Princípy činnosti práce číslicového počítača

Portál: edu.ukf.sk - Vzdelávací portál - Univerzita Konštantína Filozofa, Nitra

Kurz: Operačné systémy (KI/OS/15)

Kniha: Princípy činnosti práce číslicového počítača

Vytlačil(a): Zuzana Pavlendová

Dátum: Streda, 1 december 2021, 17:56

Opis

tekst

Obsah

Úvod: Princípy činnosti práce číslicového počítača

V tejto kapitole sa dozviete:

- Z akých podsystémov sa skladá číslicový počítač?
- Aké sú hlavné funkcie jednotlivých podsystémov počítača?
- Ako sa zobrazujú jednotlivé informácie v počítači?
- Akými metódami se určujú operandy v inštrukciách počítača?

Po jej preštudovaní by ste mali byť schopní:

- Charakterizovať činnosť počítača, rozumieť samočinnému spôsobu spracovania programov v číslicovom počítači.
- poznať spôsob zobrazovania jednotlivých typov informácií ukladaných v pamäti počítača. Pochopiť dvojkovú sústavu a základné operácie v tejto sústave, pochopiť zobrazovanie v pevnej a pohyblivej rádovej čiarke.
- Popísať rôzne metódy určovania operandov inštrukcií čo je veľmi dôležité pre pochopenie niektorých algoritmov operačných systémoch.

Kľúčové slová tejto kapitoly:

Von Neumannova schéma počítača, operačná pamäť, operačná jednotka, radič, vstupné a výstupné zariadenia, inštrukčný cyklus radiča, dvojková sústava, inštrukcia, operand.

Doba potrebná ku štúdiu: 6 hodin

Sprievodca štúdiom

Štúdium tejto kapitoly je pomerne náročné hlavne pre tých z Vás, ktorí doteraz nemajú žiadne znalosti z oblasti architektúry číslicových počítačov. V takomto prípade Vám zrejme niektoré princípy činnosti počítača budú pripadať menej pochopiteľné, ovšem nenechajte sa tým odradiť, pretože pochopením tejto časti sa Vámzjednoduší štúdium následujúcich kapitol.

Na štúdium tejto časti si vyhraďte aspoň 6 hodín. Odporúčame študovať s prestávkami vždy po pochopení jednotlivých podkapitol. Po celkovom preštudovaní a vyriešení všetkých príkladov odporúčame dať si pauzu, trebárs 1 deň, a potom sa pustite do vypracovania korešpondenčných úloh.

1.1 Úvod

Číslicové počítače sú stroje na spracovanie informácie. Obecne počítače sú predovšetkým matematickými strojmi, ktorých úlohou je transformácia číselných (diskrétnych) vstupných hodnôt, vstupných údajov. Sú ďalej strojmi elektronickými, čo určuje fyzikálne podmienky realizácie úloh transformácie číselných údajov. Konečne sú strojmi samočinnými, čo znamená, že uvedenú úlohu realizujú bez priamej účasti človeka. Základom pre samočinnosť funkcie je existencia pamäte počítača, v ktorej je uchovaný návod pre riešenie (program) a riadiacej jednotky, ktorá dokáže tento návod interpretovať vo fyzikálnych podmienkach.

Činnost počítača je možné charakterizovať veľmi obecne takto:

- Počítač prijíma na svojom vstupe texty nad vstupnou abecedou, transformuje ich na texty nad vnútornou abecedou (kódovanie vstupnej informácie do vnútornej reprezentácie) a ukladá ich do pamäti.
- V pamäti vykoná spracovanie.
- Spracované texty transformuje do výstupnej abecedy (interpretácia informácie zachytenej vo vnútornej reprezentácii) a vydává ich na svojom výstupe.

K tomu, aby počítač mohol komunikovať s okolím je nutný spoločný jazyk počítača a jeho okolia. Jedná se o jazyk, ktorému obidve strany jednoznačne a zhodne rozumejú, tzv. formálny jazyk.

Vymedzíme tri jeho určujúce zložky:

- **abecedu**, tj. výber prípustných základných symbolov,
- **syntax**, skladbu, tj. súbor pravidiel, podľa ktorých je možné zo symbolov abecedy tvoriť prípustné vyššie tvary (výrazy, slová, vety),
- **sémantiku** tj. význam, ktorý zhodne obidve komunikujúce strany syntaktickým tvarom priraďujú.

