména tyto vlastnosti:

- v každém strojovém cyklu by měla být dokončena jedna instrukce (to neznamená, že její vykonání trvalo jeden stroj. cyklus)
- mikroprogramový řadič může být nahrazen rychlejším obvodovým řadičem
- používat zřetězené zpracování instrukcí
- celkový počet instrukcí a způsobů adresování je malý
- data jsou z hlavní paměti vybírána a následně ukládána výhradně jen pomocí dvou instrukcí LOAD a STORE
- instrukce mají pevnou délku a jednotný formát, který vymezuje význam jednotlivých bitů
- je použit vyšší počet registrů
- složitost se z technického vybavení přesouvá částečně do optimalizujícího kompilátoru

Všechny uvedené vlastnosti tvoří dobře promyšlený a provázaný celek. Např. navýšení počtu registrů souvisí s omezením komunikace s pamětí na dvě instrukce LOAD a STORE. Pokud ostatní instrukce nemohou používat paměťové operandy, je třeba mít v procesoru uloženo více dat. Další souvislost je patrná mezi zřetězeným zpracováním a formátem instrukcí. Jednotná délka instrukcí dovoluje rychlejší výběr instrukcí z paměti a tím zajišťuje lepší plnění fronty instrukcí. Jednotný formát pak zjednodušuje dekódování instrukcí.
Výhody: zkracuje se vývoj procesoru a zpravidla již první realizované čipy fungují správně
Nevýhody: nutný nárůst délky programů, tvořených omezeným počtem instrukcí a také díky jednotné délce všech instrukcí (zpomalení, které by z toho mělo nutně plynout, se ale v praxi nepotvrdilo)

Zřetězené zpracování instrukcí (pipelining, proudové zpracování instrukcí)

Procesor si lze představit jako sekvenční obvod. Vstupem jsou strojové instrukce a data z paměti. Z výstupu se data ukládají zpět do paměti. Každá instrukce tedy musí projít celým obvodem a dokud se neuloží výsledky, nelze začít provádět instrukci následující.
Provedení instrukce musí projít vždy stejnými fázemi. Ve zjednodušené variantě je možno si představit, že instrukce se musí vybrat z paměti, dekódovat, vypočítat adresa operandu, připravit data, instrukci provést a nakonec uložit výsledky.



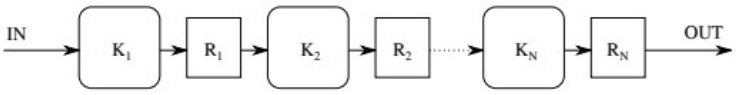
Krok		Význam
1.	**VI**	Výběr Instrukce
2.	**DE**	Dekódování
3.	**VA**	Výpočet Adresy
4.	**VO**	Výběr Operandu
5.	**PI**	Provedení Instrukce
6.	**UV**	Uložení Výsledku

Pokud bude pro provedení jednoho elementárního úkonu potřeba jeden strojový cyklus, je možno provádění instrukcí názorně zobrazit v tabulce 4. Instrukce I_2 se nezačne vykonávat dříve, než je uložen výsledek instrukce I_1. Bude-li jeden časový cyklus označen jako T_M, je potřeba pro vykonání jedné instrukce 6 cyklů. Provedení každé další instrukce pak tedy bude vyžadovat opět 6 cyklů. Zatím ovšem nepožadujeme, aby měly všechny cykly stejnou délku.

Kdyby se nám tedy podařilo osamostatnit jednotlivé části sekvenčního obvodu z obrázku 1 na samostatné obvody, aby každému obvodu odpovídala jedna fáze zpracování instrukce, mohli bychom si jej představit jako posloupnost na sebe navazujících zřetězených jednotek (v podstatě jde o princip výrobní linky, jak je známo z mnoha jiných odvětví). Obvod by pak mohl být realizován podle schématu na obrázku 2.

	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



Obrázek 2: Zřetězené zpracování (v procesoru RISC)

Mezi jednotlivé fáze zpracování však musí být pro mezivýsledky zařazen registr, který slouží jako předávací místo mezi po sobě jdoucími obvody. Z tohoto předávání mezivýsledků plyne jisté malé zpoždění. To lze ovšem v celkovém přínosu zřetězení zanedbat.
Aby uvedený princip zřetězení měl co největší přínos, musí být všechny fáze zpracování stejně časově náročné. Jinak je jasné, že nejpomalejší článek zřetězení bude brzdit všechny ostatní.
Když se podaří rozdělit vykonávání instrukce na jednotlivé úkony, přičemž časový cyklus potřebný pro každou fázi zpracování bude stejný, je možné zpracovávání instrukcí I_1 až I_7 znázornit v tabulce 5.
Je vidět, že pro vykonání 7 instrukcí stačí 12 cyklů. (v tabulce 4 byly za stejný počet cyklů vykonány pouze 2 instrukce)
Teoreticky to znamená, že v nekonečném čase nám N-úrovní zřetězení zrychlí vykonávání instrukcí $N\times$. V praxi je ovšem celkový přínos zřetězení limitován mnoha dalšími hledisky. Jednak je omezena rychlost na vstupu a výstupu zřetězené jednotky a taky dochází během vykonávání k problémům s používáním omezeného množství pomocných obvodů (registry, sběrnice, atd.). Zmíněno bylo i zpomalení předáváním mezivýsledků.

	T_1	T_2	T_3	T_4	T_5	T_6	T_7	T_8	T_9	T_{10}	T_{11}	T_{12}
VI	I_1	I_2	I_3	I_4	I_5	I_6	I_7	...				
DE		I_1	I_2	I_3	I_4	I_5	I_6	I_7				
VA			I_1	I_2	I_3	I_4	I_5	I_6	I_7			
VO				I_1	I_2	I_3	I_4	I_5	I_6	I_7		
PI					I_1	I_2	I_3	I_4	I_5	I_6	I_7	
UV						I_1	I_2	I_3	I_4	I_5	I_6	I_7

Jako nejkritičtější se ovšem ukazuje problém plnění zřetězené jednotky - a tím vzniklé fronty rozpracovaných instrukcí. Zejména podmíněné skoky znehodnocují frontu rozpracovaných instrukcí (instrukce se zpracovávají sekvenčně a pokud se vykonávání programu přesune na jinou adresu podmíněným skokem, můžeme již rozpracované instrukce zahodit). A čím je hloubka zřetězení větší, tím se zvyšují i ztráty (zde je vidět, jak může být větší hloubka zřetězení kontraproduktivní). Počet skokových instrukcí v programech je velmi vysoký, prakticky každá šestá instrukce představuje podmíněný skok.
Na základě známých technický parametrů lze hledat optimální hloubku zřetězení. (tímto se zabírat nebudeme)

Problémy zřetězeného zpracování

Zřetězené zpracování instrukcí přináší výrazné navýšení výkonu procesoru. Mezi hlavní problémy patří hazardy datové i strukturální a problémy s plněním fronty instrukcí.

Datové hazardy

- vznikají např. když některá rozpracovaná instrukce potřebuje mít k dispozici data předchozí instrukce, a ta ještě nejsou k dispozici (pro pochopení příklad na tabulce: instrukce I_5 potřebuje pro výpočet adresy v čase T_7 hodnotu, kterou instrukce I_4 uloží až v čase T_9.
- K řešení problému musí být uzpůsobena zřetězená jednotka svou konstrukcí, nebo se tyto problémy řeší už v překladači, aby se podobným situacím zabránilo.

Strukturální hazardy

- můžeme charakterizovat jako problém omezených prostředků procesoru i počítače jako celku
- např. Problém se kterým se budou potýkat některé části zřetězené jednotky:
  - při výběru instrukce, při výběru operandu i při ukládání výsledku, bude potřeba komunikovat po sběrnici, ta je ale k dispozici obvykle pouze jedna a nelze na ni povolit přístup více jednotkám současně, přístup na sběrnici je třeba koordinovat, což přinese zpomalení práce

Problémy plnění fronty instrukcí

Pro optimální činnost zřetězeného zpracování je důležitá reakce na skokové instrukce. Nejjednodušší je řešení nepodmíněných skoků a volání podprogramů s pevnou adresou. Zřetězená jednotka může být snadno upravena a začne vybírat další instrukce z cílové adresy
Trochu komplikovanější je to v případě, že cílová adresa nepodmíněného skoku se musí vypočítat. Výsledek výpočtu může být dán již rozpracovanými instrukcemi a výsledek není k dispozici dostatečně brzo a hrozí výpadek fronty. Omezit četnost těchto případů lze např. optimalizací pořadí strojových instrukcí. To je úkol zejména pro optimalizující překladače z vyšších programovacích jazyků.
Vážný problém nastane ovšem v okamžiku, kdy se při zřetězeném zpracování instrukcí narazí na podmíněný skok. Podobně jako u předchozího případu, kdy jsme neznali cílovou adresu, tady sice adresu (většinou) známe, ale nevíme, jestli se skok provede, či nikoliv. Nemá smysl zpracovávání zastavit a čekat na výsledek. Lepší je pokračovat ve zpracovávání sekvenčním způsobem a pokud se skok neprovede, tak se rozpracovaná fronta instrukcí použije. V opačném případě se rozpracované instrukce ignorují a fronta se začne plnit znovu.

Velmi dobré řešení se používá pro vysoce výkonné počítače, kde se implementují dvě paralelní fronty. Druhá fronta je obvykle kratší, podle našeho příkladu z tabulky 3 může být omezena jen na první čtyři kroky zpracování. Prováděcí jednotka pak může začít vybírat alternativně připravené instrukce z druhé fronty. Je jasné, že přepínání fronty zabere určitý čas. Tato ztráta je ovšem výrazně výhodnější, než výpadek celé fronty.
V procesorech RISC se kromě dvou uvedených krajních řešení používají i další metody, jako např. predikci skoků.

Metody pro zlepšení plnění fronty instrukcí

Procesory RISC používají vždy frontu instrukcí. Aby bylo riziko ztráty již vybraných instrukcí z paměti co nejmenší, je třeba omezit možnosti, jak sekvenční vykonávání instrukcí měnit.
V moderních systémech se předpokládá, že program nemůže modifikovat sám sebe. Dále pak podmíněné skoky jsou vždy na pevně danou adresu a pouze nepodmíněné skoky mohou používat registry pro určení cílové adresy skoku.
V praxi statistiky ukázaly, že z výše uvedených problémů se nejčastěji vyskytují podmíněné skoky. Jejich četnost je podle typu řešených úloh 15÷25%. Vyplatí se proto hledat jednoduchá a levná řešení, jak chování podmíněných skoků předpovídat.

Bit predikce skoku
Tato metoda předpokládá, že se ve formátu instrukce vyhradí jeden bit predikující, zda se skok provede či nikoliv. Jednotka výběru instrukcí pak vybírá instrukce z předpokládané adresy.
Predikce může být statická, nebo i dynamická.

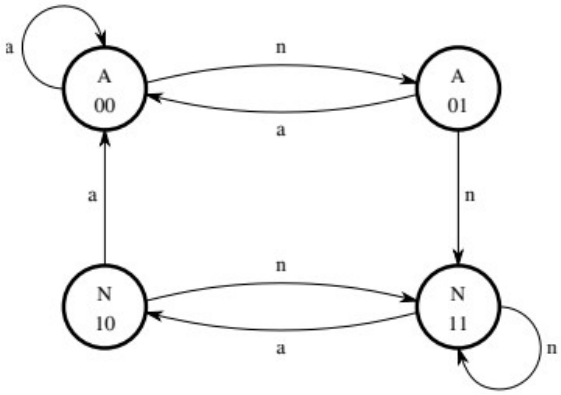
Statická predikce

- do instrukce se vkládají příslušné bity predikce již při kompilaci, nebo přímo programátorem při tvorbě programu

Dynamická predikce

- při každém provedeném podmíněném skoku zaznamená, jestli se skok provedl, či nikoliv
- je výhodnější, protože se predikce přizpůsobí aktuálním podmínkám
- vyžaduje nutnost zapisovat příslušný bit predikce a to může být technicky komplikované, případně i nemožné (kód programu je chráněn proti přepisování, případně může být uložen v nepřepisovatelné paměti).

Jednobitová predikce je velmi jednoduchá, ale její použití v podmínce cyklu znamená, že dojde k jednomu selhání vždy na začátku cyklu, a k jednomu na konci (u dynamické metody). Pokud si ale představíme dva nebo i více vnořených cyklů, je četnost selhání výrazně vyšší.
Lepší chování nabízí predikce dvoubitová. Ta dokáže snížit u cyklů selhání predikce na jediné a to na konci cyklu. Chování dvoubitové predikce si můžeme znázornit stavovým automatem na obrázku 3.



Jde o čtyřstavový automat, kde stav A predikuje provedení skoku, zatím co stav N říká, že skok se provádět nebude. Přechody a a n označují, zda se naposledy skok prováděl, či nikoliv. Je tedy patrné, že pouze pokud se skok dvakrát po sobě provede, může se změnit predikce z N na A a obráceně. V ustáleném stavu se tedy predikce nastaví na $A - 00$ nebo $N - 11$.

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).
Podívejme se teď na tabulku 6. Pokud I_5 bude skoková instrukce a skok se provede, budou rozpracované instrukce označené jako X ignorovány.
Co by se však stalo, kdybychom činnost procesoru po skokové instrukci nezastavili? Rozpracované instrukce by se provedly a pokračovalo by se instrukcí I_6. Takové řešení by zjednodušilo logiku vnitřního řízení procesoru, ale po skokové instrukci by se provedlo ještě několik rozpracovaných instrukcí, což by mohlo narušit logiku programu.
Můžeme tedy místo za podmíněným skokem vyplnit prázdnými instrukcemi *NOP*. Tím se sice logika programu nezmění, ale vykonáváním prázdných instrukcí se efektivita činnosti procesoru nezvýší.
Pokud by se ale podařilo před skokovou instrukcí najít několik instrukcí nesouvisejících přímo s podmíněným skokem a vyhodnocením jeho podmínky, mohli bychom je zařadit na prázdné místo označené v tabulce 6 jako X.
Jak bude vypadat provádění upraveného kódu se můžeme podívat v tabulce 7. V optimalizujícím kompilátoru je poměrně snadné realizovat požadované přeskupení instrukcí a využít tak zpoždění skoku k vyšší efektivitě práce zřetězené jednotky.

	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

Pro názornost si můžeme představit celé řešení na jednoduchém příkladu v programovacím jazyce C a jeho následném přepise do assembleru:

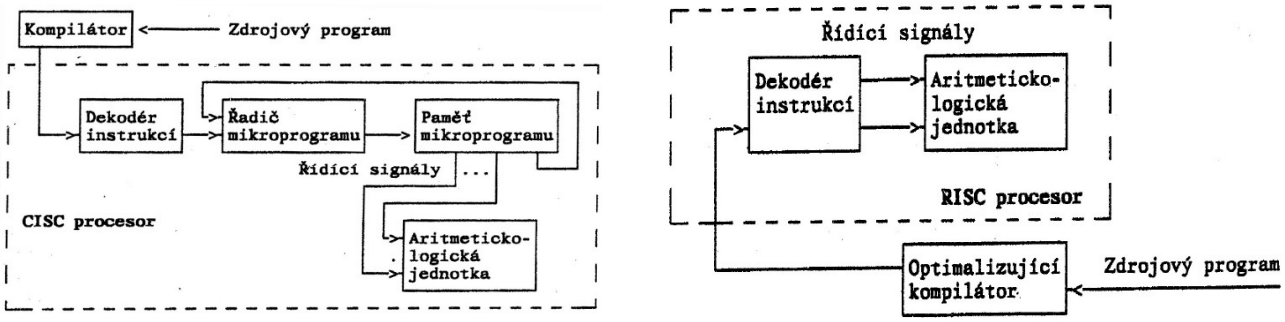
```
/* Příkaz v jazyce C */                    ;Přepis do assembleru
if ( i++ == j++ ) { /* blok příkazů ... */ }   CMP i, j
k++;                                        JNE pokracuj
                                            INC i
                                            INC j
                                            ; blok příkazů ...
                                        pokracuj:
                                            INC k
```

Z příkladu je vidět, že zvýšení hodnoty proměnných *i* a *j* musí být provedeno ještě před vykonáním „bloku příkazů“. Přesto pokud víme, že skoková instrukce bude zpožděna, můžeme instrukce pro inkrementaci proměnných *i* a *j* nechat až za skokovou instrukcí. Pro analogii s tabulkou 7 si představme, že instrukce I_5 je podmíněný skok JNE pokracuj a inkrementace proměnných budou instrukce I_3 a I_4.

Použití paměti skoků

Velmi rozšířeným řešením se stala tabulka provedených podmíněných skoků, realizovaná přímo jako součást procesoru. Velikost tabulky je předem pevně stanovena a do tabulky se ukládají adresy posledních provedených skoků. K údajům v tabulce se může aplikovat jednobitová, nebo dvoubitová predikce skoků. Tato metoda nevyžaduje úpravu formátu strojových instrukcí, není spojena s činností překladače a je možno ji použít i v nových generacích procesorů, kde je vyžadována zpětná kompatibilita strojového kódu.

CISC architektura **RISC architektura**



Samostudium: Vyberte si z v současnosti vyráběných RISC procesorů jeden a popište ho v souladu se základními konstrukčními vlastnostmi RISC.

ARM - Architektura **ARM** se nejvýrazněji uplatňuje v mobilních zařízeních (mobilní telefony, tablety) a ve vestavěných systémech (pevný disk, USB flash disk, Wi-Fi čipy, routery apod.). Nízká spotřeba energie při vysokém výpočetním výkonu má zásadní význam hlavně v zařízeních napájených bateriemi, avšak je velkou výhodou také u zařízení pracujících v náročných tepelných podmínkách. Nízkopříkonové procesory totiž nepotřebují složité a přitom relativně nespolehlivé chlazení.

