Otázka 09 - Y36SAP

Zadání: **Základní principy organizace počítačů. Struktura počítače, funkce hlavních jednotek. Instrukce a jejich struktura; způsoby adresace základní operace. Přerušení. Sběrnice. Paměti, jejich typy a organizace. Skrytá paměť (cache) a virtuální paměť. V/V jednotky a jejich řízení; DMA. Řadič a jeho organizace. Proudové zpracování informace (pipelining). Procesory typu CISC a RISC.**

Informace od statnic 2009 : Jeden člověk dostal otázku JEN na přerušení: Co to je přerušení, kde se detekuje, jaké jsou typy přerušení, co dělá procesor když dostane víc přerušení, indikace přerušení atd.. Prostě můžete dostat jen podotázku toho celého, ale víceméně to pak hodnotí mnohem mírněji, tak se nestresujte :) Napsal jsem jim o přerušení 5 řádek, trochu to okecal a dostal sem za B z ústní + A z BP a celkově A. :)

Slovníček pojmů

Pojem	Význam	Bližší popis
ALU	Arithmetic logic unit [http://en.wikipedia.org/wiki/Arithmetic_logic_unit]	obvod (typicky kombinační), který se stará o aritmetické a logické operace s daty
BIOS	Basic Input/Output System [http://en.wikipedia.org/wiki/BIOS]	stará se o zavedení operačního systému při startu počítače
Bus	Sběrnice [http://en.wikipedia.org/wiki/Bus_(computing)]	stará se o přenos dat mezi jednotlivými součástmi počítače; může být paralelní nebo sériová
Bus mastering	Bus mastering [http://en.wikipedia.org/wiki/Bus_mastering]	schopnost sběrnice umožňující zařízení na ní napojenému iniciovat transfer dat, používá se u DMA
Cache	Vyrovnávací paměť [http://cs.wikipedia.org/wiki/Cache]	rychlá a drahá paměť, která si pamatuje data, jež jsou v atuálních procesech často potřeba, aby se pro ně nemuselo sahat do pomalé operační paměti
CAM	Asociativní paměť [http://en.wikipedia.org/wiki/Content-addressable_memory]	speciální typ paměti, do které se přistupuje pomocí klíčů, používá se např. pro cache
CPU	Central processing unit [http://en.wikipedia.org/wiki/Cpu]	výpočetní jednotka, alias procesor
DMA	Direct memory access [http://en.wikipedia.org/wiki/Direct_memory_access]	umožňuje některým zařízením (diskům, síťovým kartám apod) přistupovat přímo do operační paměti, nezávisle na procesoru
IRQ	Interrupt request [http://en.wikipedia.org/wiki/IRQ]	přerušení, umožňuje jednotlivým zařízením signalizovat procesoru, že potřebují obsluhu

Základní principy organizace počítačů

Počítač

Matematicky vzato je počítač zobrazení, které nějakým vstupním datům přiřazuje data výstupní.

Zobrazení dat

- **Nespojité – diskrétní (číslicové, digitální)**

- **Spojité - analogové**

V základní rovině se dělí na:

- **počítač *analogový* - který data zobrazuje spojitě**
- **počítač *digitální* neboli *číslicový*, který data zobrazuje nespojitě**
- ***hybridní*, který oba přístupy kombinuje pomocí A/D a D/A převodníků**

Mooreův zákon

Hovoří o dramatickém rozvoji výpočetních technologií, když tvrdí, že se každých 18 měsíců zdvojnásobí hustota integrace. Reálně to znamená např.:

- **logická kapacita procesoru o 30% za rok; hodinová frekvence 20% ročně**
- **kapacita disku se za rok zvýší o 60%**
- **kapacita hlavní paměti o 60% za rok ;cena za bit operační paměti se sníží ročně o 25%**
- **šířka pásma počítačových sítí se zvýší ročně o 100%**

Z čeho se skládá výpočetní jádro počítače na různých úrovních abstrakce

Úroveň abstrakce

Funkční, strukturní i fyzikální reprezentace může být použita na různé úrovni abstrakce (granularity), podle použitých typů objektu.

Úroveň abstrakce	Funkční popis	Strukturní bloky	Fyzikální objekty
transistor	Diferenciální rovnice, volt-ampérová charakteristika	Transistor, odpor, kondenzátor	Analogové a číslicové buňky - layout
hradlo	Boolovské rovnice, konečný automat	Hradlo, klopný obvod	Moduly, bloky
registr	Algoritmus, vývojový diagram, soubor instrukcí	Sčítačka, komparátor, čítač, registr	Mikročipy
procesor	Specifikace funkce, program	Procesor, řadič, paměť	Desky plošných spojů, vícečipové moduly

Součásti počítače

Software

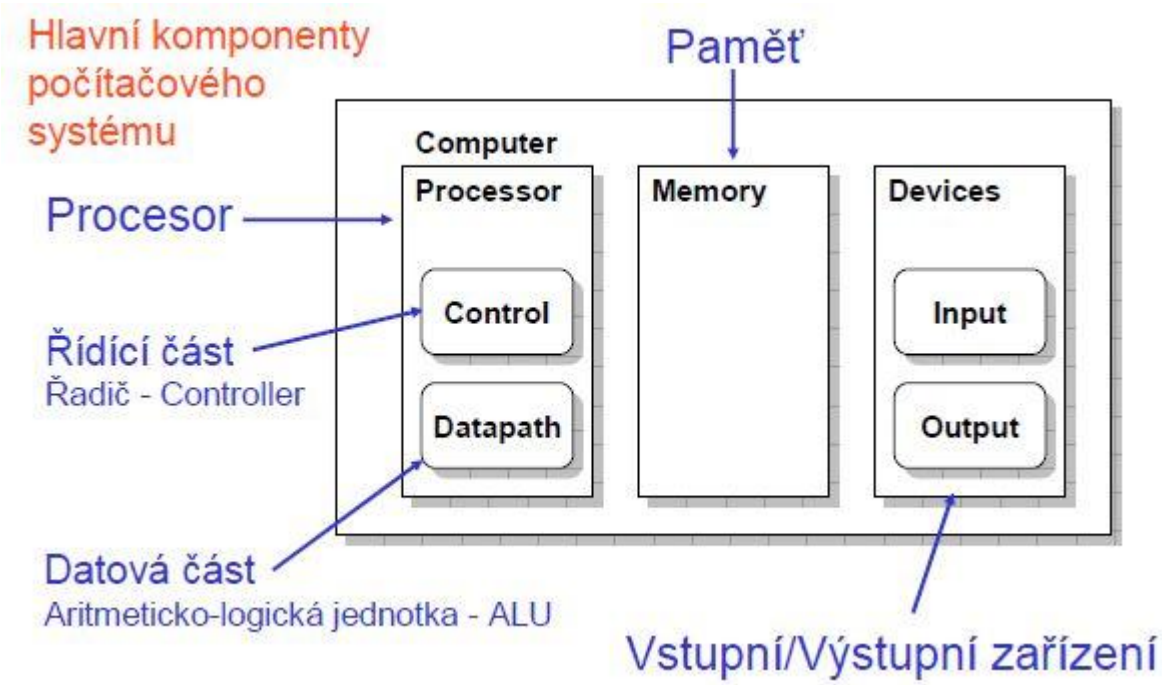
- **firmware - jednoduché programy, které řídí elektronické součástky (klávesnici, lcd obrazovku, atd.),BIOS, adresní módy, ISA, assembler**
- **operační systém**
  - **jakýsi most mezi uživatelskými aplikacemi a hardwarem počítače.**
  - **Struktura souborů na disku, privilegia, ochrana, přepínání úloh, správa paměti a zařízení.**
- **vývojářský software - překladače, linkery (propojují knihovny do jednoho spustitelného souboru), debugger atp.**
- **aplikace - programovací jazyky, editory, hry, prohlížeče atd.**

