1. Architektura počítačů

Jaké jsou základní principy fungování počítače?

- Počítač je programován obsahem paměti
- Instrukce se vykonávají sekvenčně
- Každý následující krok závisí na tom předchozím

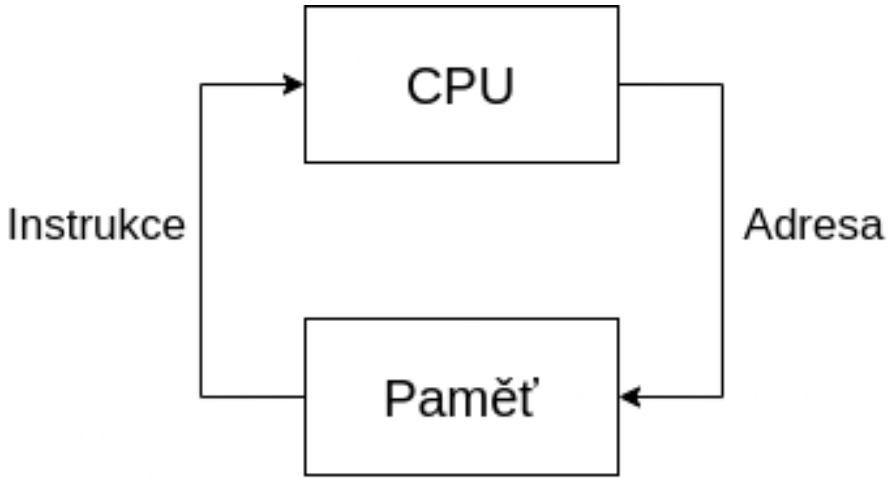


Figure 1: princip_pocitace

- Procesor si přes sběrnici vyžádá instrukci z paměti na adrese IP
- Poté co instrukci získá ji provede
- Zvýší IP/PC
- Cyklus čtení a provedení se opakuje

Kritéria a Principy dle von Neumanna:

- Počítač je řízen obsahem paměti (struktura počítače je nezávislá na typu úlohy)
- Strojové instrukce a Data jsou v jedné paměti (lze přistupovat jednotným způsobem)
- Paměť je rozdělena do buněk stejné velikosti (Jejich pořadové číslo je jejich adresa)
- Následující krok je závislý na tom přechozím
- Program je sekvence instrukcí, ty jsou vykonávány sekvenčně, v pořadí v jakém jsou zapsány do paměti
- Změna pořadí instrukcí je možná pomocí skoku
- Pro reprezentaci čísel, adres, znaků.. se používá dvojková soustava

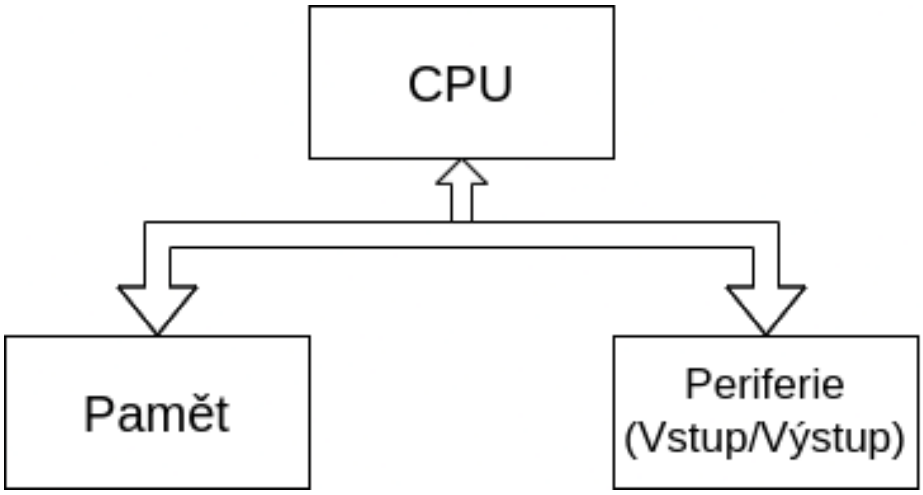


Figure 2: von_neumann

Jaké má výhody a nevýhody architektura dle von Neumanna?

- Výhody
  - Rozdělení paměti pro kod a data určuje programátor
  - do paměti se přistupuje stejném způsobem pro data i instrukce
  - jedna sběrnice => jednodušší výroba
- Nevýhody
  - jedna paměť může mít při chybě za následek přepsání vlastního programu
  - jediná sběrnice je úzké místo

Přinesla harvardská architektura nějaká vylepšení proti von Neumannově?

- Oddělení paměti dat a programu
  - Program už nemůže přepsat sám sebe
  - Paměti můžou být vyrobeny různými technologiemi
  - Dvě sběrnice umožňují přistupovat k instrukcím a datům zároveň
- Nevýhody:
  - dvě sběrnice jsou dražší
  - nevyužitou část paměti dat nelze použít pro program.. a naopak

Jaká je podpora paralelismu u obou architektur počítačů?