Charakteristika architektury ARM

- přístup do paměti pouze instrukcemi Load/Store
- částečné překrývání vnitřních registrů (25 částečně se překrývajících 32bitových registrů)
- možnost podmíněného vykonání instrukcí
- jednoduchý a výkonný instrukční soubor, jednoduše využitelné kompilátory vyšších programovacích jazyků
- (3 stupňové zřetězení instrukcí)

Procesor ARM obsahuje 44 základních instrukcí s jednotnou šířkou 32 bitů.
Procesor pracuje ve čtyřech základních režimech: uživatelský režim USR, privilegovaný režim supervizora SUP, privilegovaný režim přerušení IRQ, privilegovaný režim rychlého přerušení FIQ
Procesory ARM podporují dva adresové módy. Můžeme adresovat buď prostřednictvím čítače instrukcí, nebo pomocí bázové adresy uložené v jednom z vnitřních registrů.
Dva typy přerušení:
FIQ (fast interrupt request), který je pro neodkladné události
IRQ (interrupt request), který se používá pro události nevyžadující extrémně krátkou dobu odezvy

6. Procesory CISC firmy Intel - historie i8086 až CORE2.

Intel 8080

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

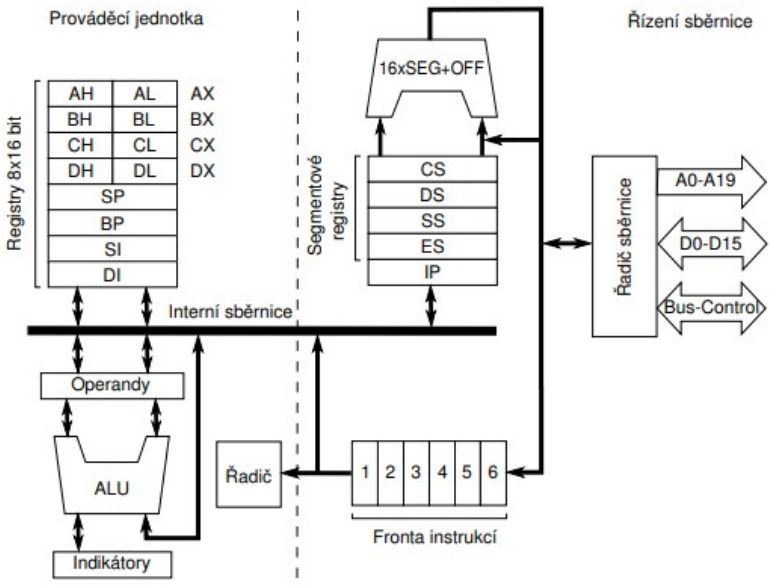
- 8 bitový mikroprocesor není přímo prvním členem v řadě x86
- stal se základem pro celou řadu prvních jednodeskových počítačů
- svou instrukční sadou inspiroval další výrobce při vývoji 8 bitových procesorů
- je na úrovni assembleru kompatibilní s následnou 16 bitovou verzí 8086.

Intel 8086

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

- prvním 16 bitovým procesorem, dovoloval adresovat až 1MB paměti a to díky segmentaci paměti na 64kB bloky.

Z obrázku je patrná jednoduchá vnitřní architektura s oddělením výkonné a adresovací části procesoru. Osm 16 bitových registrů tvoří základ, který je implementován ve všech následujících generacích x86.



Obrázek 1: Architektura procesoru i8086

Intel 8088

- uveden rok po uvedení 8086
- nejedná se o nový typ, vnitřní architektura zůstává zachována
- změněn výstup datové sběrnice, byla zúžena na 8 bitů, došlo tím k zpomalení práce procesoru, protože přístup k 16bitovým datům musel být na sběrnici multiplexován
- díky užší datové sběrnici bylo možno konstruovat levněji a bylo možné v té době využít celou řadu periferních obvodů navržených pro 8 bitové mikroprocesory
- nejlepším důkazem o vhodnosti využití tohoto procesoru pro výrobu osobních počítačů byl nový standard PC-XT, navržený firmou IBM

Intel 80186/188

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

- 1982 – Intel přichází s vylepšenou verzí procesoru 8086/88
- vylepšena architektura, přidány, upraveny a zrychleny některé instrukce
- navržen primárně pro vestavěna (embedded) zařízení a obsahoval celou řadu pomocných obvodů přímo na čipu, zejména DMA, hodiny, časovače a porty
- nebyl kompatibilní s platformou PC-XT

Intel 80286

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

- navýšení výpočetního výkonu zkrácením času potřebného pro vykonání většiny instrukcí a možností taktovat procesor na vyšší frekvence
- umožňuje přepnutí do *protected mode (P-M)* ve kterém je možné adresovat až 16MB paměti, v tomto režimu mohou být programy vykonávány se 4 různými úrovněmi oprávnění
- Real Mode R-M - režim práce, kompatibilní s předchozí verzí procesoru
- programy napsané pro R-M nemohou v P-M pracovat (proto byly tyto procesory využívány téměř na 100% v R-M)
- po přepnutí do P-M nebylo možné se vrátit zpět do R-M jinak, než resetem
- procesor byl vybaven jednotkou MMU (Memory Management Unit) – umožňovala stránkování (tím se využíval v sálových počítačích, přenesl se i do osobních počítačů)
- kromě stránkování, bylo možno používat i virtuální paměť
- procesor byl chopen pracovat s 30bitovou adresou, mohl tedy adresovat virtuálně až 1GB paměti

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 řady x86 (procesor 80386)
- plná zpětná kompatibilita s předchozími typy
- všech 8 registrů procesoru však bylo rozšířeno na 32 bitů (na stejnou šířku se rozšířila datová i adresní sběrnice)
- přidány nové instrukce
- rozšíření *real mode* a *protected mode* o další režim – *virtual mode*
- ve virtual mode bylo možno po přepnutí do P-M vykonávat programy napsané pro R-M (tímto se otevřela cesta i pro zpětnou kompatibilitu se staršími programy v nových OS)
- další novinka: řadič pro vyrovnávací paměť, bylo možné na základní desce počítače implementovat Cache L1, šlo o paměti velikosti řádově jednotky až desítky kB
  (tato implementace se stala nezbytná pro procesory, které byly taktovány na frekvenci 33Mhz a výše)
- modernizován byl i systém pro správu virtuální paměti (virtuálně bylo možno adresovat až 64TB paměti)
- firma COMPAQ prvně implementovala procesor 80386, poprvé, kdy prvním výrobcem nebyla firma IBM

Intel 80386SX

- 1988, firma Intel
- nejednalo se o inovaci, ale o krok zpět
- stejně jako u 8088, vedly ekonomické důvody k výrobě upravené verze se zúženou datovou sběrnicí v tomto případě na 16 bitů, tím se zlevnila výroba základních desek a využít součástková základna používaná pro počítače s procesory 80286.

Intel 8087/287/387

- matematický koprocesor – umožňuje výpočty s čísly s plovoucí desetinnou tečkou (FPU)
- jednalo se o obvody navržené vždy pro konkrétní typ procesoru, se kterým spolupracovaly
- samostatná činnost procesoru nebyla možná
- značení obvodů v souladu s hlavním procesorem, pro odlišení poslední číslice je 7

Intel 80486DX/i486DX

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

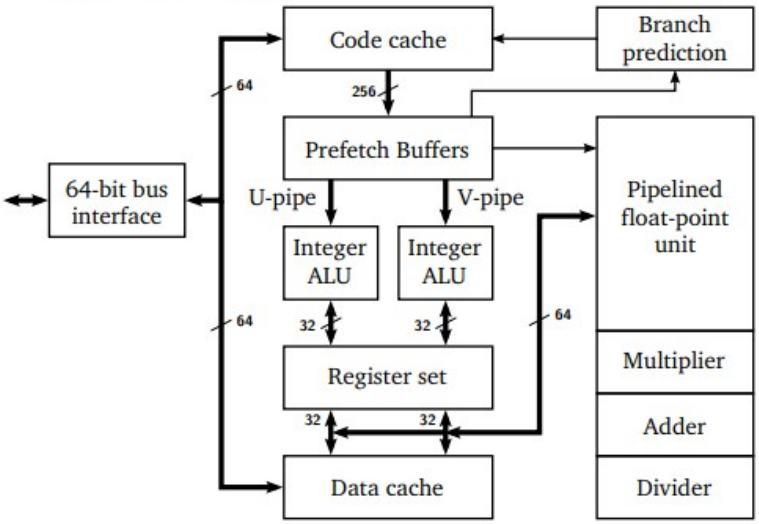
- čtvrtá generace procesorů, zůstává 32 bitová, ale téměř pětinásobný počet tranzistorů oproti předchůdci
- dvojnásobný výkon ve srovnání s procesorem 80386 při stejné frekvenci
- navýšením výkonu bylo dosaženo vylepšením ALU, delší frontou instrukcí a zlepšením propustnosti mezi jednotlivými jednotkami procesoru
- vyrovnávací paměť L1 implementována přímo v procesoru, byla společná pro data i kód a velikost byla 8kB
- vylepšená i MMU jednotka a urychlila se tak činnost procesoru v režimu *protected mode*
- integrace matematického koprocesoru – nebylo nutno pro výpočty s FPU instalovat další obvod
- připojením FPU přímo na vnitřní sběrnici bylo dosaženo většího výkonu než u koprocesoru 80387
- 1991 – verze i486SX, levnější varianta procesoru, neobsahovala jednotku FPU
- 1992 – verze i486DX2 – varianta, kde se vnitřní frekvence procesoru násobí dvěma (50/25MHz, 66/33 MHz)
  - výhoda: předchozí procesory DX se daly přímo nahrazovat touto verzí DX2
- 1994 - verze i486DX4 – navýšení vnitřní frekvence 3x (75/25 MHz, 100/33 MHz), napájen 3.3V

Intel Pentium

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

- pátá generace procesorů
- první procesor v řadě x86, kde se použila RISC architektura
- superskalární architektura tvořena dvěma paralelními ALU
- v ideálním případě mohl zpracovat i dvě jednoduché instrukce současně
- u složitějších instrukcí obě jednotky spolupracovaly při vykonání této instrukce (např. řetězové instrukce)
- rozdělení vyrovnávací paměti cache L1 na dvě – pro kód a pro data
- rozšíření sběrnic a poprvé se objevuje jednotka pro predikci skoků
- zůstává oddělena FPU jednotka od celočíselné prováděcí jednotky
- první verze napájeny 5V, 60/66 MHz
- 1994 – verze s frekvencemi 75/90/100/120 MHz a napájením 3.3 V => očekávaný krok vpřed
- 1997 – rozšíření o jednotku MMX s podporou multimediálních instrukcí

Pentium Block Diagram

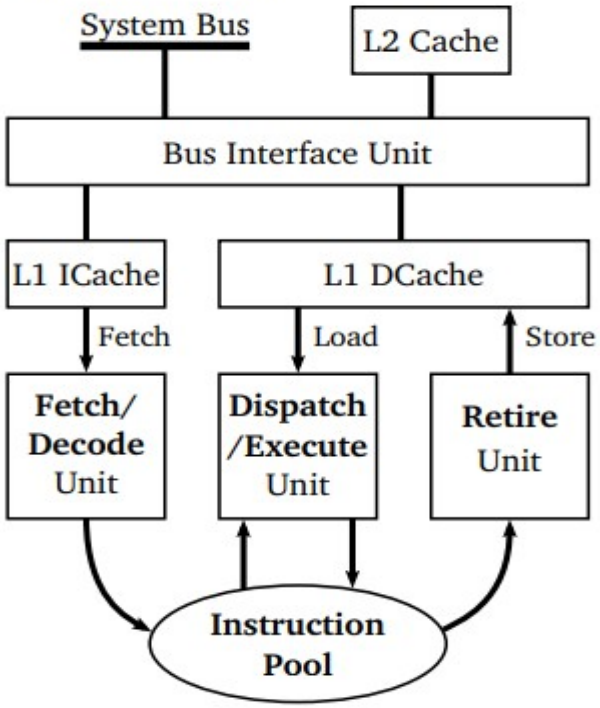


Intel Pentium Pro

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

- zásadní technologický zlom v konstrukci procesorů řady x86
- primárně určen pro segment serverů a ne desktopů, odpovídala tomu cena i výkon
- ve srovnání s Pentiem o 50% větší výkon při stejné frekvenci
- přímo na procesoru je implementována paměť cache L2 procesor i paměť na samostatném čipu
- Fetch/Decode jednotka vybírá z paměti instrukce x86 a dekóduje je na 118 bitové RISC instrukce (označení jako mikro-operace) => procesor je kompatibilní se stávající instrukční sadou
- mikro-operace z dekódovací jednotky se neřadí do fronty, ale je použita nová technologie
- instrukce se ukládají do banky dekódovacích instrukcí (Instruction pool), z této banky instrukcí si může prováděcí jednotka Dispatch/Execute vybírat instrukce mimo pořadí out-of-order, prováděcí jednotka tak sama může maximalizovat svůj výkon a vykonávat instrukce mimo pořadí a stále maximálně využívat všechny obvody prováděcí jednotky
- provedené instrukce jsou z prováděcí jednotky uloženy zpět do banky instrukcí, odkud jsou vybírány ukončovací jednotkou Retire Unit, odtud plynou data zpět do registrů a paměti cache L1, dále pak přes rozhraní sběrnice do cache L2 a do hlavní paměti
- aby bylo možné takový výkonný systém využít naplno, není možné jej propojit obyčejnou lineární frontou instrukcí, jedině banka instrukcí zajistí, že si prováděcí jednotka bude umět sama rozhodovat o pořadí provádění instrukcí, aby maximálně využila paralelní činnost všech svých obvodů

Pentium Pro Block Diagram



Intel Pentium II

Výroba	1997
Technologie	BiCMOS, 0.35 ÷ 0.18 μm
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

- přebírá celé technické řešení Pentia Pro
- nejvýkonnější verze pro servery představena v roce 1998, měla 512 kB cache L2, pro dosažení lepšího výkonu, stále však implementována jako samostatný čip na modulu procesoru, nesly označení Xeon
- v roce 1999 byla představena verze procesoru s integrovanou pamětí cache L2 256 kB společně s procesorem na jednom čipu, procesor byl určen pro přenosné počítače

Intel Pentium III

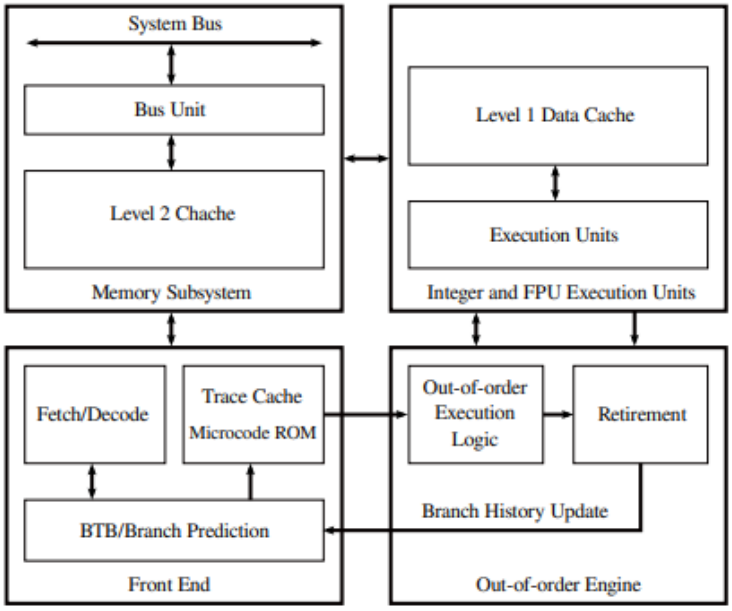
Výroba	1999 ÷ 2003
Technologie	BiCMOS, 0.25 ÷ 0.13 μm
Tranzistory	jádro $9.5 \cdot 10^6 + 18.5 \cdot 10^6$ pro 265 kB cache L2
Frekvence	450 MHz ÷ 1.4 GHz
Datová sb.	64 bit
Adresní sb.	36 bit

- první verze stejné jako Pentium II
- dále představena verze s integrovanou verzí paměti cache L2 společně na jednom čipu s procesorem
- rozšířen o další prováděcí jednotky, přidání jednotky SSE
- zlepšení predikce skoků a je optimalizováno řízení obvodů procesoru s ohledem na minimalizaci spotřeby
- 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, špatné ohlasy => stejný výkon jako Pentium 3 a více se zahříval
- blokové schéma odpovídá zhruba architektuře Ppro, procesor je rozdělen na 4 logické jednotky, i zde superskalární dekódovací jednotka předává dekódované mikro-operace do banky instrukcí a superskalární pro12 váděcí jednotka je vykonává mimo pořadí
- instrukční Cache L1 obsahuje dekódované mikro-operace, významně se tím šetří práce dekódovací jednotce, které nemusí krátké opakující se sekvence kódu opakovaně dekódovat
- díky vyrovnávací paměti s dekódovanými instrukcemi je predikce skoků dvouúrovňová. Jedna predikce skoků pracuje před dekódovacími jednotkami a druhá, menší, pomáhá při předávání již dekódovaných instrukcí do banky instrukci
- architektura NetBurst zavádí při zřetězeném zpracování v prováděcích jednotkách 20 úrovňové zřetězení. To je dvojnásobek, než se používalo u PPro



Obrázek 4: Zjednodušené blokové schéma procesoru Pentium 4

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

- přerod architektury na 64 bitů, převzetí návrhu architektury od AMD, za cenu neoptimálního technického řešení
- registry byly rozšířeny na 64 bitů, zdvojnásoben jejich počet a datová sběrnice byla 40 bitová
- pro dosažení potřebného výkonu bylo implementováno v prováděcích jednotkách velmi hluboké 30 úrovňové zřetězení
- procesory taktovány na frekvence od 3 GHz a výše, aby podávaly potřebný výpočetní výkon a velice se přehřívaly. Jejich ztrátový výkon byl od 85 až do 115 W!

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

- určený pro přenosné počítače
- velmi výkonný procesor s nízkou spotřebou energie
- vývojáři převzali energeticky nenáročný PIII
- využili stejné principy pro konstrukci rozhraní systémové sběrnice jako u P4
- vylepšili dekódování instrukcí a zlepšili predikci skoků
- přímo na čip byla umístěna 1MB cache L2

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

- pokračování řady Pentium M
- opětovné rozšíření adresní sběrnice na 36 bitů
- díky přechodu na technologii výroby 65nm se příkon procesoru snížil natolik, že bylo možno implementovat na jeden čip dvě jádra procesoru i pro přenosné počítače
- paměť cache L2 se rozrostla na 2 a 4 MB a byla společná pro obě jádra
- určen nejen pro přenosné počítače, ale i součástí desktopů

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

- procesory mají 64 bitovou architekturu Intel 64 (EM64T)
- jádro je i nadále založeno na architektuře PPro
- představením tohoto procesoru se ukončil vývoj P4 s architekturou NetBurst
- přineslo modernizaci jádra oproti verzi Core, posílení dekódovací jednotky a přidání dalších prováděcích jednotek
- pro urychlení zpracování instrukcí, byla vylepšena „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

- procesor s velmi nízkým příkonem pro ultralehké přenosné počítače a vestavěná zařízení
- založen na zcela odlišné architektuře Bonnel
- procesor má implementovanou technologii HT (Hyper-Threading), proto má frontu instrukcí i sady registrů zdvojené
- jednotlivé logické celky procesoru jsou nazývaný jako „clusters“
- návrh je směrován k minimalizaci spotřeby a ne maximalizaci výpočetního výkonu

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

- procesor RISC určený pro výkonné servery
- 64 bitový s architekturou označovanou IA-64
- zcela nová instrukční sada odlišná od procesorů x86
- využívá paralelismus na úrovni vykonávaných instrukcí a potřebné pořadí instrukcí musí být připraveno překladačem
- slabinou procesoru byla jeho zpětná kompatibilita s 32 bitovu architekturou x86
- v oblasti výkonných serverů byl např. v roce 2008 čtvrtým nejprodávanějším procesorem

Největší konkurentem Intelu je AMD. Procesory AMD K6, K6-2, K6-III, Athlon, Duron. Odchylky od P-Pro a PIII.