Hardware

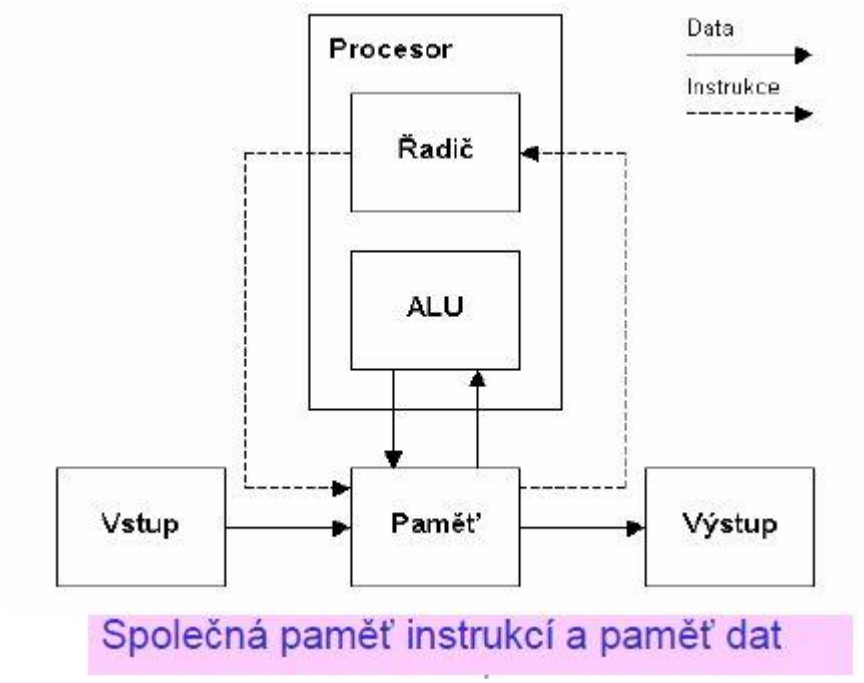
- **architektura počítače - jak je navržen procesor (architektura), jak probíhá tok dat, řízení, predikce větvení.**
- **paměťová hierarchie - systém vyrovnávacích pamětí (cache), správa paměťového systému, segmentace, stránkování**
- **uživatelské rozhraní - nebo také *periferie* - myš, klávesnice, monitor, porty počítače, atd.**
- **další rozhraní - DMA (Direct Memory Access), systém přerušení, komunikační protokoly.**

Struktura počítače

Hardwarová architektura počítače



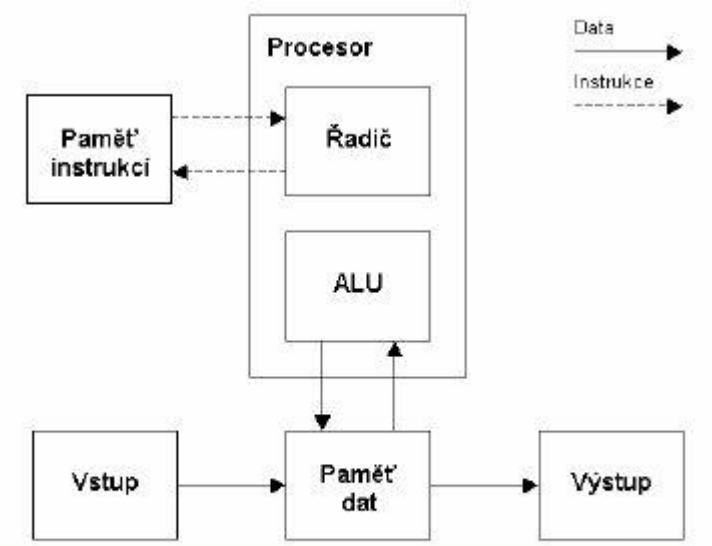
Von Neumannova architektura



Společná paměť instrukcí a paměť dat

- **Instrukce a data jsou uloženy v téže paměti,**
- **paměť je organizována lineárně (tzn. jednorozměrně) a je rozdělena na stejně velké buňky, které se adresují celými čísly (zpravidla 0, 1, 2, 3, ...),**
- **data ani instrukce nejsou explicitně označeny,**
- **explicitně nejsou označeny ani různé datové typy,**
- **pro reprezentaci dat i instrukcí se používají dvojkové signály,**
- **v instrukci zpravidla není uváděna hodnota operandu, ale jeho adresa,**
- **instrukce se provádějí jednotlivě, a to v pořadí, v němž jsou zapsány v paměti, pokud není toto pořadí změněno speciálními instrukcemi (nazývanými skoky).**
- **Procesor může najednou pouze číst nebo zapisovat data nebo instrukce = sekvenční zpracování = nižší rychlost**
- **Důsledek – podle výpisu paměti nelze poznat, zda jde o instrukce, nebo o data. Je třeba znát kontext**

Harvardská architektura



Oddělena paměť instrukcí a paměť dat

- **data a instrukce programů jsou uloženy v oddělené paměti**
- **Paměti můžou být naprosto odlišného typu**
- **Paralelní zpracování dat – lze zároveň číst/zapisovat z/do paměti programu a paměti dat najednou**

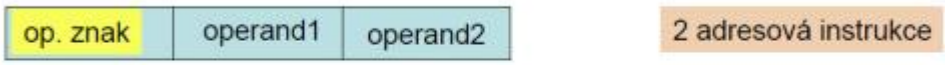
Instrukce procesoru

***Instrukce* je příkaz procesoru zakódovaný jako číslo. Skládá se typicky z názvu (*operačního znaku*) a seznamu operandů. Jako taková však musí obsahovat následující informace:**

- **co se má provést**
- **s čím se to má provést (operandy)**
- **kam se má uložit výsledek (do jakého registru)**
- **kde se má pokračovat (typicky na dalším řádku, nebo se provede skok, či volání podprogramu)**

Přičemž tyto informace mohou být v instrukci obsaženy implicitně (např. vyplynou z architektury počítače) nebo explicitně tím, že je v instrukci uvedeme.

Struktura instrukce



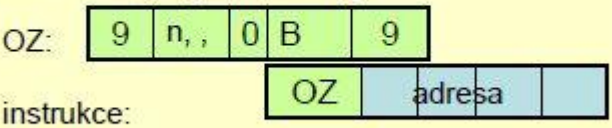
výsledek se ukládá na místo prvního operandu, zavedení operace **přesun**

$$\langle x\rangle - \langle y\rangle \to z \quad \equiv \quad \begin{cases} \langle x\rangle \to z \\ \langle z\rangle - \langle y\rangle \to z \end{cases}$$

zavedení „pracovního" registru – STŘADAČ, ACCUMULATOR

$$\langle x\rangle - \langle y\rangle \to z \quad \equiv \quad \begin{cases} \langle x\rangle \to S \\ \langle S\rangle - \langle y\rangle \to S \\ \langle S\rangle \to z \end{cases}$$

Motorola 68000
Datové registry $D_0, D_1, \ldots, D_7$ á 32b
Odčítání 32b: D_n – paměť → D_n



<D3> - <18FF20 ÷ 18FF23> → D3 **96B90018FF20**

Na obrázku vidíme příklad struktury instrukcí v procesoru *Motorola 68000*. Instrukce se skládá z operačního znaku, který identifikuje instrukci jejím číslem a z operandů, které mohou být konstantním číslem nebo adresou do datového registru (odkazem).