- Žádná .. instrukce jsou vykonávány sekvenčně, následující krok je závislí na tom předchozím
- Paralelizmy se musí simulovat až na úrovni OS

Je lepší mít oddělené paměti pro data i program? Proč ano a proč ne?

- Ano
  - Program nemůže přepsat sám sebe
- Ne
  - Jedna sběrnice => jednodušší výroba
  - Rozdělení pro kod a data určuje programátor
  - Lze efektivněji využít kapacitu paměti

Může fungovat počítač bez paměti či bez periferií?

- NE.. jak pravil von Neumann .. je potřeba procesoru, paměti a periferii

K čemu se v počítači využívá dvojková soustava?

- Pro reprezentaci čísel, adres, znaků..

Zvyšují sběrnice výkon počítače?

- Ne přímo, ale mohou jej omezit

Je možné, aby procesor prováděl instrukce jinak, než sekvenčně?

- NE instrukce se provádějí sekvenčně

Jak je v počítači organizovaná paměť?

- Je složená z za sebou jdoucích buněk stejné velikosti (obvykle 8bit), jejich pořadové číslo se využívá jako jejich adresa

2. Jazyk symbolických instrukcí

3. Komunikace s periferiemi

Z jakých částí se skládá sběrnice a co je účelem jednotlivých částí?

- Sběrnice dělíme na Adresovou, Řícicí, Datovou
- Adresová
  - Přenáší adresy
  - Zdroj adresy je mikroprocesor
  - Počet bitů (vodičů) sběrnice odpovídá počtu bitů adresy
- Řídicí
  - Některé signály jsou generovány mikroprocesorem, některé jinými bloky
  - Nejčastější řídicí signály:
    - RESET

            - má každý mikroprocesor
            - uvede mikroprocesor do výchozího stavu
        * MEMORY READ (MR)
            - zabezpečuje časování čtení z paměti (nebo jiných bloků)
        * MEMORY WRITE (MW)
            - zabezpečuje časování zápisu do paměti (nebo jiných bloků)
        * INPUT / OUTPUT READ / WRITE
            - pro čtení nebo zápis do zařízení
        * READY
            - připravenost obvodu
- Datová
    - Slouží pro přenos veškerých dat v počítači
    - Nedůležitější parametry jsou šířka (počet bitů) a časování
    - Šířka ovlivňuje rychlost komunikace
    - Lze ušetřit vodiče pomocí multiplexování

Co to je adresní dekodér a kdy je potřeba jej použít?

- Když je paměťový prostor obsazen více jak jednou fyzickou pamětí nebo periferním zařízením
- Rozhoduje, které zařízení je ke komunikaci určeno
- Jeho výstupy jsou v podstatě Chip Select signály pro jednotlivé obvody
- Může být stavěn jako:
    - úplné dekódování adresy
    - neúplné dekódování adresy
    - lineární přiřazení adresy
    - univerzálním přiřazením adresy

Řízení komunikace

- 2 případy zahájení komunikace
    - z iniciativy programu
    - z iniciativy periferie
        * počítač se může nacházet ve stavu, kdy nemůže s periferii komunikovat
        * lze řešit:
            - obvodově (bez vědomí počítače)
            - příznakovým bitem (Programové řízení)
            - přerušením .. počítač se později vrátí tam, kde byl vyrušen (Systém Přerušení)
            - přímým přístupem (DMA)

Jaký je princip komunikace s periferiemi pomocí V/V bran?

- Vstupně / Výstupní brána (Input/Output, I/O) je obvod, kter zprostředkovává předávání dat mezi sběrnicí (počítače) a periferním zařízení (počítače)

- Dělíme na
  - S pamětí
  - Bez paměti
- Základem je záchytný registr s 3 stavovým vstupem