AMD K6

K6 (Model 6, 350 nm)

- L1 Cache: 32 + 32 KB (Data + Instrukce)
- Multimediální rozšíření: MMX
- Patice Socket 7, sběrnice 66 MHz
- Napětí jádra: 2.9 V (166/200) 3.2 V (233)
- Uveden na trh: 2. duben, 1997
- Frekvence: 166, 200, 233 MHz

- jádro vyvinuto společností NexGen, kterou AMD koupila
- 57 instrukcí MMX
- 64 kB L1 cache
- pro způsob zpracování instrukcí x86 pomocí jádra na základě RISC byl procesor o něco výkonnější než Pentium a Pentium Pro na stejné frekvenci

K6-2

K6-3D (Chomper, 250 nm)

- L1 Cache: 32 + 32 KB (Data + Instrukce)
- Multimediální rozšíření: MMX a 3DNow!
- Patice Super Socket 7, sběrnice 66 či 100 MHz
- Napětí jádra: 2.2 V
- Uveden na trh: 28. květen, 1998
- Frekvence: 233, 266, 300, 333 a 350 MHz

- mikroprocesor od společnosti AMD, který byl taktovaný na frekvencích 233 - 550 MHz
- přímou konkurencí procesorům Intel Celeron a Pentium II
- relativně velká L1 cache 64 kB (32 kB Data + 32 kB Instrukce)
- jádro běželo na napětí 2,2 V
- vyráběl se pro základní desky s architekturou Super Socket 7

K6-III

K6-III (Sharptooth, 250 nm)

- L1 Cache: 32 + 32 KB (Data + Instrukce)
- L2 Cache: 256 KB (stejná frekvence jako jádro)
- Multimediální rozšíření: MMX a 3DNow!
- Patice Super Socket 7, sběrnice 100 MHz
- Napětí jádra: 2.2 V nebo 2.4 V
- Uveden na trh: 22. únor, 1999
- Frekvence: 400 a 450 MHz

- mikroprocesor architektury x86 od společnosti AMD a současně poslední a nejvýkonnější mikroprocesor pro základní desky s paticí Super Socket 7
- upravený procesor K6-2 s přidanou L2 cache přímo na procesoru
- vyráběn ve frekvencích 400 až 550 MHz

Athlon

Athlon je obchodní značka série rozdílných x86 procesorů navržených a vyrobených americkou společností AMD. Původní Athlon, neboli Athlon Classic, byl první ze 7. generace x86 procesorů (786) a první si udržel na významnou dobu náskok před konkurenčními procesory od Intelu. AMD pokračuje v této řadě s Athlonem 64, procesorem 8. generace obsahujícím technologii AMD64. Výkonnost Athlonu byla doplněna zachováním nižší ceny proti procesorům firmy Intel.

Duron

AMD Duron je procesor architektury x86 firmy AMD. Byl uveden 19. června roku 2000 jako levnější varianta k procesoru Athlon firmy AMD a procesorům Pentium III a Celeron od konkurenční firmy Intel. Jeho výroba byla ukončena roku 2004 a byl nahrazen procesorem Sempron.

Duron „Spitfire“ (Model 3, 180 nm)

- L1-Cache: 64 + 64 KiB (data + instrukce)
- L2-Cache: 64 KiB, na rychlosti procesoru
- MMX, 3DNow!, Extended 3DNow!
- Socket A (EV6)
- Front Side Bus: 200 MT/s
- VCore: 1.50 V - 1.60 V
- Uvolněn: 19. červen 2000
- Takt: 600 MHz - 950 MHz

Duron „Morgan“ (Model 7, 180 nm)

- L1-Cache: 64 + 64 KiB (data + instrukce)
- L2-Cache: 64 KiB, na rychlosti procesoru
- MMX, 3DNow!, Extended 3DNow!, SSE
- Socket A (EV6)
- Front Side Bus: 200 MT/s
- VCore: 1.75 V
- Uvolněn: 20. srpen 2001
- Takt: 900 MHz - 1300 MHz

Duron „Applebred“ (Model 8, 130 nm)

- L1-Cache: 64 + 64 KiB (data + instrukce)
- L2-Cache: 64 KiB, na rychlosti procesoru
- MMX, 3DNow!, Extended 3DNow!, SSE
- Socket A (EV6)
- Front Side Bus: 266 MT/s
- VCore: 1.50 V
- Uvolněn: 21. srpen 2003
- Takt: 1400, 1600, 1800 MHz

7. Polovodičové paměti statické a dynamické, hierarchické uspořádání paměti počítače.

(Základní historický přehled)

- V roce 1955 fungovala feritová paměť na principu zmagnetizovaných feritových jader.
- V bubnových pamětech byl magnetický materiál nanesen na nemagnetický buben, který se otáčel vysokou rychlostí.
- Bublinové paměti byly magnetické paměti, které jsou založeny na využití velkokapacitních magnetických posuvných registrů.
- Polovodičové paměti byly jednobitové a vícebitové posuvné registry
- V roce 1960 byla vyvinuta polovodičová technologie MOS.
- V roce 1970 byly představeny DRAM a v roce 1971 SRAM paměti.

Základní klasifikace pamětí.

Podle typu přístupu mohou být paměti rozděleny na:

- RAM (Random Access Memory) - paměti s libovolným přístupem.
- SAM (Serail Access Memory) - paměti se sériovým přístupem.
- Paměti se speciálními způsoby přístupu - asociativní paměť, paměť typu fronta, paměť typu zásobník, vícebránové paměti, paměti s kombinovaným řízením.

Podle možnosti zápisu/čtení mohou být paměti rozděleny na:

- RWM (Read Write Memory) - paměti pro zápis i čtení.
- ROM (Read Only Memory) - paměti pouze pro čtení.
- Kombinované paměti
  - NVRAM (Non Volatile RAM) - kombinace RWM a E2PROM.
  - WOM (Write Only Memory) - paměť, do které lze pouze zapisovat.
  - WORM (Write Once-Read many times Memory) - optické disky CD ROM.

Podle principu elementární buňky mohou být paměti rozděleny na:

- SRAM statické paměti.
- DRAM dynamické paměti.
- PROM, EPROM, EEPROM, FLASH - programovatelné paměti.

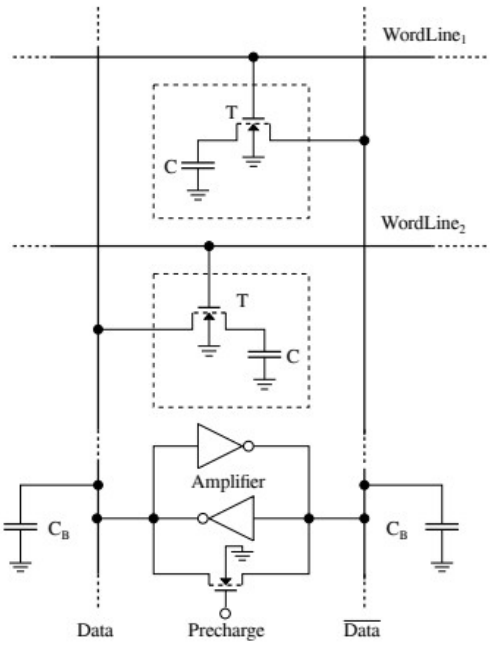
Podle uchování informace po odpojení napájení:

- Non Volatile se označují ty paměti, které informaci uchovávání i po odpojení napájení, jako jsou různé typy pamětí xxROM,
- Volatile - paměti ztrácející uloženou informaci po odpojení napájení, jako jsou paměti DRAM a SRAM.

Otázkou může v této chvíli být, kam zařadit hlavní paměť počítače, které se hovorově říká „ramka“. Z uvedeného dělení pamětí je zřejmé, že toto označení je neúplné a tedy nepřesné. Hlavní paměť počítače je určena pro čtení a zápis, musí umožňovat náhodný přístup a informace je uložena v kondenzátoru a po odpojení napájení se uložené informace ztratí. Úplné označení by tedy dle předchozího dělení být Volatile RWM DRAM.

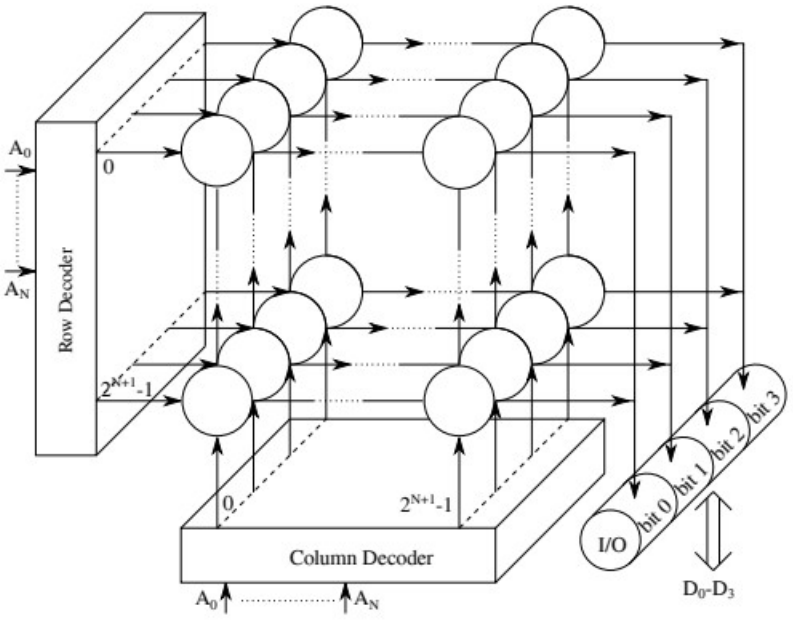
Dynamické paměti

- v dynamických pamětech je informace uložena ve formě náboje v kondenzátoru. Kondenzátor může být buď nabitý (logická 1) nebo vybitý (logická 0)
- Na obrázku 1 jsou znázorněny dvě jednotranzistorové paměťové buňky dynamické paměti.
- Kondenzátory jsou extrémně miniaturní (jsou to jen parazitní kapacity na tranzistoru, kapacita velmi malá: jednotky až desítky fF femto Farad)
- Kondenzátor si tak není schopen zachovat uloženou informaci po neomezenou dobu, i velmi malý proud tekoucí do nebo z tohoto kondenzátoru, vyvolá velké změny jeho napětí v krátkém čase
- Je proto nutné často obnovovat napětí kondenzátoru - tato procedura je označovaná jako občerstvení neboli refresh
- Pro tento proces jsou dnes na čipu implementovány speciální obvody, těm stačí pouze pravidelně přečíst libovolnou buňku z každé řady a následovně bude občerstvena celá tato řada
- Dynamické paměti zapomenou všechna svá uložená data během přibližně 10 ms



Obrázek 1: Paměťové buňky DRAM

- Obrázek 2 ukazuje zjednodušenou organizaci paměti DRAM, kde každé kolečko reprezentuje jednu paměťovou buňku
- Paměťové buňky jsou umístěny ve čtvercové matici v jedné nebo více vrstvách
- Výběr buňky tak musí být proveden ve dvou krocích pomocí řádkového (Row) a sloupcového (Column) dekodéru
- Vodiče vedoucí z dekodérů řádku a sloupce slouží k výběru paměťových buněk a vodiče vedoucí z paměťových buněk slouží k přenosu uložených dat do I/O bufferu
- Rozdělení adresování na dva dekodéry přináší výhodu při adresování



Obrázek 2: Organizace paměti DRAM

- Například pro adresování 2^{20} (1 Mbit) lineárně uspořádaných paměťových buněk, je potřeba 20 adresových vodičů $A_0 \div A_{19}$. Pokud se však paměťové buňky uspořádají do matice, stačí řádkovému i sloupcového dekodéru pro adresování stejné kapacity 1 Mbit ($2^{20} = 2^{10} \times 2^{10}$) jen polovina adresních vodičů $A_0 \div A_9$
- Během přístupu do paměti je nejprve dodána adresa řádku a na stejných vodičích následně adresa sloupce vybrané buňky, díky tomu je počet vodičů zredukován a obsahová kapacita zvětšena
- Pro výběr dekodéru, kterému je právě adresa určena, však musí být navíc přidány dva řídící signály RAS a CAS

Organizace čipu DRAM

- s vývojem paměťových čipů s většími kapacitami byly zavedeny různé formy organizace
- 1 Mbit čip s jedním datovým vývodem má organizaci 1 Mword na 1 bit (To znamená, že paměťový čip se skládá z 1 M slov o šířce 1 bit na každý vývod)
- další forma organizace pro 1 Mbit čip je 256 Kword × 4 bitová organizace (Tyto čipy mají tedy 256 Kwords se šířkou čtyři bity, takže mají čtyři vývody, kapacita paměti je také 1Mbit)
- první číslo vždy znamená počet slov (words) a druhé počet bitů na slovo
- hlavní vlastností je počet datových vývodů, to jest šířka, ve které slovo může být vloženo na vstupu nebo obdrženo na výstupu během přístupu do paměti

Princip činnosti DRAM

Podle adresy, kterou poskytlo CPU přijme adresový buffer adresu paměti jako výstup externího paměťového kontroléru. Z tohoto důvodu je adresa rozdělena na dvě části, adresu řádku a adresu sloupce. Tyto dvě adresy jsou čteny do adresového bufferu jedna za druhou. Tento proces se nazývá mutliplexing. Důvod tohoto dělení je zřejmý: na adresování jedné buňky v 4 Mb čipu s 2048 řádky a 2048 sloupci by bylo třeba celkem 22 adresových bitů (11 pro řádek a 11 pro sloupec). Pokud by měly být přesunuty všechny adresové bity najednou, bylo by třeba 22 adresových vývodů. Potom by musel být pouzdro čipu velmi velký.

Proto je lepší přesunout adresu paměti ve dvou částech. Obvykle adresový buffer nejprve čte adresu řádku a potom adresu sloupce. Tento adresní multiplexing je kontrolován RAS (Row Address Strobe) a CAS (Column Address Strobe) řídícími signály. Pokud paměťový kontrolér pošle adresu řádku, tak zároveň aktivuje RAS signál. RAS informuje čip DRAM, že dodaná adresa je adresa řádku. Nyní kontrolér DRAM aktivuje adresový buffer k získání adresy a přesune ji do dekodéru řádku, který ji dekóduje. Pokud později paměťový kontrolér poskytne adresu sloupce, potom aktivuje CAS signal. Tak kontrolér DRAM pozná, že tentokrát je přesunována adresa sloupce a aktivuje znovu adresový buffer. Adresový buffer přijme poskytnutou adresu a přesune ji do dekodéru sloupce.

Paměťová buňka adresovaná tímto způsobem předá na výstup uložená data, která jsou zesílena čtecími zesilovači a přesunuta do I/O bufferu. Buffer nakonec poskytne informace jako výstupní data *Dout* přes datové vývody paměťového čipu.

Pokud mají být data zapsána, paměťový kontrolér aktivuje WE (Write Enable) signál a přesune zapisovaná data Din do I/O bufferu. Pomocí čtecích zesilovačů je informace zesílena, přesunuta do adresované paměťové buňky a v ní uložena.

Paměťový kontrolér počítače tedy řeší 3 různé úkoly: rozdělení adresy získané z CPU na adresu řádku a sloupce, které jsou přesunuty do paměti jedna po druhé; správně aktivuje RAS, CAS, WE a READ signály; přesunuje uložená data a přijímá data k zapsání do paměti. Neupravené adresové a datové signály z CPU nejsou vhodné pro paměť, proto je paměťový kontrolér nezbytnou součástí počítačového paměťového subsystému.

Čtení a zápis dat

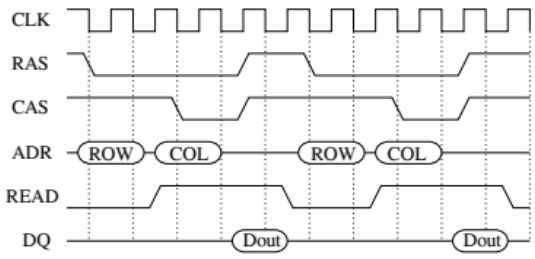
Paměťová buňka má kondenzátor, který udržuje data ve formě elektrického náboje, a přístupový tranzistor, který slouží jako přepínač pro výběr kondenzátoru. Báze (gate) tranzistoru je připojena na adresový vodič. Pole paměťových buněk obsahuje jeden adresový vodič, číslovaný 0 až $2^{N+1} - 1$, na každou zformovanou řadu.

Pole paměťových buněk, kromě adresových vodičů, také obsahuje takzvané páry datových vodičů DATA a $\overline{\text{DATA}}$. Existuje právě jeden pár datových vodičů na každý sloupec v poli paměťových buněk. Datové vodiče jsou střídavě připojeny k emitorům přístupových tranzistorů. Jádrem celé buňky je kondenzátor, který představuje paměťový element jedné buňky. Jedna z jeho elektrod je připojena na kolektor odpovídajícího přístupového tranzistoru a druhá je uzemněna.

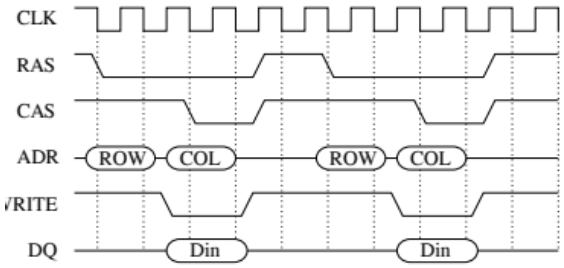
Paměťový kontrolér adresující paměťovou buňku na čipu, nejprve poskytne signál adresy řádku a aktivuje odpovídající adresový vodič. Všechny přístupové tranzistory připojené k tomuto adresovému vodiči se zapnou. Náboje všech kondenzátorů z adresovaného řádku protečou do odpovídajících datových vodičů a do kondenzátorů C_B a dále přes čtecí zesilovače do I/O bufferu. Dekodér sloupce dekóduje přivedený signál adresy sloupce a aktivuje právě jeden datový vodič. I/O buffer zesílí znovu signál dat a předá jej jako výstupní data Dout. Jelikož přístupové tranzistory zůstávají zapnuty, přečtená data se zapíší zpět do paměťových buněk jednoho řádku pomocí kondenzátorů C_B, ve kterých jsou původní přečtená data. Přečtení jedné paměťové buňky tudíž současně vede k občerstvení celého řádku.