Zapsané instrukce pak tvoří tzv. strojový kód. Programovat přímo ve strojovém kódu je velice nepřehledné a neefektivní, proto vznikly

- **vyšší programovací jazyky (které známe dobře - *C*, *Pascal*, *Java*)**
- **jazyky symbolických instrukcí (tzv. assembler), který strojový kód obléká do srozumitelnější podoby**

Architektura Sady Instrukcí (ISA - Instruction Set Architecture)

Zahrnuje

- **Typy a formáty instrukcí, instrukční soubor**
- **Datové typy, kódování a reprezentace, způsob uložení dat v paměti**
- **Módy adresování paměti a přístup do paměti dat a instrukcí**
- **Mimořádné stavy**

Je definována:

- **umístění operandů a výsledku (obvykle definuje nějaké pracovní registry)**
- **typy dat a velikosti operandů**
- **podporované operace (sčítání, násobení, přesuny dat, bitové operace, atd.)**

- **výběr další instrukce (skoky, větvení programu, volání podprogramů)**

Umožňuje:

- **abstrakci - stejný kód může běžet na různých implementacích stejné architektury procesoru**
- **definici rozhraní mezi nízkoúrovňovým SW (operačním systémem) a HW**
- **standardizaci instrukcí**

Základní třídy ISA

Akumulátorově (střadačově) orientovaná ISA (1 registr=střadač):

1 operand	ADD A	acc ← acc + mem[A]
	ADD (A + IX)	acc ← acc + mem[A + IX] IX je **indexovací registr**

Zásobníkově orientovaná ISA

0 operandů	ADD	stack(top-1) ← stack(top) + stack(top-1) top--

ISA orientovaná na registry pro všeobecné použití
(GPR = General Purpose Registers):

2 operandy	ADD A B	EA(A) ← EA(A) + EA(B)
3 operandy	ADD A B C	EA(A) ← EA(B) + EA(C)

EA … **Efektivní adresa** (určuje registr, nebo operand v paměti)

Akumulátorově orientovaná ISA

Výhody

- **jednoduchý HW**
- **minimální vnitřní stav procesoru**
- **rychlé přepínání kontextu**
- **krátké instrukce**
- **jednoduché dekódování instrukcí**

Nevýhody

- **častá komunikace s pamětí**
- **omezený paralelismus mezi instrukcemi**

Zásobníkově orientovaná ISA

Výhody

- **jednoduchá a efektivní adresace operandů**
- **krátké instrukce**
- **vysoká hustota kódu (krátké programy)**
- **jednoduché dekódování instrukcí**
- **neoptimalizující překladač se dá snadno napsat**

Nevýhody

- **nelze náhodně přistupovat k lokálním datům**
- **zásobník je sekvenční**
- **přístupy do paměti je těžké minimalizovat**

GPR ISA

Dnes nejpoužívanější. Po roce 1975 používají všechny nové procesory nějakou podobu GPR
Výhody

- **Registry jsou rychlejší než paměť(včetně cache)**
- **K registrům lze přistupovat náhodně. Zásobník je přisně sekvenční.**
- **Méně častý přístup do paměti - potenciální zrychlení.**
- **jednoduché dekódování instrukcí**
- **neoptimalizující překladač se dá snadno napsat**

Nevýhody

- **omezený počet registrů**
- **složitější překladač**
- **přepnutí kontextu trvá déle**
- **registry nemohou obsahovat složitější datové struktury**
- **k objektům v registrech nelze přistupovat přes ukazatele**

Instrukční cyklus

V jakých krocích probíhá zpracování instrukcí:

Co musí být definováno

Formát a kódování instrukcí

- **Jak se instrukce dekóduje?**

Umístění operandů a výsledku

- **Kolik operandů je v instrukci pro ALU?**

Jak jsou operandy umístěny v paměti nebo jinde?

- **Který operand může být v paměti?**

Typy dat a velikosti operandů Operace v ISA

- **Které jsou podporovány?**

Výběr další instrukce

- **Skoky, větvení, volání podprogramu**

Přerušení

Přerušení (anglicky interrupt) je v informatice metoda pro asynchronní obsluhu událostí, kdy procesor přeruší vykonávání sledu instrukcí, vykoná obsluhu přerušení a pak pokračuje v předchozí činnosti. Původně přerušení sloužilo k obsluze hardwarových zařízení, které tak signalizovaly potřebu obsloužit (tj. odebrat z vyrovnávací paměti data nebo naopak do ní další data nakopírovat, odtud označení vnější přerušení). Později byla přidána vnitřní přerušení, která vyvolává sám procesor, který tak oznamuje chyby vzniklé při provádění strojových instrukcí a synchronní softwarová přerušení vyvolávaná speciální strojovou instrukcí, která se obvykle používají pro vyvolání služeb operačního systému.

Přijde-li do procesoru signalizace přerušení, je v případě, že obsluha přerušení je povolena, nejprve dokončena právě rozpracovaná strojová instrukce. Pak je na zásobník uložena adresa následující strojové instrukce, která by měla být zpracována, kdyby k přerušení nedošlo. Pak je podle tabulky přerušení vyvolána obsluha přerušení, která obslouží událost, kterou přerušení vyvolalo. Obsluha přerušení je zodpovědná za to, aby na jeho konci byl uveden stav procesoru do stavu jako na jejím začátku, aby výpočet přerušené úlohy nebyl ovlivněn, což se z důvodu vyšší rychlosti obvykle dělá softwarově (některé procesory umožňují uložit svůj stav pomocí speciální strojové instrukce). Na konci obsluhy přerušení je umístěna instrukce návratu (RET, někdy speciální IRET), která vyzvedne ze zásobníku návratovou adresu a tak způsobí, že z této adresy bude vyzvednuta následující strojová instrukce. Přerušená úloha tak až na zpoždění nepozná, že proběhla obsluha přerušení. Tabulka přerušení umožňuje, aby procesor mohl rozlišit více různých přerušení (rozlišených čísly), ke každému vyvolat odpovídající obsluhu přerušení (podprogram) a aby šlo jednotlivé obsluhy umístit na libovolná místa v paměti. Obsluha přerušení je obvykle uložena v ovladači, který spolu s novým hardwarovým zařízením do operačního systému instalujeme.

Průběh hardwarového přerušení

1. **Vnější zařízení vyvolá požadavek o přerušení**
2. **I/O rozhraní vyšle signál IRQ na řadič přerušení (na port IRQ 2)**
3. **Řadič přerušení vygeneruje signál INTR – „někdo“ žádá o přerušení a vyšle ho k procesoru.**
4. **Procesor se na základě maskování rozhodne obsloužit přerušení a signálem INTA se zeptá, jaké zařízení žádá o přerušení.**
5. **Řadič přerušení identifikuje zařízení, které žádá o přerušení a odešle číslo typu přerušení k procesoru**
6. **Procesor uloží stavové informace o právě zpracovávaném programu do zásobníku.**
7. **Podle čísla typu příchozího přerušení nalezne ve vektoru přerušení adresu příslušného obslužného podprogramu.**

8. **Vyhledá obslužný podprogram obsluhy přerušení v paměti a vykoná ho.**
9. **Po provedení obslužného programu opět obnoví uložené stavové informace ze zásobníku a přerušený program pokračuje dál.**

Průběh softwarového přerušení

1. **Při příchodu přerušení se uloží stavové informace o právě zpracovávaném programu do zásobníku.**
2. **Zakáže se další přerušení.**
3. **Procesor zjistí vektor přerušení (podle operandu)**
4. **Nalezne obslužný podprogram a vykoná ho.**
5. **Po návratu z podprogramu obnoví uložené stavové informace o přerušeném programu.**

Vlastnosti jednotlivých přerušení

- **hardwarové - dějí se samy (zmáčknutá klávesa, dochází baterie, dělení nulou)**
- **softwarové - pomocí instrukcí („int číslo“) a tím se vyvolá - např. breakpoint**
- **vnitřní - nastanou při zpracování instrukce (dělení nulou, sahání do paměti) a musí se řešit okamžitě**
- **vnější - jdou mimo procesor a chvíli počkají (časovač, docházející baterka)**
- **Tabulka přerušení - na začátku paměti je tabulka s adresami funkcí, které se mají zavolat v případě přerušení. Typicky třeba 16 přerušení a každé má jeden záznam v tabulce. Tento záznam může být i prázdný a v takovém případě se nic neprovede.**