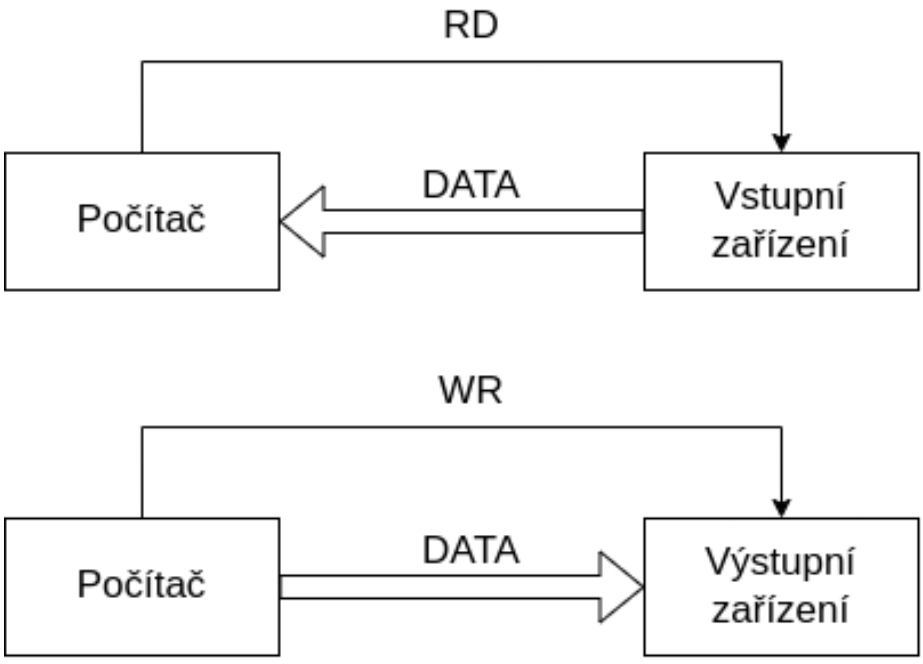


Figure 3: VV_nepodmineny

Nepodmíněný vstup a výstup dat:

- Při vstupu počítač vyšle signál RD, tím přikáže vstupnímu zařízení předat data do vstupní brány počítače
- Při vstupu počítač vyšle signál WR a výstupní zařízení převezme data
- Jednoduchý způsob, předpokládá, že je perif. zařízení pořád ready

K čemu slouží u komunikace V/V bran indikátor a jaké přináší výhody?

- Zajišťuje, že informace budou správně podány (další otázka)

Popište, jak probíhá přenos dat pomocí V/V brány s indikátorem.

Podmíněný vstup a výstup dat

- Jsou-li poskytována platná data ze vstupu, pak se za pomocí STB(strobe) impulsu nastaví Q na 1

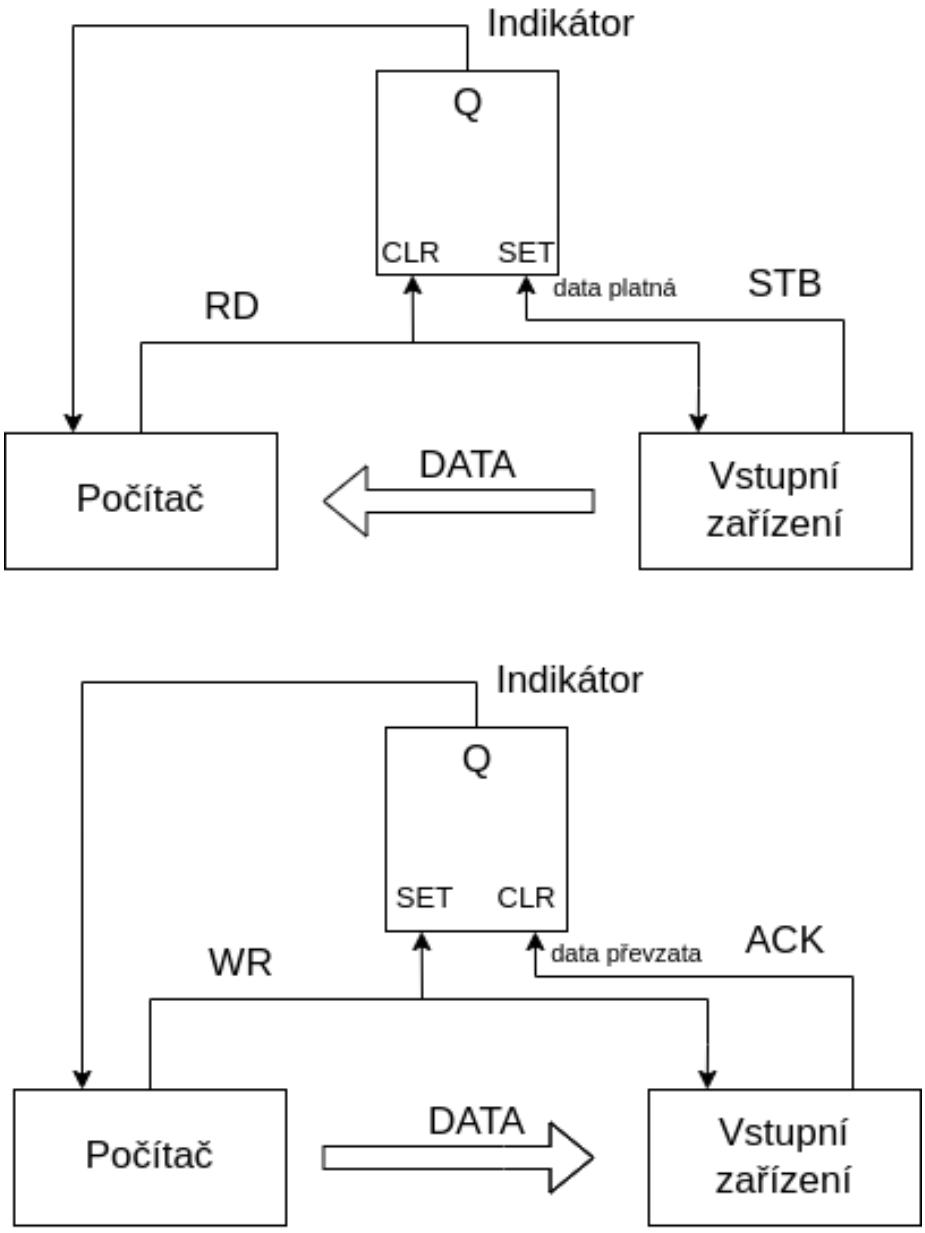


Figure 4: VV_podmineny

- Když je Q na 1, data jsou předány počítači pomocí impulsu RD a po přenosu je indikátor vynulován
- V opačném případě se nastavuje Q na 1, když jsou data převzata, pomocí ACK signálu

Jaký je rozdíl mezi programově řízenou komunikací s perifériemi a pomocí přerušení?