Zápis dat je proveden téměř stejným způsobem jako čtení dat. Nejprve paměťový kontrolér poskytne signál adresy řádku, poté aktivuje RAS adresní signál. Ve stejnou chvíli také aktivuje WE řídící signál. Zapisovaná data (*Din*) jsou poslána do vstupního datového bufferu, zesílena a přesunuta do I/O bufferu. Dekodér řádku dekóduje signál adresy řádku a aktivuje odpovídající adresový vodič. Přístupové tranzistory se sepnou a přesunou uložené náboje z kondenzátorů do párů datových vodičů DATA, $\overline{\text{DATA}}$. Potom paměťový kontrolér aktivuje CAS signál a poskytne adresu sloupce do dekodéru sloupce. Data jsou přesunuta z I/O vodičů do odpovídajícího čtecího zesilovače. Potenciály datových vodičů jsou přesunuty zpět do kondenzátorů jako odpovídající náboje.

Obrázek 3 ukazuje chování nejdůležitějších paměťových signálů během vykonávání procesu čtení dat.
Obrázek 4 ukazuje chování nejdůležitějších paměťových signálů během vykonávání procesu zápisu dat.



Obrázek 3: Časové schéma čtení z DRAM



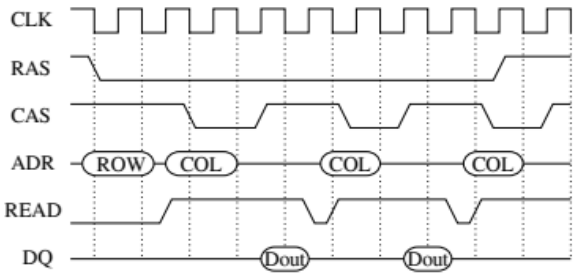
Obrázek 4: Časové schéma zápisu do DRAM

Další operační módy

Minulá sekce popisovala normální mód DRAM. Paměťové čipy mohou také vykonávat jeden nebo více jiných sloupcových módů pro snížení přístupové doby. Ten nejznámější je stránkový mód (Page Mode).

Stránkový mód (Page Mode)

Čtení a zápis dat, se zmiňuje, že v průběhu přístupu do paměťové buňky je nejprve zadána adresa řádku s aktivním RAS signálem a potom adresa sloupce s aktivním CAS signálem. Pokud se další přístup do paměti vztahuje na stejný řádek a pouze jiný sloupec (to znamená, že adresa řádku zůstala stejná a jen adresa sloupce je změněna), není nutné vkládat a dekódovat znovu adresu řádku (ve stránkovém módu je změněna pouze adresa sloupce, ale adresa řádku zůstala zachována). Takže jedna stránka koresponduje právě s jedním řádkem v poli paměťových buněk.



Obrázek 5: Časové schéma čtení z FP DRAM

Během stránkového módu při přístupu do paměťové buňky ve stejném řádku paměťový kontrolér nedeaktivuje RAS signál. Pouze CAS signál je deaktivován na krátkou dobu a poté znovu aktivován. Všechny přístupové tranzistory připojené na adresovém vodiči adresovaného řádku proto zůstanou sepnuty a všechna přečtená data jsou na konci datových vodičů. Nová adresa sloupce je dekódována v dekodéru sloupce. Ve stránkovém módu je přístupová doba o 50% (a doba cyklu až o 70%) kratší, než v normálním módu. Toto samozřejmě platí pro druhý i každý další přístup

EDO mód (Extended Data Out)

V EDO módu časová vzdálenost mezi dvěma následnými CAS aktivacemi je kratší, než u stránkového módu. Adresy sloupců jsou přesunovány rychleji a přístupová doba je výrazně kratší (až o 30% ve srovnání se stránkovým módem) a tudíž přenosová rychlost je proto větší. V EDO módu musí CAS signál být deaktivován před poskytnutím nové adresy sloupce.
Přístupová doba 50 až 60ns.

Vylepšené typy DRAM pamětí

SDRAM

- synchronní DRAM (nezaměňovat s SRAM – statická RAM)
- přístupová doba pouze 8 až 15ns
- mohou fungovat synchronně se systémovou taktovací frekvencí (66 MHz, 133 MHz nebo i více)
- V praxi není rozdíl mezi SDRAM a EDO DRAM podstatný
- Při porovnání s EDO DRAM je patrný větší výkon pamětí SDRAM pokud je systémová taktovací frekvence větší než 100 MHz, což je případ drtivé většiny dnešních systémů
- SDRAM pracují v burst módu a se synchronní taktovací frekvencí, ne s různým RAS a CAS časováním jak tomu je u jiných RAM čipů
- SDRAM také používají odpovídající signály RAS, CAS, WE a CE, ale používají je k přesunutí příkazů jako zápis, čtení a burst stop
- Signály RAS a CAS jsou zkombinovány pro vytvoření příkazové sběrnice
- SDRAM používá principu podobnému prokládání paměťových polí tak, že zatímco s jedním pracuje (je z něj čteno), druhé se připravuje na následující přístup

DDR SDRAM

- Přenosová rychlost dat může být zdvojnásobena, pokud jsou data přenášena nejen na náběžné hraně hodinového pulsu, ale také na sestupné hraně hodinového pulsu. Přesně tento princip používají paměti double data rate DRAM (DDR-RAM). Je to zpětně kompatibilní typ paměti.

Občerstvování DRAM (Refresh)

Postupem času se kondenzátory vybíjejí přes přístupový tranzistor a jeho dielektrické vrstvy. Díky tomu dochází k vybití uložených nábojů a tím i ke ztrátě dat. Kondenzátor musí být pravidelně občerstvován. V průběhu čtení z paměti jsou paměťové buňky adresovaného řádku automaticky občerstveny, protože proces čtení je destruktivní. Normální DRAM musí být občerstvena zhruba každých 10 ms. V současnosti se používají tři občerstvovací metody: RAS-only refresh, CAS-before-RAS refresh a Hidden refresh.

RAS-only refresh

Nejjednodušší a nejvíce používaná metoda pro občerstvování paměťové buňky je vykonání předstíraného cyklu čtení. Během tohoto cyklu je aktivován RAS signál a DRAM se poskytne adresa řádku (adresa občerstvení), zatímco CAS signál zůstává neaktivní. K občerstvení celé paměti je potřeba, aby externí obvod nebo sám procesor poskytl DRAM adresy řádků přesně jak jdou po sobě.

CAS-before-RAS refresh
Pro tento typ občerstvení má DRAM čip svou vlastní občerstvovací logiku s adresním počítadlem. Během CAS-before-RAS refresh je CAS udržován na nízké úrovni po jistou dobu, než RAS klesne na nízkou úroveň (proto CAS-before-RAS). Vnitřní občerstvovací logika je tím aktivována a vykoná automatické vnitřní občerstvení.

Hidden refresh
Zde je cyklus občerstvování "skryt" za normálním přístupem pro čtení. Během skrytého občerstvování je CAS signál udržován na nízké úrovni a pouze RAS signál je přepnut. Protože čas potřebný pro cyklus občerstvování je většinou kratší než cyklus čtení, tento způsob občerstvování šetří čas.

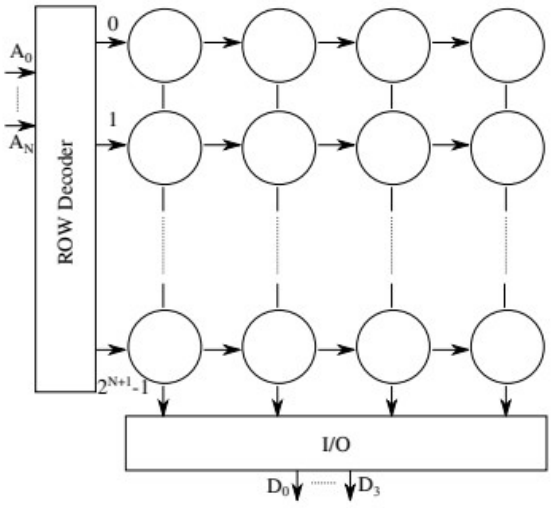
Paměťové moduly
Paměťové čipy jsou nazývány DIP, což znamená Dual Inline Package. Jsou to integrované obvody s vývody na obou stranách. Pro snadnější instalaci pamětí jsou tyto DIP čipy umístěny na modulu. Dnes se používají kompaktní moduly jako SIMM a DIMM než jednotlivé čipy. Pro dosažení patřičné paměťové kapacity je na modul instalován odpovídající počet čipů. Moduly musí být vloženy do soketů, které jsou pro ně umístěny na základní desce.

SIMM moduly
Single Inline Memory Module. Mohou mít DIP čipy na jedné nebo obou stranách a to se 30 nebo 72 piny. Normálně jsou dostupné ve verzi se 72 piny, které podporují 32 bitový přesun dat mezi procesorem a pamětí.

DIMM moduly
Double Inline Memory Module. 168 pinové DIMM mají vždy šířku 64 bitů. Jako DIMM se používají hlavně paměti SDRAM a DDRRAM.

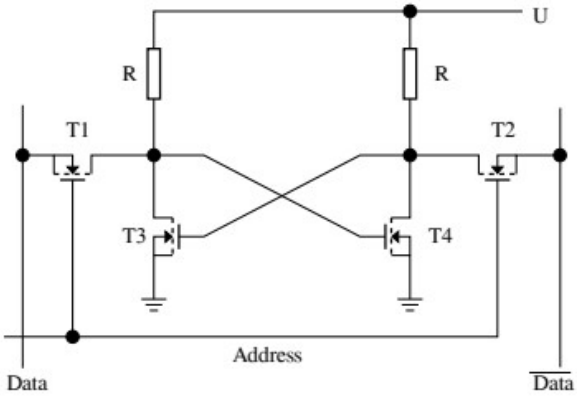
RIMM moduly
Rambus Inline Memory Module. Mají 184 kontaktů a jsou dostupné v kapacitách 64, 128 a 256 Mb. Tyto moduly se krátce používaly v počítačích s procesory Intel P4 kolem roku 2000.

Statické paměti
Ve statických pamětech je informace uložena stavem klopného obvodu. Realizace klopného obvodu je možná pomocí 4 nebo 6 tranzistorů.
Obrázek 9 zobrazuje zjednodušenou organizaci paměti SRAM, kde každé kolečko reprezentuje jednu paměťovou buňku, a ty jsou jsou organizovány jen do 2D mřížky, kde jeden řádek tvoří jedno datové slovo (word). Vodorovné čáry jsou adresové vodiče, které vybírá signál dekodéru řádku (ROW) na základě vstupní adresy $A_0 \div A_N$. Na těchto vodičích jsou připojeny přístupové tranzistory paměťových buněk. Svislé čáry jsou datové vodiče, které přenáší uloženou informaci vybraného řádku do výstupního bufferu I/0.
Paměťovému kontroléru SRAM čipů jsou adresy řádků poskytovány jako jedna informace. A jelikož zde chybí adresní multiplexing, je zapotřebí více pinů a SRAM čipy jsou větší než DRAM čipy. Vnitřní adresování paměťových buněk je tím ale jednodušší a SRAM čipy jsou tak rychlejší než DRAM. A to nejen při adresování, ale i při čtení a zápisu informace.
Díky statickému charakteru paměti není potřeba občerstvování (refresh). Mezi dvěma stavy se klopné obvody přepínají pomocí vnějšího signálu a stav klopných obvodů paměti je udržován tak dlouho, dokud je SRAM čip napájen. SRAM čip je dražší a může obecně pojmou méně dat než DRAM čip kvůli menší hustotě paměťových buněk na jednotku místa. Integrační hustota DRAM čipů je asi čtyřikrát až šestkrát větší než u SRAM čipů za použití stejné technologie. Z tohoto důvodu se SRAM čipy hlavně používají pro malé a rychlé cache paměti, zatímco DRAM čipy pro velké a relativně pomalé hlavní paměti (RAM).

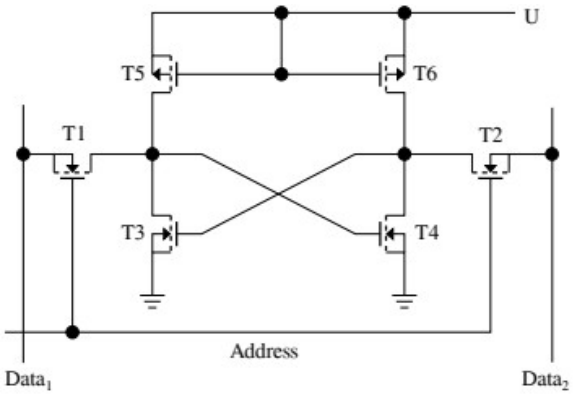


Obrázek 9: Organizace paměti SRAM

SRAM - Static Random Access Memory
Paměťová buňka SRAM
Jak již bylo zmíněno, paměťová buňka SRAM se může skládat ze 4 nebo 6 tranzistorů.
V obou případech se buňka SRAM skládá ze dvou NMOS přístupových tranzistorů T1 a T2 a klopného obvodu se dvěma NMOS paměťovými tranzistory T3 a T4. Dalších dva elementy jsou dvou odpory, nebo dva PMOS tranzistory T5 a T6, které vedle s tranzistory T3 a T4 tvoří CMOS dvojici.
V SRAM jsou paměťové buňky uspořádány v mřížce s řádky a sloupci, které se vybírají dekodéry řádků. Báze (gate) přístupových tranzistorů T1 a T2 jsou připojeny na adresové vodiče a emitory (source) na páry datových vodičů DATA, $\overline{\text{DATA}}$.
Paměti SRAM lze vyrábět i pomocí technologie TTL. Buňka takové paměti funguje na podobném principu jako paměťová buňka používající technologii MOS, ale skládá se pouze ze dvou PNP tranzistorů a dvou odporů.



Obrázek 10: Paměťová buňka SRAM se 4 tranzistory



Obrázek 11: Paměťová buňka SRAM se 6 tranzistory

Čtení a zápis dat
Protože SRAM nevyužívá adresní multiplexing, je adresa do čipu přenesena jako jedna informace. Při čtení i zápisu dat z/do paměťové buňky SRAM aktivuje dekodér řádku odpovídající adresový vodič.
Při čtení se dva přístupové tranzistory T1 a T2 zapnou a propojí klopný obvod paměti s datovými vodiči DATA a $\overline{\text{DATA}}$. Po stabilizaci dat vybere dekodér odpovídající sloupec (to jest odpovídající datové vodiče DATA, $\overline{\text{DATA}}$) a předá na výstupu data do I/O bufferu a tím do vnějších obvodů.
Zápis dat probíhá opačným způsobem. Přes vstupní datový buffer se zapisovaná data zavedou na odpovídající čtecí zesilovač. Ve stejnou dobu aktivuje dekodér řádku adresní vodič a zapne přístupový tranzistor T1. Stejně jako v procesu čtení dat se klopný obvod snaží předat uložená data na datové vodiče DATA, $\overline{\text{DATA}}$. Nicméně čtecí zesilovač je silnější než paměťový tranzistor T3 a poskytne datovým vodičům DATA, $\overline{\text{DATA}}$ signál, který odpovídá zapisovaným datům. Proto se klopný obvod přepne podle nově zapisovaných dat nebo si udržuje již uloženou hodnotu v závislosti na tom, zda se zapisovaná data shodují s uloženými daty nebo ne.

Typy SRAM
Asynchronní SRAM
- Tato paměť existuje od dob procesoru 386 a stále se nachází v pamětech cache L2 mnoha PC. Nazývá se asynchronní, protože není synchronizovaná se systémovými hodinami, a proto CPU musí na data vyžádaná z paměti cache L2 čekat (ne však tak dlouho, jako u DRAM).

Synchronní Burst SRAM
- Synchronní dávková paměť SRAM. Podobně jako SDRAM je synchronní SRAM synchronizovaná se systémovými hodinami.

Pipeline Burst SRAM
- Zřetězená dávková SRAM. Použitím dávkové technologie lze požadavky na SRAM zřetězit, neboli shromáždit je tak, že požadavky v dávce se vykonávají téměř okamžitě. PB SRAM používá zřetězení
- mírně zaostává za systémovými synchronizačními frekvencemi, ale představuje zlepšení proti synchronní SRAM, protože je navržena pro spolupráci se sběrnicemi

Paměti s trvalým obsahem

Nevýhodou paměťových modulů popsaných v předchozích sekcích je jejich neschopnost udržet si uložená data i po odpojení napájení. DRAM a SRAM čipy nejsou vhodné pro startovací proces PC, protože ve chvíli, kdy nejsou zásobovány elektřinou, je jejich obsah zapomenut. Místo toho se používají ROM čipy. Data jsou zde uložena jedním energeticky nezávislým způsobem, takže jsou udržována po dlouhou dobu, i když jsou ROM čipy odpojeny od zdroje energie.
Hlavním úkolem těchto pamětí je pamatovat si data v době, kdy je odpojeno napájení. Z tohoto důvodu se používají například pro uchování BIOSu.

ROM

- Read Only Memory – paměť pouze pro čtení
- buňka paměti je představována elektrickým odporem nebo pojistkou (výrobce některé z nich přepálí, neporušené prvky pak vedou proud, je v nich minimální napětí – tzn. nesou logickou 0, přepálené prvky proud nevedou je v nich maximální napětí – nesou informaci o logické 1)
- informace do nich zapisuje výrobce, doba pamatování není ohraničená

PROM

- programovatelná ROM
- informaci vypaluje uživatel pomocí programátoru
- data se zapisují pomocí elektrického pulsu
- jednou z metod je přepálení pojistky mezi adresovým a datovým vodičem (pojistky jsou vyrobeny z niklu a chromu nebo křemíku)
- stejně jak do ROM, tak ani do PROM není po naprogramování možný zápis
- jiná možnost je realizace PROM za pomocí bipolárních multiemitorových tranzistorů, tento typ PROM se skládá z jednoho multi-emitorového tranzistoru na každý adresový vodič, pokud se má z paměti číst, potom se na jisté adresové vodiče přivede logická 1, multi-emitorový tranzistor se otevře a ve směru kolektor-emitor prochází proud, pokud pojistka nebyla přepálena, potom proud otevře tranzistor, který je připojený jako invertor a na výstup se přečte logická 0. pokud pojistka byla přepálena, potom se tranzistor neotevře a na výstup se přečte logická 1.