Typy instrukcí

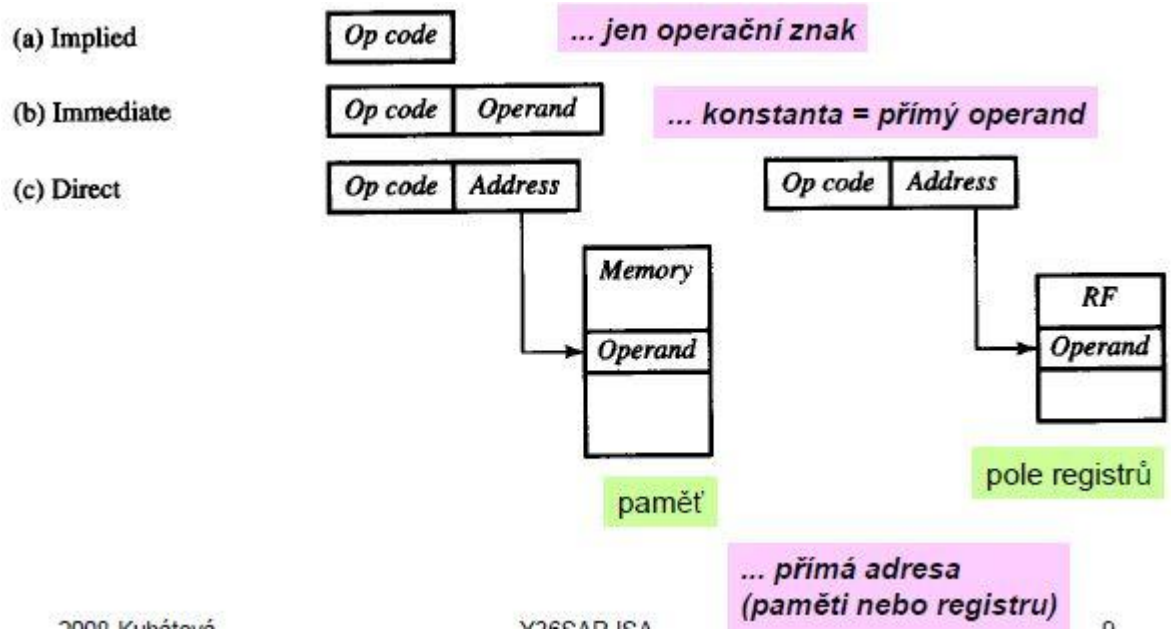
- **aritmetické (sčítání *ADD*,*SUB*, bitové operace *NOT*, *AND*, *XOR*, *OR*, ...)**
- **řídící (skok *JMP*, podmíněné skoky *JE - jump if equals*, *JNE - jump if not equals*, ...)**
- **operace s pamětí (uložení do paměti *ST*, přesuny *MOV*, ...)**

Způsoby adresace operandů

Jakým způsobem se procesor dostane k pravé hodnotě operandu.

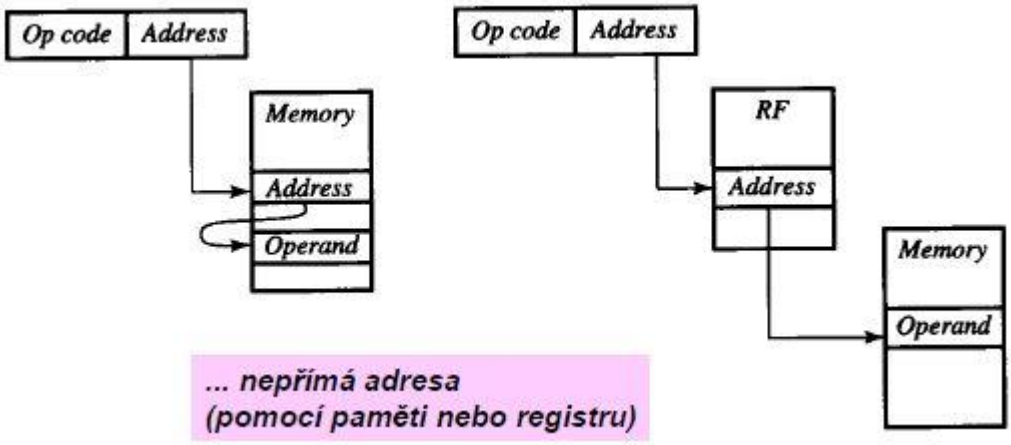
Přímá adresace

- **hodnota je zahrnuta (implied) v samotném operačním znaku (tedy instrukce bez operandu - např. návrat z podprogramu)**
- **hodnota může být okamžitě (immediate) přečtena - operand obsahuje konstantu (např. návrat z podprogramu s návratovou hodnotou)**
- **hodnota může být přímo (direct) přečtena z paměti na základě adresy uložené v operandu.**



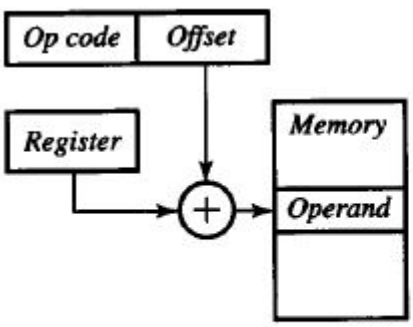
Nepřímá adresace

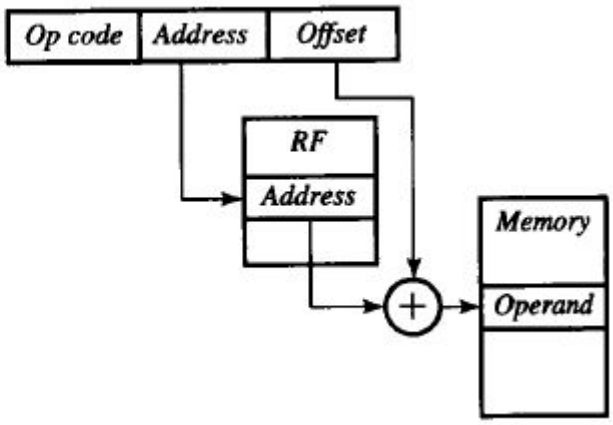
Obdoba přímé adresace, ovšem v paměti lze narazit na další „odskok“ - adresu do jiné části nebo jiného typu paměti.



Relativní adresace

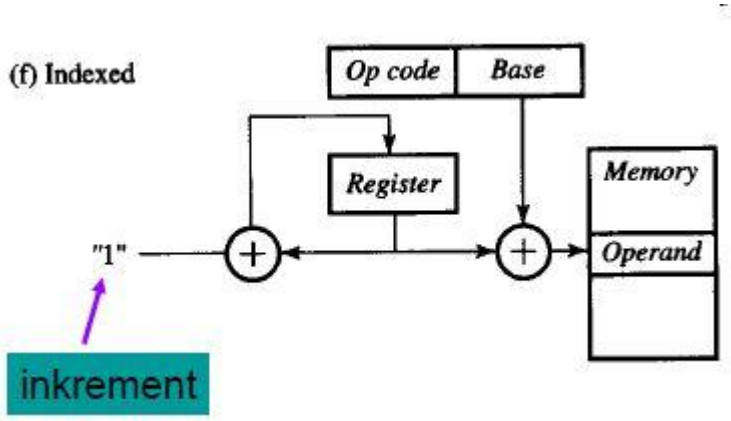
K pevné adrese paměťového bloku připočítá ještě zadaný offset. Lze tak dosáhnout datové struktury typu pole, kdy můžeme automaticky pracovat s celou řadou hodnot.