- Programové:
  - Využívá instrukce pro vstup a výstup, ve spojení s instrukcemi pro testování logických proměnných a skoků
  - Prostě testuje stavové bity ..
- Přerušení:
  - Periferie aktivuje přerušovací signál, procesor přeruší program, přejde do obslužného režimu, poté pokračuje v provádění hl. programu tam, kde byl přerušen

Jaké výhody přináší řízení komunikace s využitím přerušení?

- Procesor pořád nemusí zbytečně testovat stavové bity => neztrácí výkon

Z jakých částí se skládá řadič DMA?

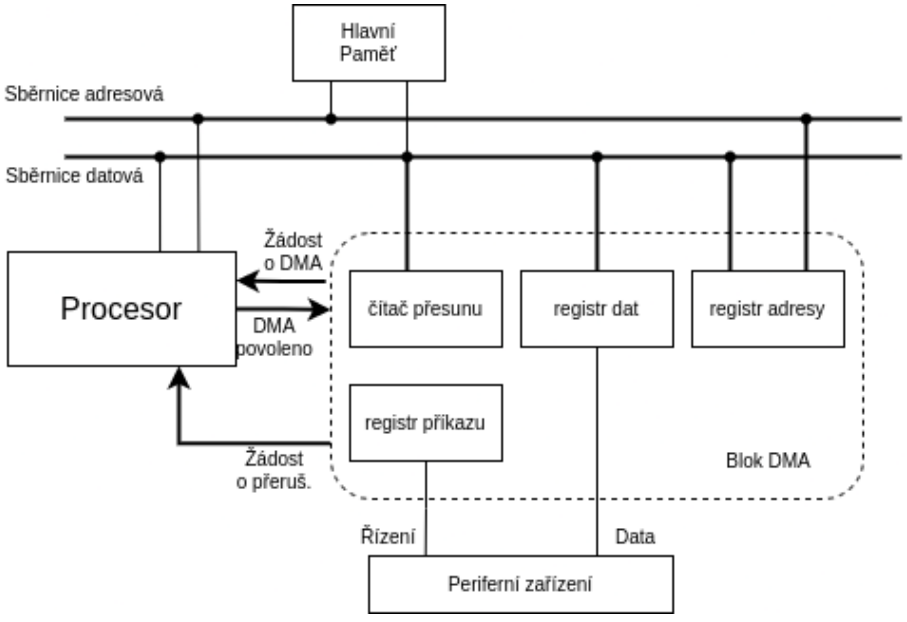


Figure 5: blok_DMA

- Registr dat - Obsahuje slovo pro přesun

- Registr adresy - Adresa hl. paměti kam bude slovo zapsány, nebo odkud bude přečteno
- Čítač přesunu - požadovaný počet slov, které mají být ještě přesunuty

Jak probíhá přenos dat s použitím DMA?

1. Naprogramování procesorem bloku DMA
2. blok DMA spustí periferní zařízení, a čeká než zařízení bude připraveno data příjmou nebo vyslat
3. Procesor dokončí strojový cyklus a pak reaguje na žádost o DMA, přímý přístup se provadí během činosti procesoru.. blok DMA a procesor se střídají v používání paměti
4. Procesor vyšle vybrané jednotce ACK a uvolní sběrnici, jednotka pak pošle obsah registru adresy na addr. sběrnici a obsah registru dat na dat. sběrnici a čeká na provedení cyklu paměti.. pak obsah registru adresy zvětší o jedničku a čítač přesunu zmenší o jedničku.. pokud není nulový, testuje zda bylo předáno nové slovo do registru dat.. když ne, dočasně se ukončí přesun dat a přestane se vysílat žádost o DMA.. řízení je předáno procesoru
5. Procesor dále pokračuje v provádění svého programu do doby, než obrží další žádost o DMA
6. Pokud je obsah čítače přesunu nulový, blok DMA ukončí cel přesun a uvolní sběrnici

Jaké má výhody řadič DMA proti přenosu dat s využitím CPU?

- Všechno nemusí dělat procesor

I2C ???

Příklad jedné sběrnice: I2C, viz cvičení, zapojení, cyklus sběrnice, adresování, přenos dat.

4. CISC A RISC

Kdy a proč se začaly procesory dělit na RISC a CISC?