EPROM

- Erasable Programmable Read Only Memory - paměť, do nichž je možné opakovatelně zapisovat
- paměťová informace se uchovává pomocí elektrického náboje, ten je kvalitně izolovaný a tak udrží svojí hodnotu i po odpojení elektrického napětí
- k naprogramování je potřeba elektrický pulz trvající 50 ms o napětí + 5 V (u některých typů až + 12 V)
- také EPROM se programuje pomocí speciálního programátoru
- je ji možné vymazat pomocí ultrafialového záření a po vymazání do ní opět zapsat nová data
- EPROM lze poznat podle okénka na pouzdře, kterým vstupuje do paměti mazací ultrafialové záření
- doba pamatování je omezena na 10 až 20 let. Používá se kapacita izolovaného hradla tranzistoru MOS
- Okénko v pouzdře a UV lampa pro mazání dat jsou komplikovaná a taky drahá vybavení pro mazání čipů

EEPROM

- EEPROM - elektricky vymazatelná PROM
- Programování paměťové buňky se provádí stejným způsobem jako u EPROM, to jest relativně dlouhým (50 ms) elektrickým pulzem o napětí + 5 V (nebo + 12 V)
- Pro vymazání se pouze obrátí polarita pulzu
- Počet zápisů a mazání do EEPROM je ohraničený, doba uložené informace je omezena na 10 až 20 let

Flash paměti

- (typ EEPROM paměti - využití jako náhrada za diskety a pevné disky)
- největší výhodou je možnost ji rychle naprogramovat přímo v počítači
- doba uchování uložené informace je nejméně deset let a většinou kolem sto let
- struktura jejich paměťových buněk je v základě stejná jako ta u pamětí EEPROM
- pro mazání a programování je potřeba pulz trvající jen 10 μs a méně napájecího napětí
- vymazání celé paměti je velmi rychlé, možnost vykonat 10 000 a více programovacích a mazacích cyklů, dle typu paměti.
- Adresový buffer přijímá signály adres a přesunuje je do dekodéru řádku a sloupce. Flash paměti, stejně jako SRAM čipy, nevykonávají adresní multiplexing. Dekodéry řádku a sloupce vyberou jeden adresní vodič a jeden nebo více datových vodičů tak jako v běžném čipu. Přečtená data jsou předána na výstup přes vstupně/výstupní datový buffer anebo v případě zápisu jsou zapsána do adresované paměťové buňky 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

Běžné DRAM paměti použité pro video karty většinou nemají dostatečně široké přenosové pásmo pro udržení velkého rozlišení a barevné hloubky při akceptovatelné obnovovací frekvenci. Kvůli tomu byl vyvinut nový typ paměti nazvaný video RAM neboli VRAM.

- **Video RAM (VRAM)**
  - Tato paměť je dvouportová : má dva přístupové porty pro paměťové buňky, jeden se používá pro neustálé obnovování obrazu, druhý pro změnu dat, která se mají zobrazovat. Tyto dva porty tedy znamenají zdvojnásobení kapacity přenosového pásma a v důsledku toho vyšší grafický výkon.
- **WRAM**
  - dvouportovým typem paměti RAM a používá se výlučně pro zvýšení grafického výkonu
  - širší celkové přenosové pásmo (zhruba o 25% než VRAM) a několik grafických funkcí, jež mohou využít tvůrci aplikací
  - vyšší obnovovací frekvence zobrazení
- **SGRAM**
  - Synchronní Grafická RAM v podstatě funguje jako SDRAM
  - SGRAM je optimalizována pro nejvyšší možný přenos dat (SDRAM pro pro nejvyšší možnou paměťovou kapacitu)

FIFO paměti

- realizují se přímo v mikroprocesoru, nebo k dispozici jako stavební členy s různou organizací
- rozdělení na dva typy:
  - **bez přesouvání obsahu** - zápis a čtení z fronty se řídí dvojicí registrů - čte se podle obsahu registru začátku fronty, zapisuje se podle obsahu registru konce fronty, řídící obvody též musejí vytvářet dva důležité stavové signály - fronta prázdná a fronta plná (k předejití podtečení a přetečení).
  - **s přesouváním obsahu** - obvodové realizace fronty s přesouváním obsahu při čtení a při zápisu jsou stejně složité; mají jeden přídavný registr, fronta s probubláváním posouvá asynchronně každou položku po zápisu až do posledního volného místa, přečtením jedné položky ze začátku fronty se jedno místo uvolní, načež je položky stejným mechanismem obsadí. Princip vyžaduje, aby u každého paměťového místa existoval indikátor obsazenosti (klopný obvod)

Cache paměti

Je to jakýsi mezisklad dat mezi různě rychlými komponentami počítače. Jeho účelem je vzájemné přizpůsobení rychlostí - rychlejší komponenta čte data z cache a nemusí čekat na komponentu pomalejší (u které si cache data již načetla).

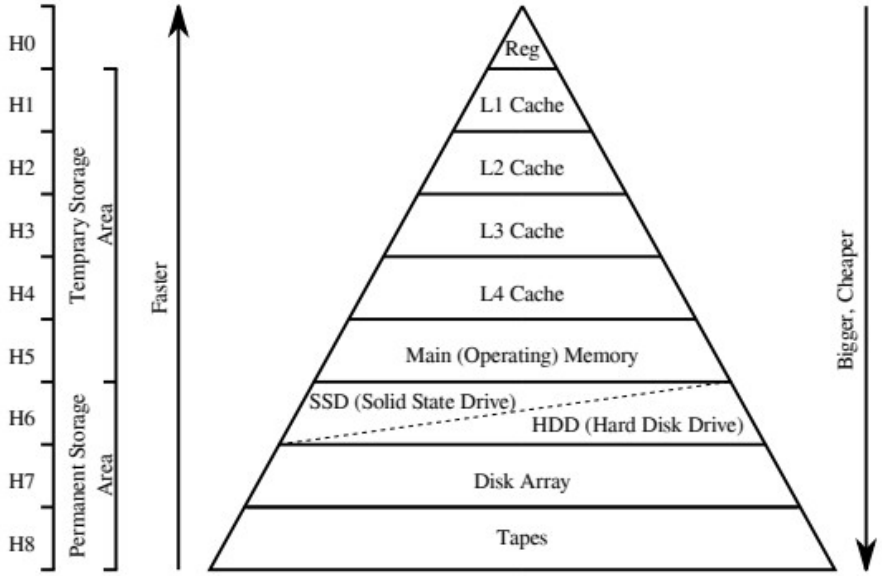
- **Paměti L1 cache**
  - paměť je integrována přímo na procesoru
  - slouží k zásobování procesoru daty ze sběrnice
  - cache přečte více dat ze sběrnice, které potom čekají v tomto meziskladě. Jakmile je procesor potřebuje, přečte si je z cache. Protože cache pracuje rychleji než sběrnice, nemusí procesor čekat, jak by tomu bylo v případě odebírání dat přímo ze sběrnice
- **Paměti L2 cache**
  - umístěna mezi mikroprocesorem a operační pamětí, takže všechna data, která putují mezi těmito dvěma díly, v cache uvíznou a pokud je mikroprocesor znovu potřebuje, přečte si je z rychlejší cache
  - cache je ovládána speciálním řadičem, který se snaží předpovědět, která data bude asi mikroprocesor v nejbližší době požadovat
  - pracuje také jako víceportová paměť, kde mohou být data zároveň zapisována i čtena
  - Cache paměti používají 3 režimy:
    - **Write-Through (zápis skrz cache; přímý zápis) –** nejstarší a nejpomalejší způsob, typický pro mikroprocesory 486, data ukládaná do cache zapisuje současně i do operační paměti, při čtení pak řadič cache porovnává požadované adresy operační paměti s adresami již uloženými, pokud jsou potřebná data v cache nenalezena, jsou z ní přečtena
    - **Write-Back (opožděný zápis) –** novější a rychlejší metoda používaná u Pentií a rychlejších řad 486, data jsou zapisována pouze do cache a teprve při odstranění z cache jsou zapsána do operační paměti, než se data do operační paměti dostanou, mohou v cache několikrát změnit svou hodnotu, v tomto režimu se tedy šetří čas, potřebný na opakované zápisy do pomalejší operační paměti
    - **Pipeline Burst –** nejnovější a nejrychlejší systém práce, nyní běžně používaný, pracuje tak, že provede více operací zřetězeně – pokud čte z určité adresy informaci, přečte zároveň informace i z následujících adres, přístupová doba k datům se pohybuje mezi 9 až 15 ns

Hierarchie pamětí v počítači

Každá technologie má své vlastnosti a výrobní cenu za jednotku kapacity a to ji předurčuje, kde a v jaké velikosti je ekonomicky rentabilní tu kterou paměť použít. Kdyby byla k dispozici technologie, která umožňuje rychlý přístup k datům, uchovává si svůj obsah i po odpojení napájení a je levná, tak by měly počítače pouze jednu paměť a jejich konstrukce by se zjednodušila. Zatím ale taková technologie není k dispozici a stávající konstrukce počítačů je kompromisem mezi cenou a rychlostí a kapacitou.

Paměti jsou v počítači uspořádány do více vrstev a to platí pro všechny počítače. Tomuto uspořádání pamětí se obvykle říká „Paměťová hierarchie“.

Tato ukázka je pouze ilustrativní, protože uspořádání pamětí se bude vždy lišit mezi různými typy počítačů. Jinak to bude u minipočítače, např. Raspberry Pi, jinou konfiguraci bude mít běžný pracovní notebook a něco jiného nalezneme ve výkonnějším serveru, kterému by právě konfigurace na obrázku mohla odpovídat.



Bez ohledu na konkrétní počítač však můžeme hierarchické uspořádání pamětí v počítačích shrnout do několika společných bodů:

– Paměti jsou v počítači uspořádány podle rychlosti, ceny a kapacity, nejrychlejší a nejdražší jsou paměti v CPU, tedy registry na vrcholu pyramidy

– 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 nebo v desítkách 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

Paměť je elektronické uložiště a všechna tato zařízení mají potenciál vracet nesprávné informace odlišné od těch, které byly původně uloženy. DRAM paměti jsou díky svojí charakteristice náchylné občas vracet paměťové chyby. DRAM ukládá jedničky a nuly jako náboje v malých kondenzátorech, které musí být neustále občerstvovány, aby se předešlo ztrátě dat. DRAM je méně spolehlivá než statické úložiště používané SRAM.

Existují dva typy chyb, které mohou nastat v systémové paměti. První se jmenují opakující se nebo tvrdé chyby. V této situaci je část hardware rozbitá a bude neustále vracet chybný výsledek. Je relativně jednoduché takovéto chyby rozpoznat a opravit, protože jsou konzistentní a opakující se.

Druhý typ chyb se nazývá přechodné nebo měkké chyby. Nastávají ve chvíli, když se jednou zpět přečte bit se špatnou hodnotou, ale následně už funguje správně. Tyto problémy jsou pochopitelně hůře rozpoznatelné a také, bohužel, běžnější. Příležitostně se měkká chyba většinou zopakuje, ale to se může stát kdykoliv mezi pár minutami až po několik let.

Jediná skutečná ochrana proti chybám paměti je použít nějaký typ jejich detekce a opravných metod. Některé metody mohou pouze detekovat chyby v jednom bitu z bytu (osm bitů); jiné mohou automaticky detekovat chyby ve více než jednom bitu. Další mohou jak detekovat tak i opravovat chyby.

Parita

Parita se dělí na fyzickou a logickou. Fyzická parita znamená, že paritní bity jsou poskytnuty paměťovému kontroléru během procesu zápisu dat a jsou uloženy v paměťovém modulu. Během čtení paměťový modul předá na výstup informaci o uložené paritě. Když je použita logická parita, paměťový kontrolér generuje paritní bity a poskytne je paměťovému modulu během zápisu, ale modul si tyto paritní bity neuloží. Během čtení jednoduchý obvod v modulu generuje paritní informace z bitů uložených dat. Proto nemůže nikdy nastat paritní chyba; paritní informace poskytované modulem nejsou důležité. Takové moduly ušetří více než 10% (4 až 36 bitů) paměťové kapacity a jsou proto levnější.

Paměťové moduly bývají tradičně dostupné ve dvou typech: s paritou a bez parity. Běžná paměť je bez parity - obsahuje přesně jeden bit paměti pro každý bit uložených dat. Osm bitů je třeba k uložení každého bytu dat. Paměti s paritou přidávají navíc jeden bit pro kažých osm bitů dat, který se používá pouze pro detekování chyb. Devět bitů je třeba k uložení každého bytu. Paměti s paritou mohou použít kontrolu parity, jednoduchou formu detekce chyb. Paměti bez parity neposkytují žádné možnosti detekce chyb.

Kontrola parity

Kontrola parity je jednoduchý způsob pro detekci jednobitových chyb v systémové paměti. Každý byte dat uložených v paměti obsahuje osm bitů realných dat, buď nulu nebo jedničku. Je možné spočítat počet nul nebo jedniček v bytu. Např. byte 10110011 má tři nuly a pět jedniček. Některé byty budou mít sudý počet a některé lichý počet jedniček. Když se do paměti zapíše byte, logický obvod nazvaný generátor/kontrolér parity vyhodnotí tento byte a určí, zda má sudý nebo lichý počet jedniček. Pokud má sudý počet jedniček, potom je devátý paritní bit nastaven na jedna, jinak je nastaven na nula. Výsledkem je, že bez ohledu na to, kolik bylo jedniček v původním bytu, v celých devíti bitech jich bude vždy lichý počet. Tomuto se říká lichá parita.

Během čtení dat z paměti slouží paritní obvod pro kontrolu. Čte všech devět bitů a znovu zjišťuje, jestli je tam sudý nebo lichý počet jedniček. Pokud je počet jedniček sudý, musela být v jednom z bitů chyba, protože při ukládání byl nastaven paritní bit, takže by počet jedniček měl být vždy lichý. Tímto způsobem funguje detekce chyb u paměti s paritou. Systém ví, že v jednom bitu je chyba, ačkoliv neví ve kterém. V případě detekování paritní chyby paritní obvod vygeneruje “nemaskovatelné přerušení” neboli “NMI”, které se většinou používá pro okamžité zastavení procesoru.

Kontrola parity má své meze. Řekněme, že byte dat “00100100” je uložen jako “001001001 1” včetně paritního bitu. Nyní řekněme, že se zpětně přečte jako “01100000 1”. Zde došlo k prohození dvou bitů, nicméně počet jedniček zůstal lichý. Jak lze vidět, parita neumožňuje ochranu proti chybám ve dvou bitech.

ECC paměti

Kontrola parity umožňuje detekovat jednobitové chyby paměti, ale není schopen detekovat vícebitové chyby a neposkytuje žádný způsob opravy paměťových chyb. Z tohoto důvodu byl vynalezen zdokonalený protokol pro detekci a opravu chyb s názvem ECC (Error Correction Code). Tento protokol nejenže detekuje jednobitové a více bitové chyby, ale je schopen opravit jednobitové chyby během čtení.

ECC používá speciální algoritmus pro kódování informací do bloku bitů, které obsahuje dostatek detailů pro obnovu chyby jednoho bitu. Na rozdíl od parity, která používá jeden bit jako ochranu pro osm bitů, ECC používá skupiny bitů: sedm bitů pro ochranu 32 bitů nebo osm bitů pro ochranu 64 bitů.

ECC může detekovat chyby dvou, tří a dokonce i čtyř bitů, ale nemůže je už opravovat. ECC paměť se s těmito vícebitovými chybami vypořádá stejně jako paměť s paritou pomocí nemaskovatelného přerušení (NMI). Více bitové chyby jsou ale v pamětech velice vzácné.

Na rozdíl od kontroly parity ECC způsobí nepatrné zpomalení systémových operací. Důvodem je, že algoritmus ECC je více komplikovanější a chvíli trvá, než ECC opraví nalezené chyby. Z toho důvodu je potřeba vložit jeden čekací stav (wait state) během čtení z paměti. Reálně to znamená snížení výkonu přibližně o 2-3%.

8. Monolitické počítače.

Co to je monolitický počítač, základní konstrukční pravidla a požadavky na monolitické počítače, typická konstrukce M.P.

Pro řadu nenáročných aplikací se již přes 30 let vyrábějí malé počítače integrované v jediném pouzdře. Tvoří tedy jeden celek - monolit. Odtud tedy obecně používaný a zažitý název - monolitické počítače.
V praxi se ale často užívá řada dalších názvů, jako: jednočip, mikročip, mikrokontrolér a mikropočítač. Jde o názvosloví odvozené od zvyklostí různých výrobců a v tomto textu bude použito i poslední uvedené.
(Jednočipové počítače se vyznačují velkou spolehlivostí a kompaktností, proto jsou určeny především pro jednoúčelové aplikace jako je řízení, regulace apod)
- obsahují procesor, paměť a vstupní/výstupní periferie integrované v jednom pouzdře
- obvykle nesou znaky RISC, ale ve zjednodušené podobě, často se pro ně používá Harvardská architektura (lze použít různé paměti pro data a program - ten je často ROM - E(E)PROM nebo Flash)

Architektura monolitických počítačů

Převážně se používá Harvardská architektura. To umožňuje snadno konstruovat počítač, který má paměť dat i programu vyrobenou jinou technologií a dovoluje mít odlišnou velikost nejmenší adresovací jednotky. Dále využívají monolitické počítače převážně rysy architektury RISC, často ve velmi zjednodušené formě.

Organizace paměti a účel: pracovní registr(y), zápisníková paměť, RWM.

V monolitických počítačích lze nalézt 3 základní typy paměti:

pracovní registry

- ve struktuře procesoru jsou obvykle jeden nebo dva.
- ukládají se do nich aktuálně zpracovávaná data a výsledky operací
- nejčastějším operandem strojových instrukcí
- nejsou určeny pro dlouhodobé ukládání dat

univerzální zápisníkové registry

- slouží pro ukládání nejčastěji používaných dat
- instrukční soubor dovoluje, aby se část strojových instrukcí prováděla přímo s těmito registry
- formát strojových instrukcí ovšem obvykle nedovoluje adresovat velký rozsah registrů
- omezení počtu zápisníkových registrů se lze vyhnout implementací několika stejných skupin registrů vedle sebe, s možností přepínání mezi skupinami (těmto skupinám se říká registrové banky)

paměť dat RWM

- slouží pro ukládání rozsáhlejších nebo méně používaných dat
- instrukční soubor obvykle nedovoluje s obsahem této paměti přímo manipulovat (kromě přesunových instrukcí, těmi se data přesunou např. do pracovního registru)
- některé procesory dovolují, aby data z této paměti byla použita jako druhý operand strojové instrukce (výsledek ale nelze uložit přímo zpět do této paměti)

Dále jsou v počítači i **speciální registry**. **Čítač instrukcí,** ukazující na právě vykonávanou instrukci, která je v **instrukčním registru**.
Pro ukládání návratových adres slouží **zásobník.** Je realizován přímo v datové paměti na přesně vyhrazeném místě, nebo jako speciální paměť LIFO (jejíž obsah nelze programově číst). K zásobníku musí být jednoúčelový registr – **ukazatel vrcholu zásobníku**. Hloubka zásobníku je často omezena u mikropočítačů na 2 až 8 úrovní.

Zdroje hodinového signálu, ochrana proti rušení, rozsah napajecího napětí, Brown-Out, Watch-Dog, reset, sleep a stanby režim.

Monolitický počítač, stejně jako všechny sekvenční obvody, vyžadují pro svou činnost zdroj synchronizace.