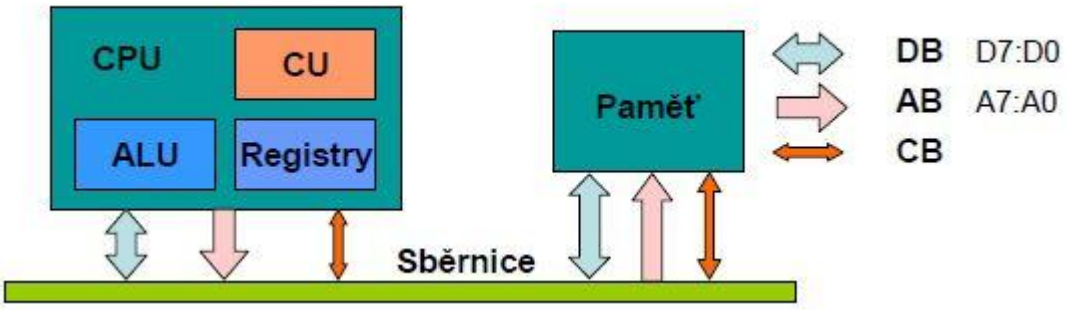


Indexovaná adresace

Obdoba relativní, ovšem offset je nahrazen vnitřním registrem, který se sám postupně načítá. Taktéž vhodné pro zpracování datových struktur založených na poli.



Princip práce procesoru (Von Neumannova typu)



***Centrální výpočetní jednotka (CPU)* se skládá ze 3 hlavních částí:**

- **ALU - *aritmeticko-logická jednotka* se stará o matematické výpočty a obsahuje kombinační obvody**
- **CU - neboli *řadič*, který řídí zpracování instrukcí (sekvenční obvody)**
- **registry - pro dočasné uchování dat**

Registry

Paměťové bloky pro ukládání pracovních dat a provádění výpočtů. U každého procesoru nás zajímá:

- **kolika-bitový procesor**
- **kolik má registrů, jak velkých a jak přístupných**
- **kolika bity se registry adresují (např 16 bitů adresuje 64 tisíc slov - slova můžou mít ale různou velikost, obvykle 2B)**
- **mohou mít příznaky, jež je možné speciálními instrukcemi testovat: *zero* (nula), *sign* (znaménko minus), *overflow* (přetečení), *carry* (přenos na vyšší byte)**
- **reprezentace hodnot (*přímý kód*, *doplňkový kód*, a další)**
- **syntaxe jazyka symbolických adres**

Sběrnice

Subsystém, který se stará o přenos informací mezi jednotlivými částmi počítače. Často bývá sdílena více jednotkami najednou a může být buď *paralelní* nebo *sériová*. V historii výpočetní techniky se střídají období, kdy je v kurzu jedna nebo druhá, dnes je na špici přístup sériový (USB - universal serial bus, SATA, PCI Express, atd.)

Paměti

Základní Pojmy

- **paměťová buňka - základní stavební blok paměti, slouží k záznamu 1 bitu**
- **paměťové místo - skupina paměťových buněk, které lze najednou zapisovat nebo číst**
- **položka - obsah paměťového místa**
- **adresa - číselné označení (index) paměťového místa, jímž lze vybírat jednotlivé položky**
- **kapacita paměti - počet položek**
- **paměťová matice - skupina paměťových míst uspořádaná tak, že je lze vybírat adresou**

Typy pamětí

Podle způsobu a možnosti změny informace

Volatilní

energeticky závislé, uložená informace zanikne po odpojení napájení

- **RAM (Random Access Memory) - paměť umožňující libovolný přístup (na jakoukoliv adresu) pro zápis i čtení**
  - **SRAM Paměti SRAM uchovávají informaci v sobě uloženou po celou dobu, kdy jsou připojeny ke zdroji elektrického napájení. Paměťová buňka SRAM je realizována jako bistabilní klopný obvod, tj. obvod, který se může nacházet vždy v jednom ze dvou stavů, které určují, zda v paměti je uložena 1 nebo 0.**
  - **DRAM V paměti DRAM je informace uložena pomocí elektrického náboje na kondenzátoru. Tento náboj má však tendenci se vybíjet i v době, kdy je paměť**

připojena ke zdroji elektrického napájení. Aby nedošlo k tomutu vybití a tím i ke ztrátě uložené informace, je nutné periodicky provádět tzv. refersh, tj. oživování paměťové buňky.

SRAM vs. DRAM SRAM, neboli Static Random Access Memory je tzv. statická paměť. Od DRAM se liší schopností udržet data i po odpojení od zdroje, neboť není potřeba provádět obnovu „refresh" dat po určitých časových úsecích. SRAM paměti jsou velice rychlé, ale také drahé. Problém je, že „storage cells" (jednotlivé buňky sloužící k uložení dat) jsou složeny z minimálně 4 (spíš více) tranzistorů (DRAM: 1 storage cell = 1 tranzistor = nízká cena). Z tohoto důvodu se statické paměti používají jen zřídka; obvykle pro cache.

Nevolatilní

energeticky nezávislé, obsah zůstane zachován i po odpojení napájení

- **ROM (Read Only Memory) - obsah určen při výrobě, dále ho již není možné měnit a paměť tak slouží jen pro čtení**
- **PROM (Programable Read Only Memory) - obsah se dá jednorázově naprogramovat, pak už slouží jen ke čtení**
- **EPROM (Eraseable Programable Read Only Memory) - obsah pouze pro čtení, je však možné pomocí silného UV záření paměť vymazat a naprogramovat znovu**
- **EEPROM (Electrically Eraseable Programable Read Only Memory) - obsah pouze pro čtení, ale obsah této paměti se dá elektricky smazat a naprogramovat znovu, takže není příliš věrna svému jménu**
- **Flash - speciální typ EEPROM, s tím rozdílem že obsah je mazán a znovu nahráván po celých blocích, čtení je možné z jakéhokoliv místa**

Podle způsobu výběru položek

- **s adresovým výběrem - pro získání dat z paměti potřebujeme znát jejich adresu**
- **asociativní (CAM - content addressable memory) - k datům v paměti přistupujeme pomocí speciálního slova tzn. *klíče*, zpátky dostaneme buď adresy na kterých se dané slovo nachází nebo data s ním asociovaná, jde o hardwarovou implementaci asociativního pole (např. hash, map)**
- **zásobník (LIFO - last in first out) - data, která byla přidána jako poslední, jsou vybrána jako první**
- **fronta (FIFO - first in first out) - data, která byla přidána jako první, přijdou jako první na řadu**

Paměťová hierarchie

- **několikaúrovňové uspořádání pamětí různých typů, aby byl dosažen optimální poměr cena/výkon**
- **cena paměti je totiž přímo úměrná kapacitě a nepřímo úměrná době přístupu**

Typ paměti	Typická realizace	Doba přístupu	Kapacita
registry	klopné obvody	jednotky ns	desítky až stovky B
vyrovnávací paměť	statická RAM	10-20 ns	stovky kB - jednotky MB

hlavní paměť	**dynamická RAM**	**50-70 ns**	**desítky MB - jednotky GB**
vnější paměť	**pevný disk**	**5-15 ms**	**jednotky - stovky GB**
záložní paměť	**optický disk, magnetická páska**	**stovky ms - stovky s**	**stovky GB - jednotky TB**

Virtuální paměť

- **mechanismus správy hlavní paměti**
- **motivace: při zpracování jsou části programů a data přesouvány do hlavní paměti, aby k nim mohl přistupovat procesor, ale programy a data občas nejsou třeba zavádět celé a jen nám zbytečně ubírají cenné místo v hlavní paměti → virtualizace nám umožní využít i vedlejší paměť, ze které si budeme v případě potřeby data vyvolávat, procesor však o tomto triku nic tušit nebude**
- **při virtualizaci jsou všechny adresy, se kterými procesor pracuje, pouze virtuální (logické), správa virtuální paměti má pak na starosti překlad na fyzickou adresu (tedy již na konkrétní adresu v hlavní nebo vedlejší paměti)**
- **virtualizace umožní využít větší prostor než nám fyzicky poskytuje hlavní paměť, data jsou v případě, že je procesor právě nepotřebuje uložena do vnější paměti (většinou HDD), v případě potřeby jsou nahrána zpět do hlavní paměti**
- **hlavní paměť = fyzický paměťový prostor**
- **logické adresové prostory jsou ve skutečnosti vnější paměti**
- **zajištění ochrany dat před neoprávněným přístupem a modifikací**

Existují dvě metody virtualizace paměti:

Stránkování

Při stránkování je paměť rozdělena na větší úseky stejné velikosti, které se nazývají *stránky*. Správa virtuální paměti rozhoduje samostatně o tom, která paměťová stránka bude zavedena do vnitřní paměti a která bude odložena do odkládacího prostoru (*swap*). Pro překlad adres se používá *tabulka stránek*, což je datová struktura v hlavní paměti či v cache procesoru, která pro každou virtuální stránku obsahuje záznam s informacemi zda se nachází v hlavní paměti a když ano, tak kde.