- V 70. letech, kvůli narůstající složitosti procesorů ..

Jaké byly zásadní důvody, proč se začaly procesory RISC vyvíjet?

- Výzkumy ukázaly, že programátoři a compilátory používají instrukce velmi nerovnoměrně (v 50% případů se vyskytují pouze 3 instrukce)
- Snahy o nalezení optimálního instrukčního souboru => vznik RISC

Jaké jsou základní konstrukční vlastnosti procesorů RISC?

- Malý instrukční soubor

- V každém strojovém cyklu by měla být dokončena jedna instrukce
- Zřeťezené zpracování instrukcí
- Data jsou z hlavní paměti vybírána a úkládána výhradně pomocí LOAD a STORE instrukcí
- Instrukce mají pevnou délku a jednotný formát
- Je použit vyšší počet registrů
- Složitost se přesouvá na optimalizující kompilátory

Jak přispěly jednotlivé charakteristické vlastnosti procesorů RISC ke zvýšení výpočetního výkonu?

- Jednotná délka instrukcí => rychlejší výběr instrukcí z paměti => lepší plnění fronty instrukcí
- Jednotný formát => zjednodušuje dekódování
- Zřetězené zpracování instrukcí

Jaký je princip zřetězeného zpracování instrukcí v RISC procesorech?

- Provedení instrukce musí vždy projít stejnými fázemi (né nutně těma co jsou na obrázku)
- Funguje jako "výrobní linka"

CISC:

	T1	T2	T3	T4	T5	T6	T7	T8	T9	T10	T11	T12
VI	I1						I2					
DE		I1						I2				
VA			I1						I2			
VO				I1						I2		
PI					I1						I2	
UV						I1						I2

RISC:

	T1	T2	T3	T4	T5	T6	T7	T8	T9	T10	T11	T12
VI	I1	I2	I3	I4	I5	I6	I7					
DE		I1	I2	I3	I4	I5	I6	I7				
VA			I1	I2	I3	I4	I5	I6	I7			
VO				I1	I2	I3	I4	I5	I6	I7		
PI					I1	I2	I3	I4	I5	I6	I7	
UV						I1	I2	I3	I4	I5	I6	I7

Legend:

short name	full name
VI	Výběr Instrukce
DE	Dekodování
VA	Výpočet Adresy
VO	Výběr Operandu
PI	Provedení Instrukce
UV	Uložení Výsledku

Jakého zrychlení lze zřetězeným zpracováním instrukcí dosáhnout?

- V ideáním světě, při délce zřetězení 6-ti instrukcí, udělá během 12 cyklů
  - CISC: 2 instrukce
  - RISC: 7 instrukcí
- viz. tabulky v minulé otázce

Jaké problémy přináší zřetězené zpracování instrukcí v procesorech RISC?

- Datové a strukturální hazardy
  - Datové: Když instrukce potřebuje mít k dispozici data předchozí instrukce (a ta ještě nejsou k dispozici)
  - Struktrální: Problém omezených prostředků procesoru (a počítače jako celku) .. např. jen jedna sběrnice
- Problémy plněním fronty instrukcí
  - Podmíněné skoky
  - Nepodmíněné skoky na adresu, která se musí vypočítat

Co to je predikce skoků, proč se používá a jaké způsoby predikce se využívají?

- Statická
  - Do instrukce se vkládají příslušné bity již při kompilaci (nebo programátorem při psaní programu)
- Dynamická
  - Během běhu programu se zaznamenává, jestli se skok provedl, nebo ne
- Může být:
  - Jedno bitová
  - Dvou bitová

Co to jsou datové a strukturální hazardy v RISC procesorech? Co je způsobuje?

- uvedeno výše..

Jak funguje dvoubitová dynamická predikce skoků a proč se využívá?