Zdroj synchronizace

– vytváří časovou základnu pro provádění všech operací v počítači

– často je zdroj integrován přímo v počítači, toto provedení je vhodné tam, kde není nutná přesná doba vykonávání instrukce a není potřebná vazba na reálný čas (vliv teploty na stabilitu, odchylky kmitočtu, ale méně externích součástek)

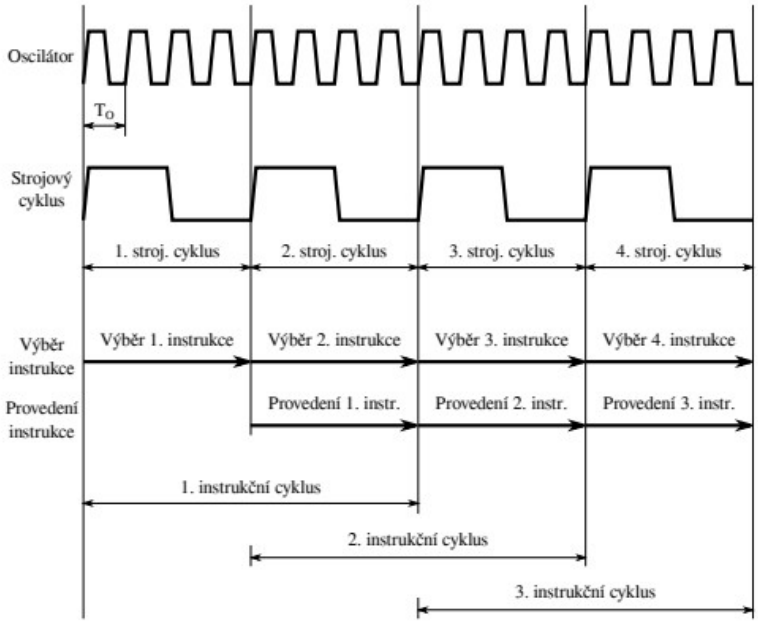
- častěji se používají generátory, umožňující zvolit kmitočet pro řízení mikropočítače (na pouzdře jsou určeny vývody pro připojení časovacích prvků)

-jako externí generátory hodinového signálu se používá: krystal, keramický rezonátor, obvod LC, obvod RC

- pro dobrou stabilitu je vhodný krystal nebo keramický rezonátor, pro běžné aplikace se volí kmitočty v rozsahu 1 až 20 MHz.

- u požadavku na minimalizaci ceny jsou výhodné RC oscilátory (kmitočet se mění nejen s teplotou ale i s velikostí napájení)

Příklad přepočtu hodinového cyklu na cyklus strojový a instrukční:



Obrázek 1: Příklad časových závislostí v počítači

V tomto příkladě se strojový cyklus vypočítá z hodinového dělením čtyřmi. Pro výběr strojové instrukce je třeba právě jeden strojový cyklus, stejně jako pro její vykonání. Instrukční cyklus tedy trvá 2 strojové cykly. Díky principu zřetězeného zpracování, kdy výběr následující instrukce probíhá současně s prováděním aktuální instrukce, je v každém strojovém cyklu dokončeno vykonávání jedné instrukce. Ovšem pokud nedojde k výpadku výběru instrukce, např. při skoku.

RESET monolitického počítače

- počáteční stav počítače
- po provedení RESETu je u všech počítačů nastavena počáteční hodnota čítače instrukcí, obvykle je to hodnota 0 nebo samé jedničky
- výrobce definuje dobu trvání, kterou musí mít signál RESET aby byla provedena správně
- zdroje signálu RESET může být vnější nebo vnitřní (vnější – tlačítko s RC obvodem, vnitřní – např. Schmittovým klopným obvodem)

SLEEP – snížení příkonu mikropočítače (napájecí proud je snížen asi na 20 mikro A)

Brown-Out - obvod hlídající pokles napětí´ (při poklesu pod dovolenou úroveň vyvolá RESET)

Ochrana proti rušení

Jde většinou o ochranu mechanickou. V mnoha případech je mikropočítač vystaven elektromagnetickým vlivům z okolí. Pronikání těchto vlivů do mikropočítače může mít za následek nestabilitu a zničení obvodu. Proto se obvody musí odstínit a jednotlivé vývody i případně galvanicky oddělit od okolí. V neposlední řadě může jít o chyby samotného programátora nebo náhodné rušivé vlivy z okolí.

Vykonávaný program je určitá sekvence instrukcí, která se na základě stavu mikropočítače okolí opakuje. Vlivem okolí se ale může stát, že program občas „zabloudí“. Pro odstranění podobných situací je v mikropočítačích implementován speciální obvod nazývaný WATCHDOG.

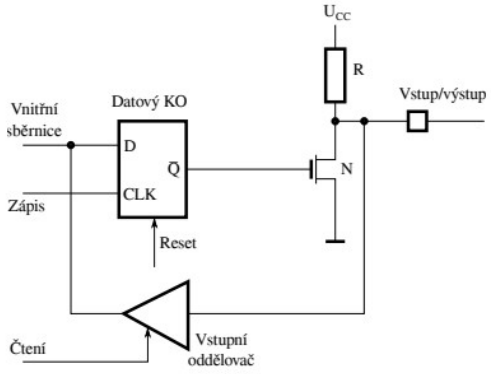
WATCHDOG
– samostatně běžící časovač, který svým přetečením, či podtečením, provede pomocí vnitřního RESETu reinicializaci mikropočítače.
- pro vynulování čítače se používá speciální instrukce mikropočítače (pokud je tato instrukce pravidelně prováděna řídícím programem, k reinicializaci mikropočítače nedojde)

Rozsah napájecího napětí
- účinným způsobem ochrany před rušivými vlivy, je větší **„rozsah pracovního napětí“.**
- pokud je rozsah např. 3 až 6 V a standardní napájecí napětí 5V klesne i pod 4V, nedojde v mikropočítači k žádným nežádoucím změnám hodnot v registrech a nedojde ke kolizi programu

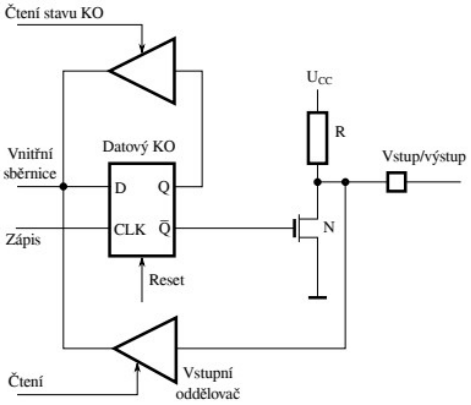
U monolitických počítačů také existuje speciální obvod, který hlídá pokles napájecího napětí pod dovolenou úroveň. Pokud k takovému poklesu dojde, automaticky se provádí RESET.

Typické periférie a jejich vlastnosti: obousměrné I/O porty, čítače, časovače, seriové porty USART a I²C, A/D a D/A převodníky, obvody RT, řadiče LCD.

Periferie - Obvody, které zajišťují komunikaci mikropočítače s okolím
Nejjednodušší a nejčastější používané rozhraní pro vstup a výstup informací je u mikropočítače **paralelní brána – port**
- organizována jako skupina 4 nebo 8 jednobitových vývodů
- lze jednotlivě nastavit, které bitové vývody budou sloužit jako vstupní a které jako výstupní
- brány jsou implementovány tak, že s nimi instrukční soubor může pracovat jako s množinou vývodů nebo jako s jednotlivými bity
Vnitřní strukturu obousměrných vstupně-výstupních vývodů ukazují obrázky 3, 4. K řízení směru přenosu se používá způsob známý ze sběrnicových systémů číslicových počítačů:
analogické spojování logických obvodů s otevřeným kolektorem



Obrázek 3: Struktura V/V bran - otevřený kolektor

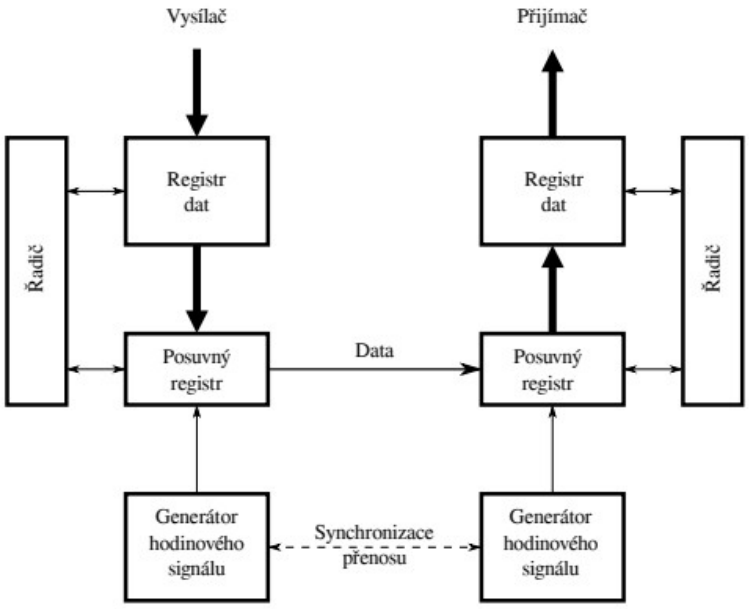


Obrázek 4: Struktura V/V bran - s odděleným KO

Na řízení směru komunikace není nutný samostatný registr. Zapsání logické „1“ do výstupního datového klopného obvodu nesepne tranzistor, a vývod je možno použít jako vstupní.
Obvod na obr. 4 je podobný. Liší se jen tím, že data zpětně čtená po zapsaní do záchytného registru nemusí být stejná, jako data čtená ze vstupu. U obousměrných bran musí vývojáři mikropočítače zajistit, aby se proti sobě neobjevily dva výstupy s opačnou logickou úrovní. Proto jsou obvykle obousměrné brány během RESETu nastaveny do vstupního režimu.

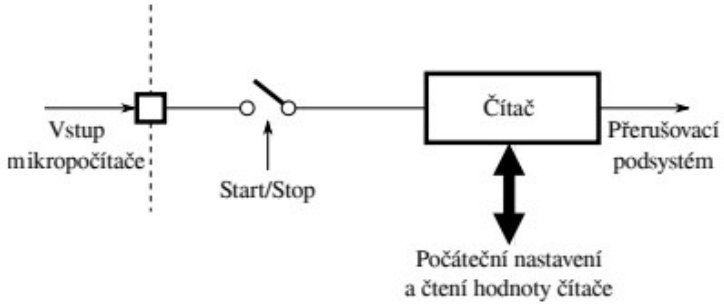
Sériové rozhraní
- přenáší data na relativně velké vzdálenosti při použití minimálního počtu vodičů
Nevýhody: nižší přenosová rychlost (delší doba přenesení informací), nutnost vystřídat na jednom vodiči všechna data (je nutno data zakódovat a na přijímači správně dekódovat)

Podle vzdálenosti, na kterou jsou data přenášena, dělíme komunikaci na dvě základní kategorie:
1. Komunikace mezi elektronickými zařízeními - jedná se o komunikaci na větší vzdálenosti, např. mezi řídícím počítačem a podřízenými stanicemi. Jde o přenos synchronní nebo asynchronní, pomocí RS232.
2. Přenos dat uvnitř elektronického zařízení - používá se pro přenos dat mezi integrovanými obvody jedné aplikace. Typickým standardem je I^2C pro komunikaci mezi integrovanými obvody.
USART může být nakonfigurován v následujících módech:
Asynchronní (full duplex) – V tomto módu je nakonfigurován jako plně duplexní asynchronní systém, který může komunikovat s periferiemi jako jsou např. CRT terminály, osobní počítače (PC) atd..
Synchronní (half duplex) – Druhá možnost je nakonfigurovat USART jako „poloduplexní“ synchronní systém, který může komunikovat např. s periférièmi jako jsou A/D a D/A převodníky, sériová EEPROM atd. Tento synchronní mód můžeme nastavit jako – Master nebo Slave.
I^2C je sériová sběrnice, používána k připojování nízkorychlostních periferií k základní desce, vestavěnému systému nebo mobilnímu telefonu. Pro řízení komunikace se na I2C používá metoda s detekcí kolize. Každá ze stanic může zahájit vysílání, je-li předtím sběrnice v klidovém stavu. Během vysílání musí neustále porovnávat vysílané bity se skutečným stavem SDA (synchronous data). Je-li zjištěn rozdíl mezi očekávaným a skutečným stavem linky SDA, je to indikace kolize mezi několika stanicemi.
Sériový synchronní a asynchronní přenos dat
Každý počítač je dnes vybaven sériovou linkou RS232, u které je možno snadno nastavit parametry přenosu, jako rychlost, parita, počet bitů ve slově a zabezpečení.
Na obr. 6 je blokové schéma sériového přenosu dat jedním směrem.
Při **synchronním** přenosu jsou generátory hodinového signálu vzájemně synchronizovány. Tím je zabezpečena správnost přenosu jednotlivých bitů. Nevýhodou je nutnost synchronizace generátorů, což obvykle vyžaduje další vodič, nebo pomocné obvody, které dokážou synchronizaci udržet rekonstrukcí dat z linky.
U **asynchronního** přenosu není nutná přesná synchronizace posuvných registrů. Tento princip dovoluje jisté malé odchylky u přijímače a vysílače. Sériová linka zůstává při nepřítomnosti dat v klidovém stavu v úrovni logické „1“. Přenos dat se zahajuje START bitem, který je vždy „0“. Přijímač musí detekovat sestupnou hranu na lince a od tohoto okamžiku zahajuje proces příjmu dat. Délka souvisle přenesených dat je pak omezena, aby se udržela vzájemná synchronizace v požadované toleranci. Výhodou je větší volnost synchronizace, ale za cenu nižší přenosové rychlosti, protože s každými daty se musí navíc přenést minimálně START a STOP bit.



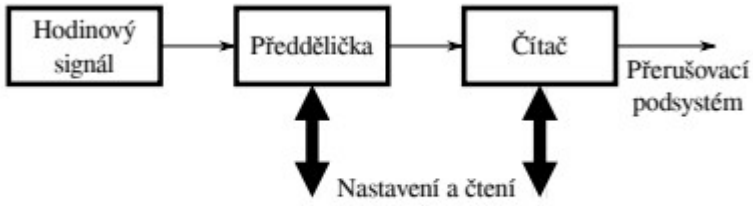
Obrázek 6: Sériový přenos

Čítače a časovače
Do skupiny nejpoužívanějších periférií mikropočítače určitě patří čítače i časovače.
Čítač
- registr o N bitech, který nejčastěji čítá vnější události
- lze nastavit, zda provádí inkrementaci při náběžné, nebo sestupné hraně vnějšího signálu
- při přetečení čítače se obvykle automaticky předává výzva do přerušovacího podsystému mikropočítače
- počáteční hodnota čítače se nastavuje programově a čítač je možno v libovolné chvíli odpojit a opět připojit k vnějšímu signálu



Obrázek 7: Čítač vnějších událostí

Časovač
- příliš se neliší od čítače (není inkrementován vnějším signálem ale přímo vnitřním)
- inkrementován vnitřním hodinovým signálem používaným pro řízení samotného mikropočítače
- při přetečení časovače se může automaticky předávat signál do přerušovacího podsystému mikropočítače
- kromě nastavení počáteční hodnoty, lze programově ještě nastavit předdělička před samotným časovačem a tak lépe přizpůsobit chování časovače



Obrázek 8: Časovač

A/D převodníky
Pro zpracování počítačem však potřebujeme informaci v digitální, tedy číselné formě. K tomuto účelu slouží analogově–číslicové převodníky (analog/digital), kde se převádí analogová veličina, nejčastěji napětí, na číslo.
Komparační A/D převodníky
Princip porovnávání měřené veličiny (napětí) s referenční hodnotou, rozdělenou na několik hodnot v určitém poměru (např. odporovou děličkou). Rozdělením referenčního napětí na několik hodnot dostáváme paralelní převodník. Velmi rychlý převodník, pro zpracování signálů.
A/D převodníky s D/A převodem
Základem těchto převodníků je použití jediného komparátoru a proměnného zdroje referenční hodnoty. Podle způsobu řízení referenční hodnoty rozlišujeme dva základní druhy převodníků: **sledovací** a **aproximační**.

D/A převodníky
Reálné prostředí, do kterého mikropočítače nasazujeme, vyžaduje nejen reakci na naměřené veličiny, ale i zásahy do tohoto prostředí. Tyto řídící podněty musí být často také v analogové formě
Používáme dva základní principy: PWM a paralelní převodníky.
PWM (pulse-width modulation) - šířková modulace pulzů
Princip PWM lze v mikropočítači realizovat pomocí technických prostředků, nejčastěji pomocí čítače, nebo i programově. Počet bitů použitého čítače N určuje rozlišovací schopnost převodníku. Výstupní hodnota může mít 2^N různých úrovní.
Paralelní převodníky – odstraňují základní nedostatek PWM převodníků – velké zpoždění
- jsou založeny na přímém převodu číselné hodnoty na stejnosměrný proud

Obvody reálného času (RTC - Real Time Clock)
- pro potřebu řízení udržuje skutečný čas, tedy hodiny, minuty, sekundy a případně i zlomky sekund, pro řadu aplikací navíc musíme udržovat nejen denní čas, ale i týdny, měsíce a roky
- je nutno vyřešit dva základní problémy při použití RTC:
    - Velká část aplikací pracuje s napájením ze sítě a při výpadku dochází ke ztrátám dat. Tedy i skutečného času. Proto je potřeba zajistit pro zálohování hodnot v RTC obvodu záložní zdroj pro udržení nepřetržité činnosti obvodu.
    - Druhým problémem je čtení dat z RTC obvodu. Čas je hodnota neustále se měnící. Např. pokud zahájíme čtení hodnoty v čase 10:59:59, může se snadno stát, že po přečtení prvních dvou hodnot, v našem případě hodin, se čas posune na 11:00:00, a čtení dalších hodnot bude neplatné. Proto se tyto problémy řeší buď technicky pomocnými registry v RTC obvodu, nebo vhodným programovým řešením.

Speciální periférie
Řada mikropočítačů je navrhována pro speciální účely, a tomu také odpovídají i jejich periférie:
- Pro telekomunikační techniku jsou některé mikropočítače vybaveny dvoutónovým multifrekvenčním generátorem a přijímačem.
- Vysílače a přijímače IR signálu, tedy modulaci, demodulaci a kódování.
- Řadiče LCD a LED zobrazovacích jednotek, které vyžadují dynamické řízení

Účel a použití systému přerušeni
- dovoluje výrazně efektivnější programové odezvy na změny stavu periferií a chování okolí.
- požadavky na přerušení přicházejí zcela asynchronně s činností hlavní programu, proto může dojít k přerušení programu kdykoliv.
- řízení procesoru se přenese definovaným způsobem na obsluhu přerušení, kde se musí v co nejkratším čase zajistit reakce na požadovaný podnět a vrátit řízení zpět do hlavního programu. Přitom obsluha přerušení musí uložit všechny systémové registry a na konci je vrátit do původního stavu, aby neměla žádný nežádoucí vliv na chod hlavního programu.
Pro programátora je důležité zajímat se, jak povolit a zakázat přerušení, jak detekovat, co je zdroj požadavku na přerušení a co musí povinně provést při obsluze přerušení (je to dáno výrobcem). Na některých mikropočítačích si také může určovat prioritu jednotlivých zdrojů přerušení a také je možné vnoření obsluhy více přerušení současně.