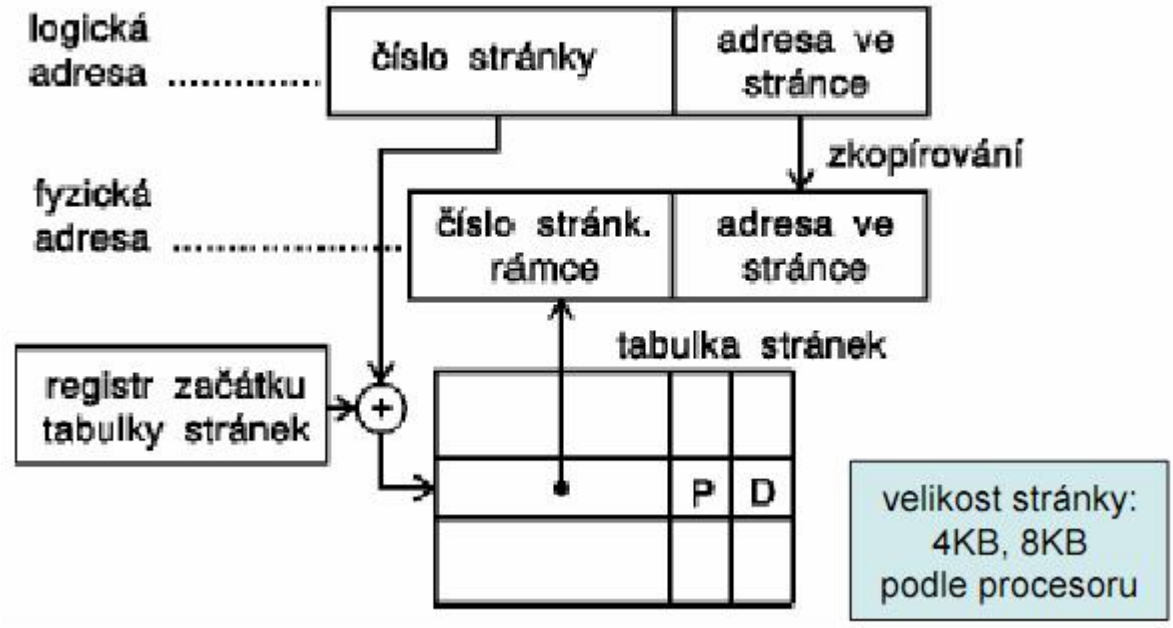
- **Logický adresový prostor je rozdělen na úseky pevné délky - stránky (logické stránky)**
- **Fyzický adresový prostor je rozdělen na stejně velké úseky - stránkové rámce (fyzické stránky)**
- **Logický adresový prostor je realizován ve vnější paměti. Data (úseky programu) se přesouvají do hlavní paměti po jednotlivých stránkách, jsou-li v průběhu výpočtu požadována a pokud se příslušná stránka již v paměti nenachází.**
- **Překlad (určení kam se stránka do hlavní paměti přesune) používá datovou strukturu ... tabulku stránek**

Tabulka stránek

- **uložena v hlavní paměti**
- **obsahuje pro každou logickou stránku jednu položku**
- **položka obsahuje informaci, zda se daná stránka nachází v hlavní paměti a pokud ano, tak kde (v kterém stránkovém rámci)**

Co všechno obsahuje záznam v *tabulce stránek*:

- **horní část fyzické adresy (viz obr.)**
- **příznak přítomnosti stránky v hlavní paměti**
- **příznak změny dat ve stránce (zda byla po dobu přítomnosti v *HP* do stránky zapisováno) tzv. *dirty bit***
- **další bity, např. určující, zda je vhodné stránku přepsat (vyhodit z hlavní paměti, podle toho jak a kdy byla použita)**

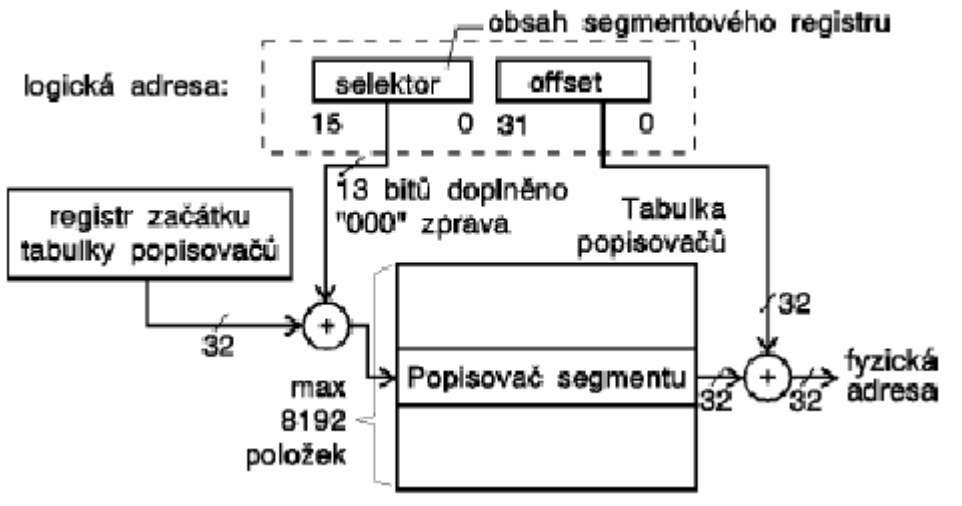


Stránkovací mechanismus:

- **je-li stránka přítomna v hlavní paměti, přeloží se logická adresa na fyzickou**
- **není-li přítomna, vyvolá se přerušení, přerušovací mechanismus vyvolá načtení stránky z vnější paměti**
- **pokud není v hlavní paměti volno, je třeba si místo udělat - přesunout nějakou stránku do vnější paměti, např. tu, se kterou se již dlouho nepracovalo**
- **nebylo-li do uvolňované stránky zapisováno (což poznáme podle *dirty bitu*), nemusíme ji nikam přesouvat a jen se přepíše**

Segmentace

- **segmenty jsou funkčně samostatné části programu proměnné délky, které lze do hlavní paměti zavádět v případě potřeby**
- **paměť je rozdělena na nestejně velké úseky - segmenty**
- **adresy v segmentu jsou relativní vůči začátku segmentu (tzv. *bázi*), což usnadňuje přemístění segmentu v hlavní paměti**
- **logická adresa se skládá z báze segmentu (*selektor*) a posunutí (*offset*)**
- **báze segmentu se nachází v**
  - **segmentovém registru**
  - **popisovači (deskriptoru) segmentu**