- Jako čtyř stavový automat

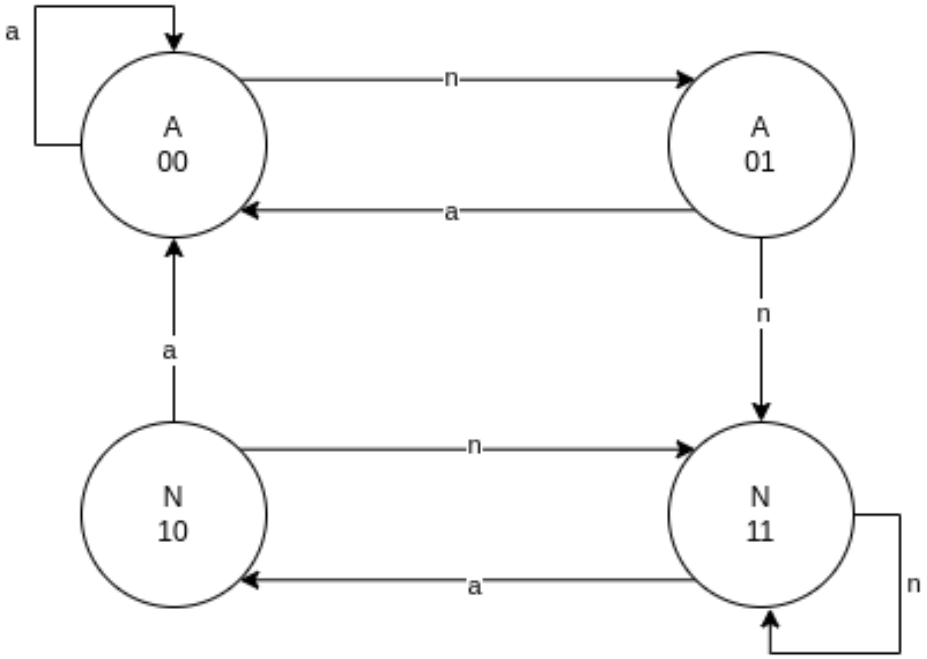


Figure 6: 2bitpredikce

- `A` predikuje provedení skoku, `N` říká, že skok provádět nebude
- `a` a `n` přechody označují, zda se skok naposledy prováděl

5. x86 Intel historie

- 8080
  - Není x86
- 8086
  - Prvním 16-bit
- 8088
  - Sběrnice zúžená na 8bit
  - Jinak stejné jak 8086
- 80186
  - Navržen pro embedded (vestavěná) zařízení
  - Má DMA
  - Vyráběn 25 let
- 80286

  - Lze přepnout do Protected modu (4 úrovně oprávnění)
  - Real mode (pro zpětnou kompatibilitu, RM programy nemůžou fungovat v novém PM)
  - má MMU (memory management unit)
- 80386dx a sx
  - sx je downgrade dx
  - První 32bit
  - Přidán Virtual Mode (po přepnutí do PM, bylo možnost vykonávat RM programy)
- 8087/287/387
  - Matematick koprocesor, pro práci s floaty, který byl zvlášt
- 80486dx (později i sx verze)
  - Dvojnásobný výkon při stejné frekvenci, než 386
  - L1 přímo v procesoru
  - Integrace matematického koprocesoru
- Pentium
  - První procesor v řadě x86, kde jsou uplatněna technická řešení typická pro RISC
  - L1 rozdělena na kod a data
  - Predikce skoků
- Pentium Pro
  - ZÁSADNÍ technologický zlom
  - Pro servery (=> velký výkon(zhruba o 50% víc než pentium) a cena)
  - L2 přímo na procesoru
  - Fetch/Decode jednotka dekoduje x86 instrukce na 118bit RISC instrukce (které intel pojmenoval jako mikro-operace)
  - Instrukce jsou po dekodování uloženy do banky instrukcí (Instruction pool), vejde se tam až 40 instrukcí
  - Dispatch/Execute jednotka si může vybírat instrukce mimo pořadí z poolu (Out of order execute)
  - 10 úrovňové zřetězení
  - Predikce skoků si pamatuje 512 hodnot
- Pentium 2
  - Vychází z Pentia Pro
- Pentium 3
  - Optimalizace z hlediska spotřeby
  - Dobré pro přenosné počítače
- Pentium 4
  - Mikroarchitektura NetBurst
  - Při stejné frekvenci jako P3 měl stejný výkon+-, ale více se zahříval
  - 20 úrovňové zřetězení
- Pentium EM64T
  - Extended Memory 64 Technology (Později jen Intel64)
  - První 64bit procesor
  - 30 úrovňové zřetězení

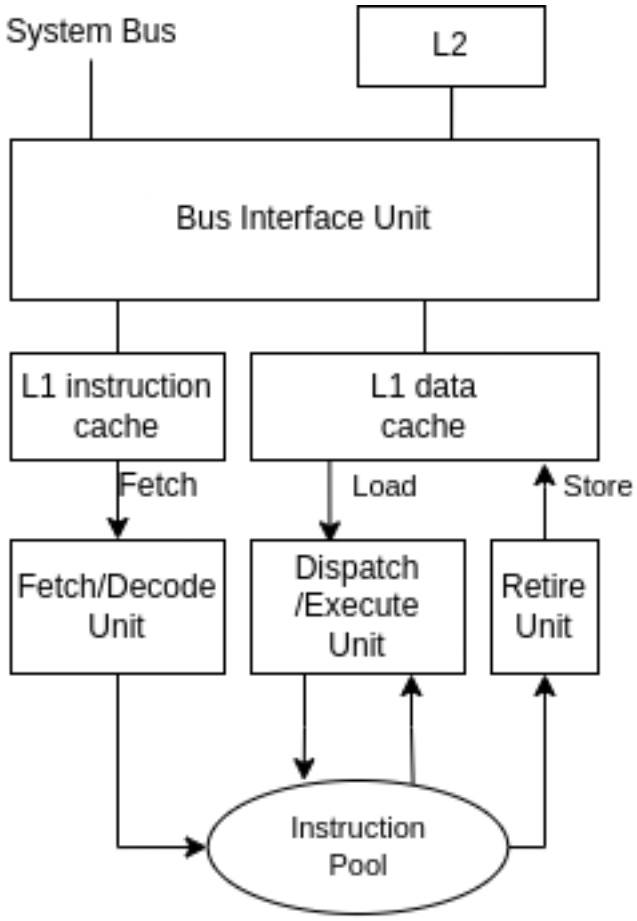


Figure 7: pentium_pro

  - Velice se přehřívaly
- Pentium M
  - Určeny pro přenosné počítače
  - Výkonný procesor s nízkou spotřebou energie
  - Obdobný výkon jako P4 při nižší frekvenci a třetinové spotřebě
- Core
- Core 2
- Atom

6. Paměti

Dle jakých kritérií či vlastností se dělí paměti počítačů?