Charakteristika sady strojových instrukcí
-aritmetické instrukce, logické instrukce -instrukce přesunu, instrukce pro práci se zásobníkovou pamětí, instrukce skoku a volání podprogramu, instrukce pro práci s registry, instrukce pro posun dat, speciální instrukce (povolení přerušení, zákaz přerušení programová inicializace systému (RESET) atd.)
Samostudium: detailní popis jednoho vybraného mikropočítače s RISC jádrem.
ARM architektura

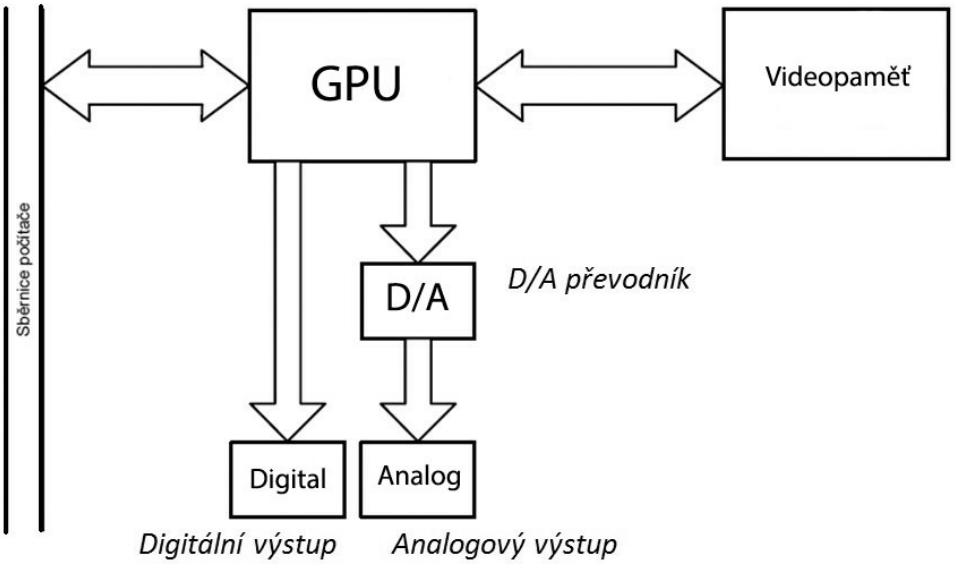
- Procesor ARM má 44 základních instrukcí a všechny mají jednotnou šířku 32 bitů.
- Pouze arit.-logické instrukce s registry, případně s přímými operandy, jsou vykonávány v jednom taktu.
- V procesoru je realizováno 3-stupňové zřetězení instrukcí
- Procesor je schopen činnosti **ve čtyřech režimech** (v jednom uživatelském a třech privilegovaných):
    - v režimu uživatelském USR, v režimu supervizora SUP, v režimu přerušení IRQ, v režimu rychlého přerušení FIQ.
- Procesor ARM obsahuje množinu 25 částečně se překrývajících 32 bitových registrů, přičemž programově přístupných je v každém režimu činnosti procesorů pouze 16 registrů.
- Zkrácení doby odezvy procesoru je dosaženo, v případě režimu rychlého přerušení FIQ, použitím čtyř lokálních univerzálních registrů a jednoho registru s návratovou adresou. Tyto registry mohou obsahovat všechny ukazatele a různé čítače používané v jednoduchých procedůrách obsluhy "Vstupu/Výstupu", takže lze dasáhnout velmi rychlého opakovaného přepínání procesoru mezi režimem "Uživatelským" a režimem "Rychlého přerušení".
- Procesor ARM poskytuje 26 bitovou adresu lineárního paměťového prostoru, což **umožňuje adresovat 64 MB fyzického paměťového prostoru**.
- Všechny instrukce obsahují **čtyřbitové pole podmínky**, které determinuje vykonání této instrukce. **Vykonání všech instrukcí je podmíněno rovností kódu podmínky**

9. Videoadaptéry, monitory.

Základní blokové schéma videoadaptéru (řadič, paměť, D/A převodník), princip činnosti.

Grafický adaptér funkce:
- vytváří ve videopaměti obraz
- obsah videopaměti se zobrazuje v zobrazovací jednotce
- aplikační program komunikuje s grafickým adaptérem pomocí rozhraní API (Microsoft Direct X nebo OpenGL)

Hardware grafického adaptéru:



Funkce:

GPU (Graphics Processor Unit) – řídí činnost grafického adaptéru, zajišťuje tvorbu obrazu

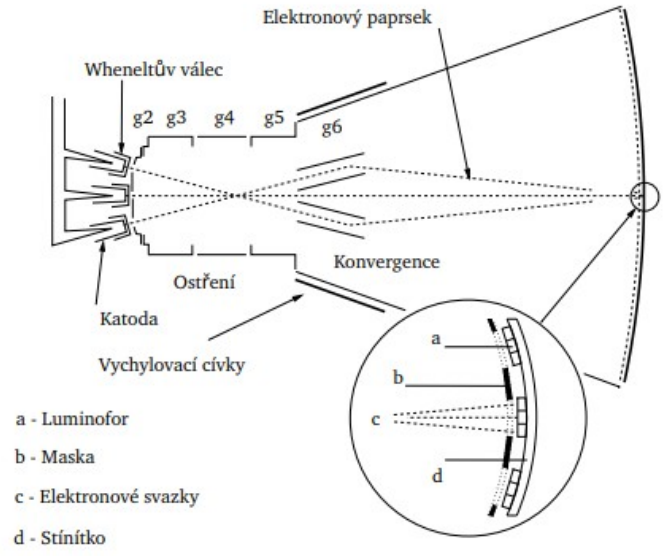
Paměť – GPU do ní ukládá hotový obraz

D/A převodník – digitálně analogový převodník, který převádí digitlní obraz z operační paměti karty na analogový signál a ten vstupuje do monitoru

Reprezentace analogových veličin v digitální podobě.

Tvorba obrazu začíná v grafické kartě počítače. Digitální signály z operačního systému nebo aplikačního software jsou přijímány adaptérem VGA. Tento signál je digitální a je třeba ho nejprve převést na signál analogový prostřednictvím digitálně–analogového převodníku (DAC – Digital to Analog Converter), kterému monitor dokáže porozumět. Obvody DAC jsou většinou uloženy na jednom specializovaném čipu, který ve skutečnosti obsahuje převodníky tři – pro každou ze tří základních barev používaných na displeji (RGB). Obvody DAC převádějí číselné hodnoty zasílané počítačem na analogové tabulky, které obsahují potřebné úrovně napětí pro tři základní barvy. Tyto barvy jsou nutné k namíchaní barvy jednoho bodu.

Principy tvorby obrazu na monitorech s klasickou obrazovkou a na LCD.



Obrázek 1: CRT displej

Obrazovka typu CRT (zkratka pro cathode ray tube) je zobrazovací zařízení, které funguje na principu katodové trubice (typ urychlovače elektronů) se stínítkem. Tato trubice je uzavřena do vakuové baňky. Obraz je vytvořen pomocí tří svazků urychlených elektronů, jeden pro každou barvu základního spektra RGB. Pro černobílé obrazovky stačí pouze jeden svazek. Svazky elektronů jsou vychylovány elektromagnetickými silami pomocí cívek. Elektrony dopadají na stínítko pokryté luminoforem.

Výhody a nevýhody CRT displejů
Výhody: Ostrost, zobrazení věrohodných barev, doba odezvy, pozorovací úhly.
Nevýhody: Velikost, spotřeba, vyzařování, hloubka.

Princip LCD displeje
LCD (Liquid Crystal Display) je zobrazovací panel, který funguje na principu tekutých krystalů. Tekutý krystal je látka, která dokáže setrvat jak v kapalném a pevném skupenství, tak vytvořit krystalickou strukturu. To vše je závislé na elektrickém náboji, který je krystalu dodáván. Tyto krystaly jsou vloženy mezi průhledné elektrody a polarizační filtry. Elektrickým nábojem je řízeno natočení těchto krystalů, a díky tomu dochází k řízení průchodu a polarizaci světla. Zobrazovaná jednotka obrazu, jeden bod, má název pixel. Ten se skládá ze subpixelů, které jsou v každém pixelu 3. A to klasické RGB (červený, zelený, modrý). Kombinací rozsvěcování těchto subpixelů lze dosáhnout vykreslení všech barev spektra. K odfiltrování nežádoucích barev zde slouží i polarizační filtry, které dovolují filtrovat barvy, které jsou nežádoucí.
Nejstarší a stále nejrozšířenější technologie zobrazení v LCD displejích je
TN (twisted nematic) - U displeje je krystal v klidovém stavu, pokud na něj není přivedeno napětí, a pak pixel propouští světlo. Pokud je na elektrody krystalu přivedeno napětí, ten se šroubovitě stočí a zamezí prostupu světla. Princip je efektivní hlavně po stránce energetické náročnosti, kdy je potřeba napětí pouze na tmavé barvy.
Výhody oproti IPS – rychlá odezva a nízké výrobní náklady
IPS - U IPS LCD je princip opačný než TN. Krystaly v klidovém stavu bez přivedeného napětí nepropouští světlo. Při elektrickém impulzu se krystal otočí o 90 stupňů, propustí světlo a pomocí polarizace se dosáhne potřebné barvy.
Oproti TN dobré podání barev a skvělé pozorovací úhly. Je ale dražší a občas má problémy se sléváním tmavších barev.
Barevné LCD
Konstrukce barevných displejů je téměř stejná jako u jednobarevných.
Každý bod displeje se skládá ze tří menších bodů, které obsahují červený, zelený a modrý filtr. Propouštěním světla do barevných filtrů a jeho složením se dosáhne výsledná barvy.
Parametry LCD monitorů: Úhel pohledu, doba odezvy, kontrast.
Výhody: Kvalita obrazu, životnost LCD displeje, spotřeba energie, odrazivost a oslnivost, bez emisí.
Nevýhody: Citlivost na teplotu, pevné rozlišení, vadné pixely, doba odezvy.

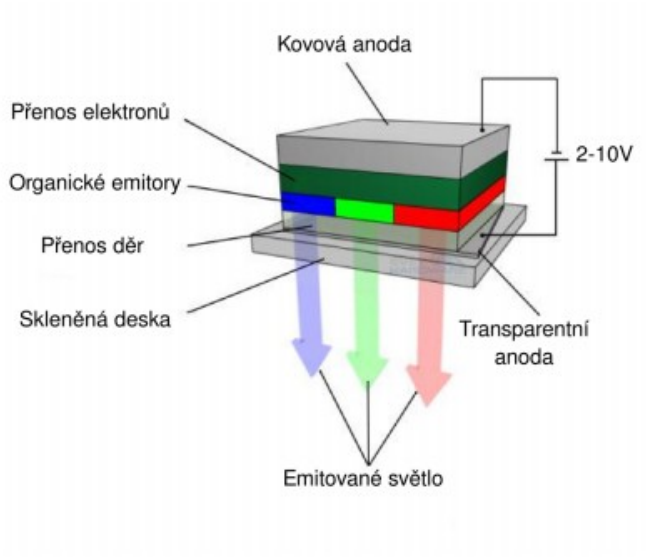
Obnovovací frekvence, šířka pásma.

Obnovovací frekvence - frekvence s jakou se obnovuje obraz na obrazovce. (Hz)
Šířka pásma - zjednodušeně se dá říct, ze čím vyšší kmitočet, tím ostřejší obraz (při daném rozlišení a obrazové frekvenci (MHz)
- kmitočet, jímž jsou zobrazovány body na obrazovce
- Pro VGA adaptér s rozlišením 640 x 480 stanovila firma IBM šířku pásma 25,175 MHz →každou vteřinu se muselo zobrazit více než 25 mil. Bodů
- Vyšší rozlišení →je potřeba přenést větší počet bitů →větší šířka pásma (podobně, pokud chceme zobrazovat větší počet snímků).
- S větší šířkou pásma souvisí i možnost zobrazovat větší počet snímků –vyšší kvalita zobrazení

OLED, E-Ink.

Zkratka OLED bývá interpretována dvěma způsoby - „Organic Light Emmiting Diode“ a „Organic Light Emmiting Display“. Pokud hovoříme o použité technologii, pak je první výklad vhodnější. Hlavním prvkem těchto zobrazovacích jednotek je organická dioda emitující světlo.

Princip činnosti OLED



Obrázek 10: Základní struktura OLED displeje

Klasický OLED displej tvoří průhledná anoda a kovová katoda, mezi kterými je několik vrstev organické látky. Jsou to vrstvy emitující díry, přenášející díry, vyzařovací vrstva a vrstva přenášející elektrony. V momentě, když je do některého políčka přivedeno napětí, jsou vyvolány kladné a záporné náboje, které se spojují ve vyzařovací vrstvě, a tím produkují světelné záření. Z toho plyne, že OLED displeje sami vyzařují světlo, nepotřebují tedy podsvícení jako LCD displeje. Struktura a použité elektrody jsou uzpůsobeny, aby docházelo k maximálnímu střetávání nábojů ve vyzařovací vrstvě. Proto má světlo dostatečnou intenzitu.

Vlastnosti OLED

OLED displeje jsou samy o sobě zdrojem světla a nepotřebují podsvětlení. Díky tomu dosahují vysokého kontrastu a jsou energeticky méně náročné, než LCD.

Výhody

Vysoký kontrast, velmi tenké, plně barevné, nízká spotřeba, dobrý pozorovací úhel, prakticky bez zpoždění, snadná výroba, možnost instalace na pružný podklad.

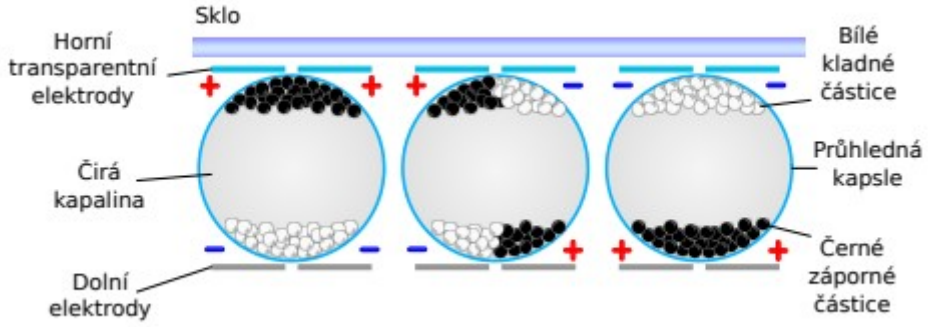
Nevýhody

Nevýhodou je vyšší cena a v současné době převažují jen malé displeje pro mobilní telefony.

E-Ink

Pro zařízení, která k zobrazení statické informace nepotřebují elektrický proud se používá označení EPD - Elektronic Paper Device, tedy elektronický papír. Jako jedna z nejznámějších technologií používaných pro tato zařízení je elektronický inkoust - E-Ink.
E-Ink je plochá zobrazovací jednotka, která odráží světlo jako normální papír, je schopna uchovat text i obrázky natrvalo bez spotřeby elektřiny, s možností změny obsahu a často je také ohýbatelný. Z důvodu nízké energetické náročnosti a tenkosti se stále více prosazuje ve čtečkách elektronických knih. Elektronický papír lze číst na slunci nebo pod lampou.

Popis technologie E-Ink
Celý displej je umístěn na TFT (Thin Film Transistor) matrici a je složen z malých částí - mikrokapslí naplněných médiem (bezbarvá olejovitá kapalina), které obsahují stovky částeček bílého a černého pigmentu. Bílé částečky jsou nabity pozitivně, černé negativně. Podle potřeby lze ve spodní části mikrokapsle vytvořit pozitivní nebo negativní elektrické pole. Opačně nabité částečky jsou k němu přitaženy a v horní části mikrokapsle zůstanou ty se stejnou polaritou, jako elektrické pole.

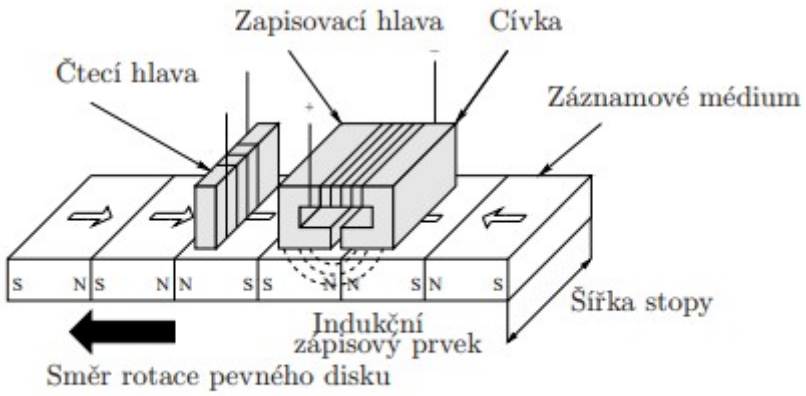


Vzhledem k tomu, že zařízení nepotřebuje podsvětlení, je vytvořený obraz velmi kontrastní a dobře čitelný i na slunečním světle. Má také velký pozorovací úhel - téměř 180° . Pokud k tomuto přidáme i další výhodu, a tou je velmi nízká spotřeba elektrické energie, ve srovnání s LCD a OLED o 2 až 3 řády, navíc pouze při překreslení obsahu, máme k dispozici to, co bylo uvedeno na začátku: elektronický papír.
Problémem ale zůstává poměrně velké zpoždění při požadavku na změnu obsahu stránky, řádově stovky ms. Tato technologie je tedy nevhodná pro zobrazování videosekvencí.
Slabinou je také malý stupeň odstínů šedé barvy. V současnosti je možné vytvořit obvykle jen 16 odstínů.
Existuje i barevná verze E-Ink. Používá se stejná metoda tvorby barev jako u LCD - barevné filtry. Před každou kapsli se umístí vždy jeden barevný filtr z trojice RGB barev. Výsledek ovšem nedává příliš věrné barvy. Omezena je i barevná hloubka na 4096 (16^3) barev.