Selektor

- **horních 13 bitů selektoru je indexem do tabulky deskriptorů**

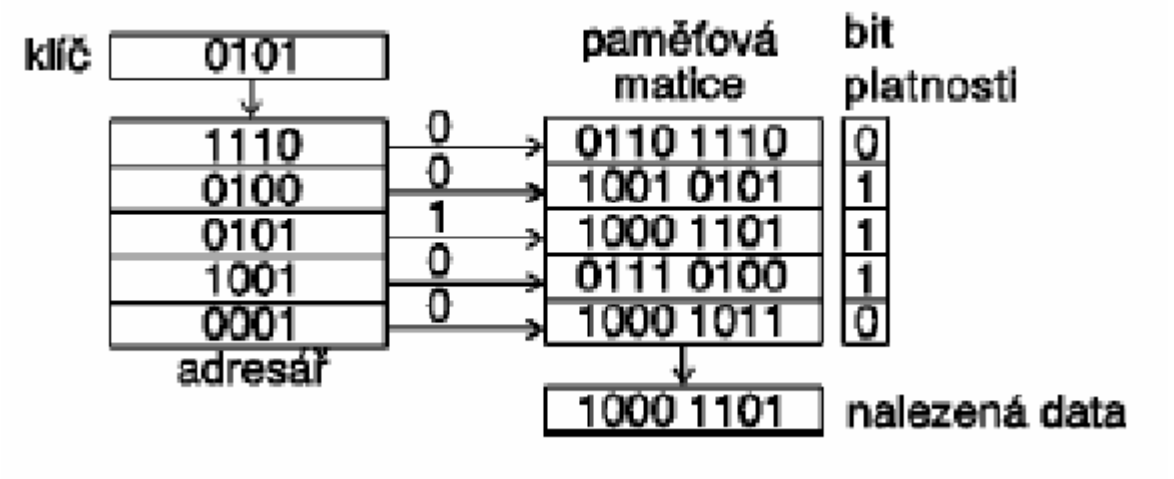
Deskriptor

- **32 bitová báze**
- **20-bitový limit segmentu**
- **Příznaky:**
  - **bit *granularity* (značí se G)**
    - **Pokud je G=0 20-bitový limit se počítá po bajtech, tzn.:Limit = 20 bitů = 1048575*bajt =+- 1 MB segment bude mít tedy limit 1MB.**
    - **Pokud bude G=1 tak se limit počítá po 4 KB, tzn.:Limit = 20 bitů = 1048575*Kb =+- 4096 MB segment bude tedy mít limit 4GB**
  - **typ segmentu (kódový, datový, systémový)**
  - **přístupová práva segmentu (READ/WRITE/EXECUTE - kombinace WRITE a EXECUTE je ovšem nenastavitelná)**

Asociativní paměť (Content-addressable memory)

Plně asociativní

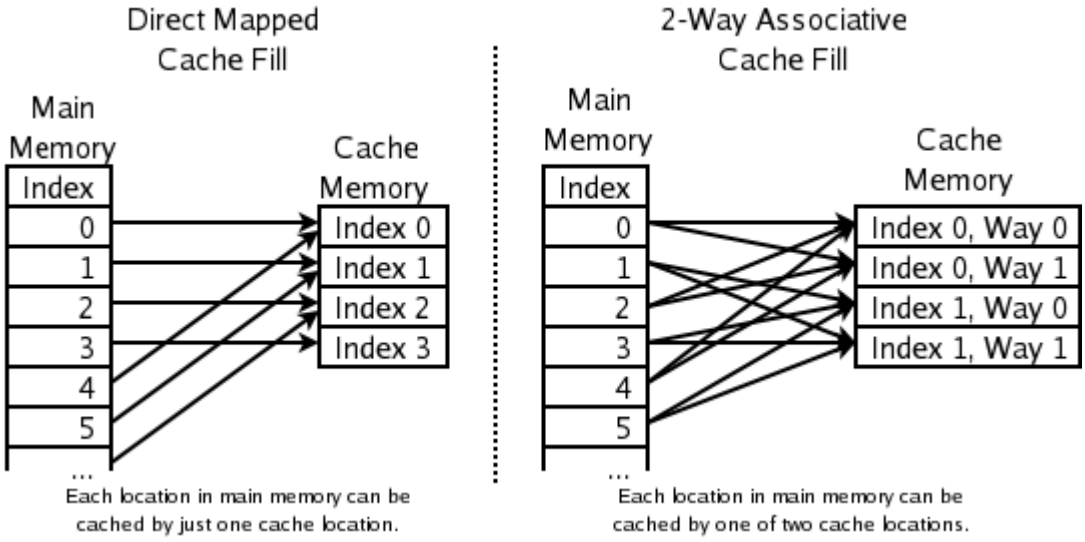
- **adresuje se částí datové položky (tzv. klíčem)**
- **paměť obsahuje tzv. adresář**
- **najednou porovnám klíč se všemi položkami v adresáři, když dojde ke shodě vezmou se z paměťové matice příslušná data (pokud jsou data platná, což nám určuje bit platnosti)**



- **nevýhody:**
  - **adresář je tvořen speciálními obvody**
  - **při stejné kapacitě trojnásobná plocha čipu**

S omezeným stupněm asociativity

- **každé položce je určeno místo (nebo několik míst - podle stupně asociativity), kde se může nacházet, toto místo je určeno částí adresy položky**
- **adresář je možné realizovat klasickou SRAM, čímž oproti plně asociativní paměti uspoříme místo**
- **na obrázku paměť se stupněm asociativity 1 (direct mapped) a 2 (2-way associative)**



Rychlá vyrovnávací paměť (cache)

Dnes typická paměť typu DDR400 operuje na pouze 200 MHz. Rychlost, s jakou je schopná dodávat data, začíná u jednoho hodinového cyklu a končí u desítek. Když vezmeme v úvahu, že této modelové situaci je jeden hodinový cyklus paměti roven deseti cyklům procesoru (frekvence procesoru je totiž desetinásobná), a ještě přičteme nějaké to zpoždění zapříčiněné řadičem pamětí a případně sběrnicí FSB, zjistíme, že načtení dat z modulu DIMM trvá běžně stovky hodinových cyklů procesoru:

Těch cyklů, při nichž procesor nic nedělá, protože musí čekat. A to není optimální. Pomalosti

pamětí DRAM si vývojáři všimli již před desítkami let, a tak se do procesoru přidává paměť typu SRAM, která se nazývá cache (vyrovnávací paměť). Tato paměť je mnohem rychlejší. Typicky pracuje na stejné frekvenci jako zbytek procesoru, navíc má ale přístupovou dobu v řádu běžně 2 až 20 cyklů - je tedy cca. 10x až 100x rychlejší než paměťový modul DIMM. Procesor pracuje pouze s cache, přímo s DRAM nikdy

- **jedno z využití asociativní paměti například v cache**
- **malá rychlá paměť zařazená mezi procesor a hlavní paměť**
- **využívá asociativního přístupu k položkám**
- **obsahuje kopie často používaných položek v hlavní paměti**
- **realizovaná pomocí SRAM**
- **klíčem je adresa položky**

Čtení

- **začne se číst zároveň z cache i z hlavní paměti, pokud se položka v cache nalezne, čtení z hlavní paměti se nedokončí. Pokud se nenalezne přečtou se data z hlavní paměti (a nejspíš se i zrovna uloží do cache).**

Zápis

- **pokud položka v cache není zapíše se zpravidla jen do hlavní paměti**
- **pokud je postupuje se dvěma způsoby**
  - **průběžný zápis (write through) - nová hodnota se zapíše zároveň do cache i hlavní paměti**
  - **odložený zápis (write back) - nová hodnota se zapíše jen do cache a při uvolňování položky z cache se musí její obsah přepsat do hlavní paměti**