- Typu přístupu
  - RAM (Random access memory) - libovolný přístup
  - SAM (Serial acess memory) - Seriový přístup
  - Speciální (paměť typu zásobník, fronta..)
- Možnosti zápisu/čtení
  - RWM (Read write memory) - pro zápis a čtení
  - ROM (Read only memory) - pouze pro čtení
  - Kombinované
    - NVRAM (Non volatile RAM)
    - WOM (Write only memory)
    - WORM (Write once - ready many times memory) - optické disky
- Principu elementární buňky
  - SRAM - statické paměti
  - DRAM - dynamické paměti
  - PROM, EPROM, EEPROM, FLASH - programovatelné paměti
- Uchování informace po odpojení napájení
  - Non-Volatile - Zachovají si informaci i po odpojení napájení
  - Volatile - Ztráci informaci po odpojení napájení (DRAM a SRAM)

Jak je v dynamických pamětech ukládána informace a jak je udržována?

- Ve formě náboje v kondenzátoru
- Zapomenou svá data cca po 10ms
- Proto je nutné obnovovat napětí kondenzárorů - Refresh

Jaká je vnitřní organizace dynamických pamětí?

- Ve čtvercové matici v jedné, nebo více vrstvách
- Výběr buňky tak musí být proveden pomocí row a column dekodéru
- Organizace paměti, strana 5

Popište stručně historii vývoje dynamických pamětí.

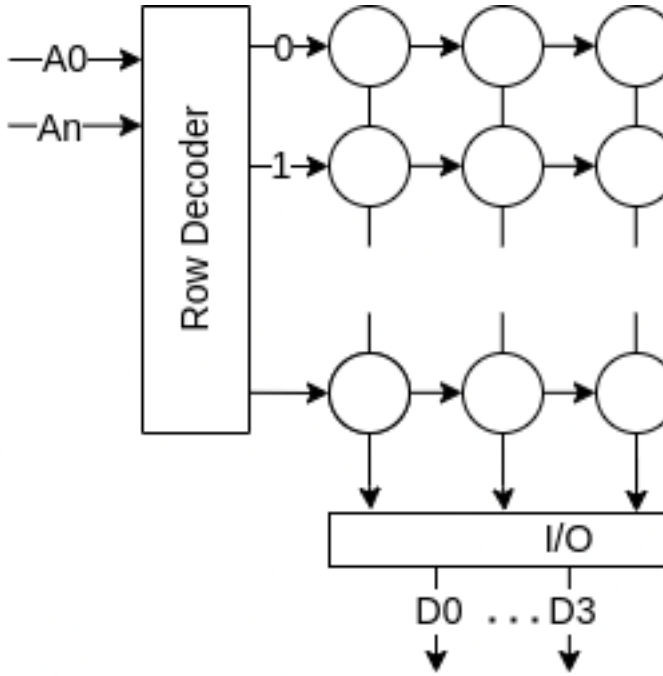
- ?? :c

Jak je ve statických pamětech ukládána informace a jak je udržována?

- Je uložená stavem klopného obvodu
- Lze realitovat pomocí 4 nebo 6 tranzistorů
- SRAM je dražší a pojme méně dat něž DRAM

Jak je organizována vnitřně statická paměť?

- Jako 2D mřížka, kde jeden řádek tvoří jedno slovo



- SRAM paměti nevyužívají adresní multiplexing

Jaké typy pamětí si udržují svůj obsah i po odpojení napájení?

- (Nevolatilní)
- ROM (Read Only Memory)
  - Informace zapisuje výrobce (je složená z odporů, které výrobce přepálí.. neporušené prvky pak vedou proud a je v nich minimální napětí.. log. 0)
  - Doba pamatování není ohraničená
- PROM
  - Programmable ROM
  - Informace se vypalijí pomocí "programátoru"
  - Lze zapsat jen jednou
- EPROM
  - Erasable PROM
  - Uchovává informaci díky kvalitně izolovaném el. napětí

  - K naprogramování je potřeba až 50ms trvající pulz o 5V
  - Lze vymazat pomocí UV záření
  - Doba pamatování 10 až 20 let
- EEPROM
  - Electrically Erasable PROM
  - Zápis stějně jak EPROM
  - Mazální pomocí el. pulzu s obrácenou polaritou
  - Doba pamatování 10 až 20 let
- FLASH
  - Lze programovat rychle přímo v počítači
  - Doba pamatování 10 až 100 let
  - Struktura buněk je podobná EEPROM, ale pro programování a mazání stači pulz 10us
  - Před 10000 programovacích a mazacích cyklů

Paměti s trvalým obsahem umožňují svůj obsah přepsat. Jak se přepis u jednotlivých typů provádí?