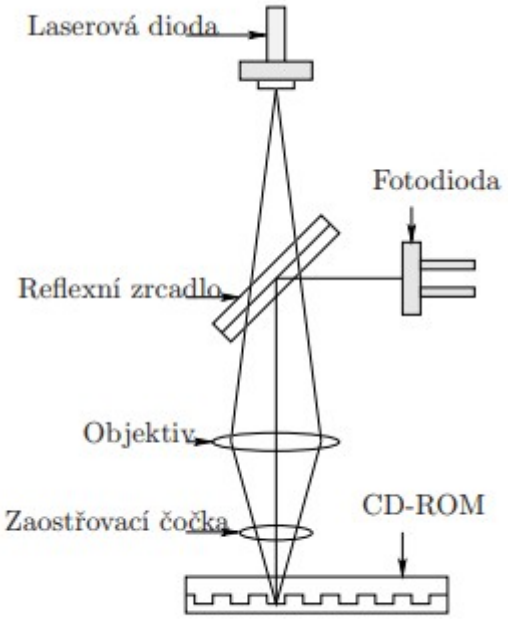
Výhody
Vysoké rozlišení, dobrý kontrast, čitelnost na přímém slunci, není nutné podsvětlení, velký pozorovací úhel, velmi tenké, možno používat i na pružném podkladu, nulová spotřeba proudu při zobrazení statické informace, minimální spotřeba při překreslení.
Nevýhody
Málo odstínů šedi a tedy i následně špatné barevné rozlišení, velké zpoždění při překreslování.

10. Diskové paměti.
Disková paměťová média. Princip získávání a ukládání informací s magnetickým, optickým a magneto-optickým přenosem.
Paměti magnetické

- **Princip ukládání dat**
- Vnější 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 s povrchem média nebo jen v nepatrné vzdálenosti od něj je štěrbina (o šířce cca 1 mikrometr) magnetického obvodu. Ten tvoří jádro (přerušené právě touto štěrbinou), na němž je navlečená cívka. Magnetický tok, který prochází jádrem (při průchodu elektrického proudu cívkou) se v místě štěrbiny rozptyluje do nejbližšího okolí a uzavírá přes vzduchovou mezeru. Jeho rozptýlené siločáry zasahují i do aktivní vrstvy záznamového materiálu, kterou magnetizují.
  Pokud se mění směr elektrického proudu v cívce, mění se směr magnetického toku jádrem i vzduchovou štěrbinou a tím i smysl magnetizace vrstvy. Když se vrstva pohybuje, vznikají v ní oblasti magnetované tím či oním směrem a mezi nimi místa magnetických změn. Těm říkáme místa magnetických reverzací. Právě jejich přítomnost na médiu představují zapsanou informaci. Při čtení se médium pod hlavičkou pohybuje stále stejným směrem. Magnetické reverzace nyní představují změny magnetického pole vyvolávané magnetickým médiem. Tyto změny způsobují vznik napěťových impulsů na svorkách cívky, které pak dále zpracovávají elektronické zesilovače.



Obrázek 1: Princip magnetického zápisu



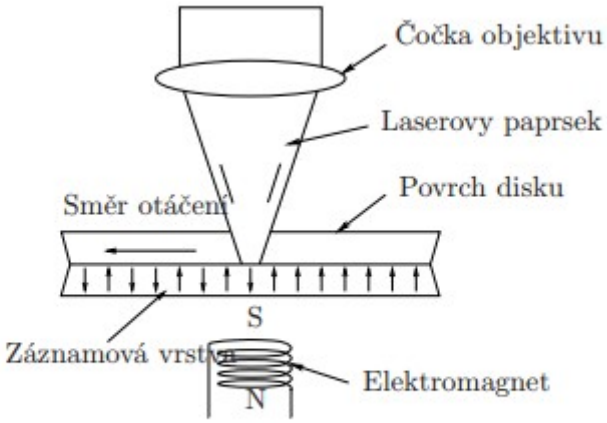
Obrázek 5: Princip optického záznamu

Paměti optické

- **Princip ukládání dat**
  Ze zdroje laseru se vysílá paprsek, který prochází přes polopropustné zrcadlo, objektiv a zaostřovací čočku a dopadá na stopu CD. Dopadne-li na pit dojde k jeho rozptylu, dopadne-li na pole odráží se, vrací se zpět do čtecí hlavy a na polopropustném zrcadle mění svůj směr. Dopadá na fotodetektor, kde vznikají elektrické impulsy. Střídáním pitů a polí vznikají sekvence impulsů, které odpovídají zapsaným datům.

Paměti magneto-optické

- **Princip ukládání dat**
  Zápis informací na magnetooptický disk je prováděn pomocí zapisovací magnetické hlavy a laserového paprsku. Feromagnetická vrstva na magnetooptickém disku je totiž vytvořena z materiálu, který je za běžných pokojových teplot možné zmagnetizovat pouze působením velmi silného pole, který zapisovací hlava nemůže vyvinout. Pokud se však teplota feromagnetické vrstvy zvýší nad takzvanou Curierovu teplotu, překoná náhodný pohyb atomů vnitřní síly působící spontánní magnetizaci a z feromagnetické látky se stává látka paramagnetická, na níž už relativně slabé magnetické pole zapisovací hlavy může působit. Teplota feromagnetické vrstvy je zvyšována laserovým paprskem, který je velmi přesně zaostřen právě na tuto vrstvu a dokáže zvýšit teplotu zápisového místa (to má velmi malou plochu) na cca 180 °C. Tento způsob záznamu umožňuje, aby byla zapisovací hlava umístěna v poměrně značné vzdálenosti od disku, protože případný rozptyl magnetického pole neovlivní okolní bity – na ně totiž laserový paprsek nesvítí, tudíž nemají dostatečnou teplotu.



Obrázek 8: Záznam dat u magnetooptických disků

Organizace dat na pevných discích, CD-ROM a DVD mediích.

Pevný disk

Uzavřená jednotka v počítači, používaná pro trvalé ukládání dat (tzv. data zůstanou na disku zachována i po vypnutí napájení počítače). Pouzdro chrání disk před nečistotami a poškozením. Pevný disk obsahuje pevné plotny diskového tvaru, které jsou většinou vyrobeny ze slitiny hliníku nebo skla. Plotny na rozdíl od disket nelze ohýbat, proto termín pevný disk. Ve většině případů platí, že plotny nelze vyjmout. Z těchto důvodů se někdy používá termín pevná disková mechanika (fixed disk drive) nebo hard disk drive HDD.

Geometrie disku

Pod pojmem geometrie disku se skrývá uspořádání prostoru na disku, konkrétně počet hlav, cylindrů a stop. Data jsou na disk ukládána v bajtech. Bajty jsou uspořádány do skupin po 512, nebo-li sektorů. Sektor je nejmenší jednotka dat, kterou lze na disk zapsat nebo z disku přečíst. Sektory jsou seskupeny do stop. Stopy jsou uspořádány do skupin zvaných cylindry nebo válce. Předpokladem je, že disk má nejméně dva povrchy. Systém adresuje sektory na pevném disku pomocí prostorové matice cylindrů, hlav a sektorů.

CD – ROM

Data jsou na povrchu disku zaznamenávána do stopy, mající tvar spirály. Stopa je dále rozdělena na sektory (rychlost čtení 75 sektorů/s). Sektory se dělí na 98 rámců informací z nichž každý obsahuje 33B. Kapacita jednoho sektoru je 3234B.

DVD

Stopa ve formě spirály. Stopa je tvořena sektory, z nichž každý obsahuje 2048B dat. Při zápisu jsou nejprve data zformátována do datových rámců o velikosti 2064B. Z toho 2048B pro data, 4B – identifikační informace, 2B – obsahují informace pro detekci chyb v identifikačních informacích, 6B – pro ochranu autorských práv, 4B - obsahují kód pro detekci a opravu chyb pro celý rámec.
Kapacita DVD se pohybuje od 4,7GB u jednostranných jednovrstvých médií do 17,1GB u dvouvrstvých oboustranných médií.

Orientační informace o přístupových dobách, přenosových rychlostech a kapacitách jednotlivých medií.

Přístupová doba

- Jedná se o průměrnou přístupovou dobu k libovolnému místu na disku v milisekundách. Je to čas potřebný k tomu, aby se čtecí nebo záznamová hlava dostala nad určený sektor. Čím nižší je tato hodnota, tím se zvyšuje reálný výkon disku.
- pevný disk - několik ms (od 4 ms, obvykle 8 ms, lacinější přes 10 ms)
- CD – ROM – 140 ms (100 ms do 300 ms)
- DVD – ROM – 160 ms

Kapacita CD je 734 MB, což postupem času přestalo vyhovovat a začala se objevovat DVD s kapacitou 4,7 GB. Pevný disk – 500GB – jednotky TB

Přenosová rychlost

– Pevný disk - desítky MB/s (náhodný přístup), přes 150 MB/s (sekvenční čtení) u M.2 disků až jednotky GB/s
- Rychlost mechaniky typu DVD se udává jako násobek 1350 kB/s, což znamená, že mechanika s rychlostí 16× umožňuje přenosovou rychlost 16 × 1350 = 21600 kB/s (nebo 21,09 MB/s).
- CD – násobky 150 KB/sec

11. GP GPU, CUDA

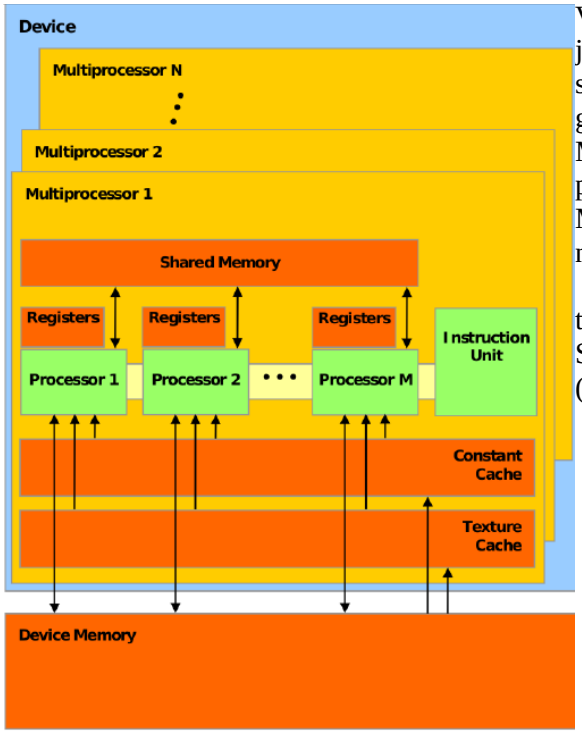
Historie GPU

1970 - V 8bitových počítačích Atari byly používány čipy ANTIC. Čip Antic byl speciálně určen pro mapování textu a grafických dat do videovýstupu.
1980 - grafický systém 8514 od IBM byl jeden z prvních systémů PC videokaret provádějící jednoduchou 2D grafiku.
1993 – založení firmy NVIDIA
1995 – první grafický čip NV1 od NVIDIE
1996 – 3dfx vydali Voodoo Graphics
1996 – GeForce 256 – Název byl odvozen od slov "Geometry" a "Force", značících vysoký výkon ve zpracování geometrie a číslo 256 znamenalo 256-bitovou vnitřní architekturu. Tento čip je považován za první opravdový grafický procesor (GPU), protože se jedná o první čip s jednotkou pro kompletní zpracování geometrie.
2000 – sjednocení NVIDIA a 3dfx
2002 – GeForce 4 - vybavené shadery pixelů a vrcholů.
2006 – Nvidia CUDA - jednotná výpočetní architektura (nerozlišuje pixely a shadery vrcholů)
The first GPU computing was the indirect computing over the OpenGL programming interface.

co to je CUDA (Compute Unified Device Architecture)

Hardwarová a softwarová architektura. Umožňuje na vybraných grafických kartách provádět výpočty.
- není potřeba znát architekturu konkrétní grafické karty, problémy se nemusí definovat v jazyce grafiky (textury), odpadá nutnost znalosti DirectX, OpenGL – programování je jednodušší (není třeba znát technické podrobnosti)
- funguje pouze na kartách od Nvidie (nevýhoda)
- poskytuje programovací rozhraní pro často používané jazyky - C/C++, Fortran, Python
- optimalizováno na matematicky náročné výpočty (zpracování obrazu, problémy definované pomocí mřížek), lze použít větvení, ale to degraduje výkonnost programu
- grafické karty obsahují velké množství ALU a malé množství řídících obvodů
Výhody: velký výpočetní výkon v podobě masivního paralelismu (velký počet malých procesorů, který zpracovává program), všechny vlákna jsou nezávislé, vhodné pro dlouhé, složité výpočty (nejlépe bez IF podmínek), GPU je optimalizované pro sekvenční přístup do hlavní paměti grafické karty. Přenosová rychlost sběrnice je až stovky GB/s, většina transistorů v GPU jsou navrženy pro výpočty.

organizace GPU z pohledu CUDA



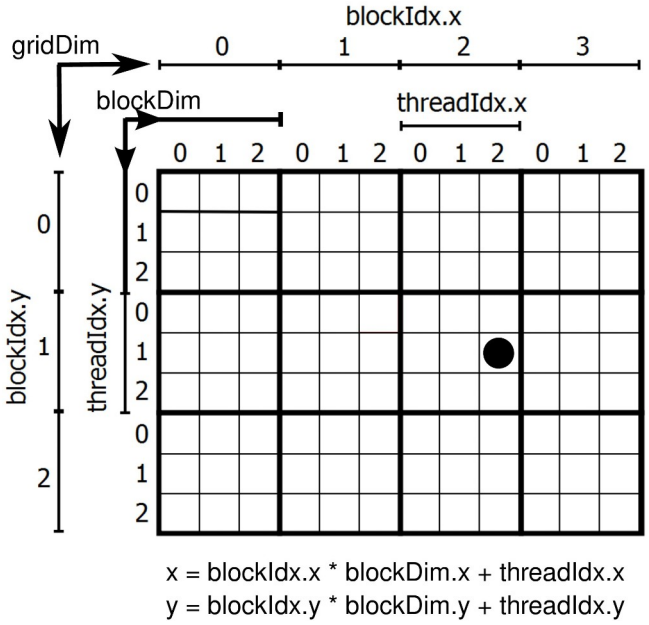
Využívá pojem multiprocesor a taky se používá tzv. CUDA jádro. Výrobce sdružuje malé množství procesorů, kterým se říká CUDA jádra na multiprocesoru. Podle výkonu grafické karty se odvíjí počet multiprocesorů. Multiprocesory mezi sebou komunikují pouze prostřednictvím paměti grafického adaptéru (Device Memory – thats where they share data), takže jsou nezávislé na sobě.
Každý multiprocesor má cache pro urychlení přístupu k texturám a paměti konstant. Oboje cache jsou readonly.
Sdílená paměť – mohou k ní přistupovat všechny procesory (vlákna) ve stejném bloku (multiprocesoru)

architektura GPU některé z posledních generací.
Nvidia Pascal GP100

64 Cuda jader, rozdělena do dvou bloků, každý blok obsahuje 32 jader, Instruction Buffer, Warp plánovač a dvě dispatch jednotky. Instrukce tečou dolů přes instrukční mezipaměť, do instrukční vyrovnávací paměti, jsou naplánovány do dispatch jednotky, která pak provádí instrukci v příslušném CUDA jádru. The dispatcher je zodpovědný za zajištění spuštění vláken. Úkol warp plánovače je jednoduchý: Každý cyklus se snaží naplnit obě dispatch jednotky. Pokud nemůže, zastaví jednu nebo obě dispatch jednotky.
LD/ST (load/store) - operace s adresou paměti,
SFU (special function units) - provádění (transcendentálních) matematických instrukcí, hierarchie paměti a plánovače warp

Organizace vláken, mřížka, bloky, identifikace vlákna, postup výpočtu - kooperace GPU a CPU.

Všechna vlákna jsou uspořádána v mřížce (viz níže) a každé vlákno v této mřížce má přesnou polohu. Každé vlákno má předdefinované proměnné, aby se mezi ostatními identifikovali.



Předdefinované proměnné:

- **dim3 gridDim** obsahuje rozměr celé mřížky
- **uint3 blockIdx** obsahuje pozici bloku v mřížce
- **dim3 blockDim** obsahuje rozměr bloku (všechny bloky v mřížce mají stejný rozměr)
- **uint3 threadIdx** pozice vlákna v bloku.
- **int warpSize** obsahuje the warp size ve vlákně.

Když nějaké vlákno z definované mřížky potřebuje znát svou vlastní pozici, musí pomocí předdefinovaných proměnných správně identifikovat, která část problému bude počítána právě v tomto vláknu a jádře.
Každé vlákno musí používat předdefinované proměnné blockDim, blockIdx a threadIdx. Pak je nutné použít následující vzorce:

- x = blockIdx.x * blockDim.x + threadIdx.x;
- y = blockIdx.y * blockDim.y + threadIdx.y;
- x a y jsou přesné pozice v mřížce (2D)

Pár funkcí:

- cudaDeviceReset() - funkci (re)inicializovat zařízení.
- cudaDeviceSynchronize() - synchronizovat zařízení a hosta
- cudaGetLastError() - vrátí cudaSuccess nebo error code
- CudaGetErrorsString(...) - vrátí řetězec pro error code
- CudaMalloc(...) - přiděluje paměť v zařízení
- CudaMallocManaged(...) - alokuje unified memory.
- CudaFree(...) - uvolní přidělenou paměť
- CudaMemcpy(...) - kopíruje paměť mezi hostitelem a zařízením. Směr kopírování je dán hodnotou **cudaMemcpyHostToDevice** nebo **cudaMemcpyDeviceToHost**.

GPU a CPU
The Device je grafická karta s pamětí DRAM a GPU multiprocesory.
The Host je jakýkoli počítač s nainstalovaným zařízením. Paměť v zařízení a paměť v hostiteli jsou propojeny pouze sběrnicí. Nejsou sdíleny!

Výpočetní proces (During the GPU computing the CPU can continue its own processes.)
1. Zkopírujte data z paměti HOST do paměti DEVICE
2. Spusťte vlákna v DEVICE
3. Spusťte vlákna v multiprocesorech GPU
4. Zkopírujte výsledky zpět z paměti DEVICE do paměti HOST

C/C++

- The kernel je funkce pro vlákna GPU. Jazyk C/C++ je rozšířen o provádění funkcí jádra příkazem “name<<<...>>>(...)”
- __device__ je modifikátor funkce. Tato funkce se provede v DEVICE a lze ji volat pouze ze DEVICE.
- __host__ je opak __device__ Funkce označené tímto modifikátorem jsou pouze pro CPU.
- __global__ je modifikátor pro jádra. Funkce bude provedena v GPU, ale volaná (spuštěná) je z CPU

Kvalifikátor typu proměnné:

- __device__ deklaruje proměnnou, která se nachází v DEVICE, dokud je aplikace spuštěna.
- __constant__ se používá pro umístění proměnných v konstantní paměti.
- __shared__ proměnná je umístěna ve sdílené paměti bloku, kde běží jádro.