Vstupní/Výstupní (V/V) nebo-li (I/O) zařízení

- **zařízení, která zprostředkovávají kontakt počítače s okolím**
- **procesor komunikuje s V/V zařízeními přes sběrnici pomocí řadičů (z hlediska programátora pak řadič vypadá jako sada hardwarových registrů, které jsou určeny pro čtení, zápis nebo čtení i zápis)**
- **při inicializaci počítače je potřeba zjistit, které řadiče jsou v počítači zapojené a inicializovat je. Při vstupu nebo výstupu dat je zpravidla nutné čtením z řadiče zjistit, zda je zařízení připraveno zapsat do řadiče instrukci a zapsat nebo přečíst data.**
- **podle architektury počítače můžeme V/V rozdělit na:**
  - **paměťově mapované**
    - **registry jsou adresovány jako paměť**
    - **přístupné pomocí běžných strojových instrukcí pro čtení a zápis do paměti**
  - **izolované**
    - **přístupné pomocí speciálních strojových instrukcí (zpravidla IN a OUT)**
    - **adresní prostory paměti a vstupně/výstupních zařízení jsou oddělené**

DMA

- ***direct memory access***

- **schopnost hardwaru přistupovat k paměti nezávisle na CPU**
- **bez DMA je procesor plně vytížen transferem dat a nezbývá mu čas na jiné úkony**
- **s DMA procesor iniciuje kopírování a věnuje se jiným úkonům, po skončení kopírování vyvolá DMA řadič přerušení**
- **DMA se používá u HDD, síťových karet, grafických karet...**
- **Typické použití DMA je přenos bloků dat z operační paměti do nebo z vyrovnávací paměti (cache) vstupně/výstupního zařízení**

Sběrnice ISA

- **přenos je proveden DMA řadičem, který je obvykle součástí čipové sady**
- **data proudí DMA kanály (starší počítače měly 4 novější 8)**

Sběrnice PCI

- **pro PCI architekturu neexistuje žádný centrální DMA řadič**
- **využívá se tu bus mastering, kdy zařízení převezme kontrolu nad sběrnicí a přenos provede samo**

Řadič procesoru

- **control unit, controller**
- **Pracuje podle instrukčního cyklu**
- **Řídí činnost všech výkonných jednotek počítače podle instrukcí a jejich kódu, podle instrukčního cyklu**
- **Je to sekvenční obvod – závisí na sekvenci vstupních (stavových) signálů, které generují výkonné jednotky (ALU, HP - instrukce) a vysílá jim řídící signály**
- **Pracuje v nekonečném cyklu – řídí zpracování instrukcí**
- **Navrhuje se podle instrukčního cyklu a výběru ISA – z grafu přechodů – vývojového digramu**

- **podle způsobu realizace existuje:**
  - **obvodový (klasický) řadič, který je realizován fyzickými sekvenčními obvody**
  - **mikroprogramový řadič s pamětí - jednotlivé dílčí operace, které se provádějí při zpracování instrukcí jsou uloženy v této paměti a říká se jim *mikroinstrukce***

Mikroprogramové vybavení (Firmware)

- **soubor mikroinstrukcí tvoří mikroprogram a soubor všem mikroprogramů je mikroprogramové vybavení (firmware)**

Pipelining

- **proudové zpracování instrukcí**
- **zpracovává instrukce po částech (princip výrobního pásu)**
- **jednotky jsou seřazeny do sérií, takže výstup jedné je vstupem druhé**
- **každá jednotka provede část operace, jednotky pracují současně**
- **např. předčítání instrukcí - jedna instrukce se čte, další dekóduje a další provádí**

- **v ideálním případě se v každém taktu dokončí jedna instrukce... v reálu ovšem dochází ke konfliktům:**
  - **datové - jsou potřeba data, která nejsou dosud uložena**
  - **skokové - adresu skoku zatím nelze určit**
- **konflikty se mohou řešit několika způsoby, nejčastějším je čekání (až budou data uložena, adresa známa,...)**

Procesory CISC a RISC

motivace: ***Jak navrhnout co nejrychlejší procesor?***

RISC

- ***reduced instruction set computers***
- **vsází na strategii „Méně je někdy více“**
- **obsahuje malý počet jednoduchých instrukcí (<128), které se velmi rychle provádějí (dokončení v jednom taktu)**
- **pipelining**
- **obvodový řadič**
- **malý počet formátů instrukcí a způsobů adresace**
- **velký počet registrů, které jsou však univerzálně využitelné**
- **používá se: architektura ARM, Power Architecture od IBM (např. procesory v Xbox360 nebo Playstation 3)**

Hlavní princip je pracovat s velkým počtem registrů. Omezit způsob adresace operandů aby to bylo co nejjednodušší a provádělo se to co nejrychleji a další důležitou věci je aby prováděné instrukce byly rychlé tak se musí provádět s daty, které jsou pokud možno v procesoru nebo co nejblíže aritmetické jednotce.Když už pracujeme s velkým počtem registrů tak aritmetické operacace budou probíhat jenom mezi registry a komunikace s pamětí bude probíhat pouze instrukci přesun kdy budeme jen natahovat potřebna data.

pozn. Power procesory su od IBM (napr. v starych radach Apple)

CISC

- ***complex instruction set computers***
- **název zvolen kvůli kontrastu s procesory RISC**
- **rozsáhlý soubor instrukcí (řádově stovky)**
- **malý počet registrů (obvykle <30)**
- **mohou pomocí jedné instrukce vykonat několik jednoduchých operací, tato instrukce je pak ovšem pomalejší**
- **dochází ovšem k sbližování obou konceptů, takže CISC převzala i některé aspekty z RISC (např. jednotaktové instrukce)**

CISC neboli počitače s rozsáhlým souborem instrukci a to jsou typický procesory , který maji mikroprogramovaný řadič, velkou paměť, spoustu instrukci, specialní instrukce pro nějakou delši aritmetiku.

Zdroje

Týkající se předmětu SAP

- **Stránky předmětu SAP [http://service.felk.cvut.cz/courses/Y36SAP/]**
- **Slajdy z přednášek SAP ke stažení [http://service.felk.cvut.cz/courses/Y36SAP/lectures.html]**
- **Videozáznam z přednášek paní Kubátové (předmět je velmi podobný našemu SAP) [http://www.avc-cvut.cz/avc.php?id=3619]**

Česká wikipedie

- **DMA [http://cs.wikipedia.org/wiki/DMA]**
- **IRQ [http://cs.wikipedia.org/wiki/IRQ]**
- **Vstup/výstup [http://cs.wikipedia.org/wiki/Vstup/výstup]**
- **Vyrovnávací paměť [http://cs.wikipedia.org/wiki/Vyrovnávací_paměť]**
- **Virtuální paměť [http://cs.wikipedia.org/wiki/Virtuální_paměť]**
- **John von Neumann [http://cs.wikipedia.org/wiki/John_von_Neumann]**
- **Přerušení [http://cs.wikipedia.org/wiki/P%C5%99eru%C5%A1en%C3%AD]**

Anglická wikipedie

- **DMA [http://en.wikipedia.org/wiki/Direct_memory_access]**
- **Asociativní paměť [http://en.wikipedia.org/wiki/Content-addressable_memory]**
- **IRQ [http://en.wikipedia.org/wiki/IRQ]**
- **Vstup výstup [http://en.wikipedia.org/wiki/Input/output]**
- **Cache [http://en.wikipedia.org/wiki/Cache]**
- **Virtuální paměť [http://en.wikipedia.org/wiki/Virtual_memory]**
- **RISC [http://en.wikipedia.org/wiki/RISC]**
- **CISC [http://en.wikipedia.org/wiki/Complex_Instruction_Set_Computer]**

Jiné zdroje

- **Root: článek o vyrovnávací paměti [http://www.root.cz/clanky/vyrovnavaci-pameti-cache/]**
- **Skripta z TU Liberec o předmětu podobného zaměření [http://www.nti.tul.cz/~kolar/os/os.pdf]**
- **http://www.fi.muni.cz/usr/pelikan/ARCHIT/TEXTY/OBSAH.HTML [http://www.fi.muni.cz/usr/pelikan/ARCHIT/TEXTY/OBSAH.HTML]**

spolecne/spol9.txt · Poslední úprava: 2010/06/15 18:32 autor: Petronyuk