- EPROM - UV zářením
- EEPROM - Elektricky, až 50ms pulzem o 5V
- FLASH - Elektricky 10us pulzem

Jaké speciální typy pamětí se používají?

- VRAM (Video RAM)
  - Dvouportová
  - Zvýšené přenosové pásmo
- WRAM (Window RAM (nemá nic společného s tím pseudo operačním systémem))
  - O 25% větší přenosové pásmo než VRAM
  - Nabízí double-buffering
- SGRAM (Synchroní Grafická RAM)
  - Funguje jako SDRAM
  - Ale SDRAM je optimalizována pro kapacitu a SGRAM pro přenos dat
- FIFO paměti (fronta)
  - Bez přesouvání obsahu
  - S přesouváním obsahu
- Cache paměti
  - Malé a rychlé
  - Rychlé komponenty čtou data z cache a nemusí čekat na komponentu pomalejší
  - L1,L2,..

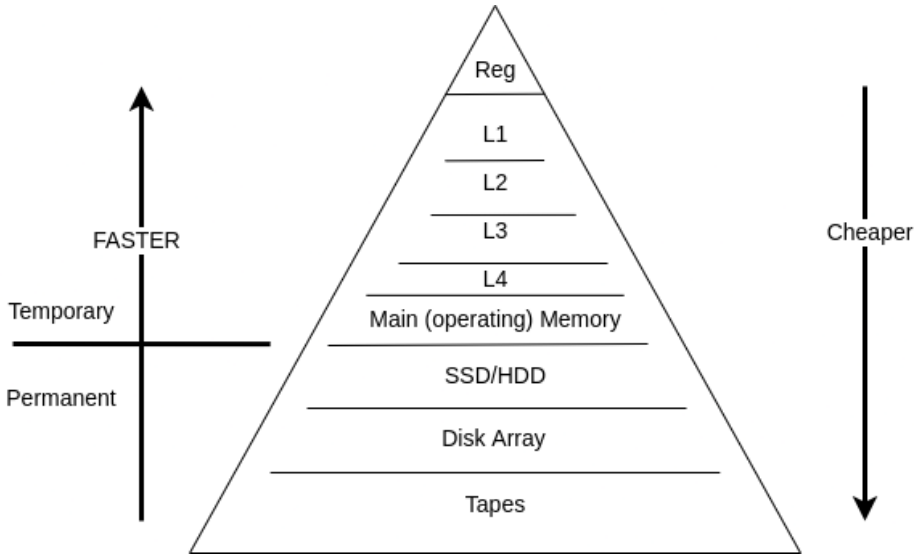


Figure 8: memory_h

Hierarchie pamětí v počítači

Jak se u pamětí detekují a opravují chyby?

- SRAM je spolehlivější než DRAM
- Tvrdé chyby - opakující se
- Měkké chyby - Neopkající se .. těžší rozpoznat
- Kontrola Parity
  - dorovnává se na lichý počet jedniček do 9. bitu
  - neopravuje, jen detekuje chybu (když je počet jedniček sudý)
- ECC - Error Correction Code
  - Detekuje více bitové chyby
  - Schopen opravit 1 bitovou chybu
  - Nutnost “Wait State” => zpomalení 2-3%

7. Monolitické počítače

Jaká je obvyklá organizace pamětí v mikropočítačích?

Jaké zdroje hodinového signálu se mikropočítačích používají?

Jak probíhá RESET mikropočítače?

Jakými způsoby se řeší ochrana proti rušení v mikropočítačích?

Jaké jsou základní vlastnosti V/V bran?

Popište obecný princip fungování sériových rozhraní? Jaká sériová rozhraní znáte?

K čemu slouží v mikropočítačích čítače a časovače? Jak fungují?

Popište konstrukci a fungování základních A/D převodníků.

Popište konstrukci a fungování základních D/A převodníků.

Jaké speciální periferie mikropočítačů znáte?

8. Monitory

Co to znamená u monitoru „šířka pásma“ a o čem vypovídá?

Na jakých principech fungují CRT monitory?

Na jakých principech fungují LCD monitory?

Jaké jsou základní výhody a nevýhody LCD monitorů?

Jak fungují OLED zobrazovací jednotky?

Jaké jsou výhody a nevýhody OLED technologie?

Jak funguje zobrazovací jednotka s technologií E-Ink?

Jaké jsou výhody a nevýhody E-Ink?

Jak je u E-Ink řešena podpora více barevných úrovní?

Co je princip multiplexu na zobrazovacích zařízeních? ???

9. Disky

10. CUDA

11. Mikropočítač a RISC CPU