1.2 Von neumannova schéma počítača

Von Neumannova schéma bola navrhnutá v roku 1945 americkým matematikom (narodeným v Maďarsku) Johnom von Neumannom ako model samočinného počítača. Tento model s určitými výnimkami zostal zachovaný dodnes.

Princíp činnosti počítača podľa von Neumannovej schémy:

1. Do operačnej pamäti sa pomocou vstupných zariadení cez ALU umiestni program, ktorý bude vykonávať výpočet.
2. Rovnakým spôsobom sa do operačnej pamäti umiestnia údaje, ktoré bude program spracovávať.
3. Prebehne vlastný výpočet, ktorého jednotlivé kroky vykoná ALU. Táto jednotka je v priebehu výpočtu spolu s ostatnými modulmi riadená radičom počítača. Mezivýsledky výpočtu sú ukladáné do operačnej pamäti alebo do registrov procesora.
4. Po skončení výpočtu sú výsledky posielané cez ALU na výstupné zariadenie.

Z hľadiska systémového sa výpočtový systém skladá z:

1. pamäťového podsystému,
2. operačného podsystému,
3. riadiaceho podsystému,
4. vstupného a výstupného podsystému.

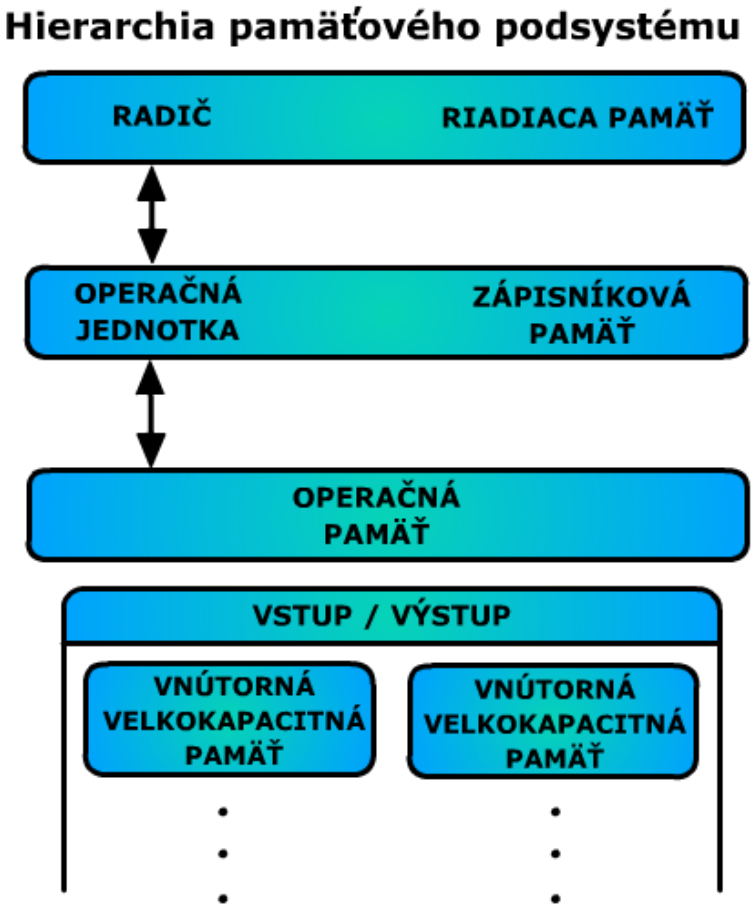
Centrálnym podsystémom je **pamäť.** Preberá informácie od vstupného podsystému a predáva ich výstupnému podsystému. Riadiaci podsystém (radič) získava z pamäti inštrukcie programu a prideľuje ostatným podsystémom postupne úlohy. Operačný podsystém získava podľa pokynov riadiaceho podsystému z pamäťového podsystému údaje, spracováva ich a vracia späť do pamäti. Pamäťový podsystém je schopný si zapamätať tj. uchovať bez zmeny určité množstvo informácie. Podsystém tvorí:

- blok adresovatelných pamäťových buniek,
- adresový register pamäti (register adresy pamäti – RAP),
- údajový register pamäti (register údajov pamäti – RDP),
- radič pamäti.

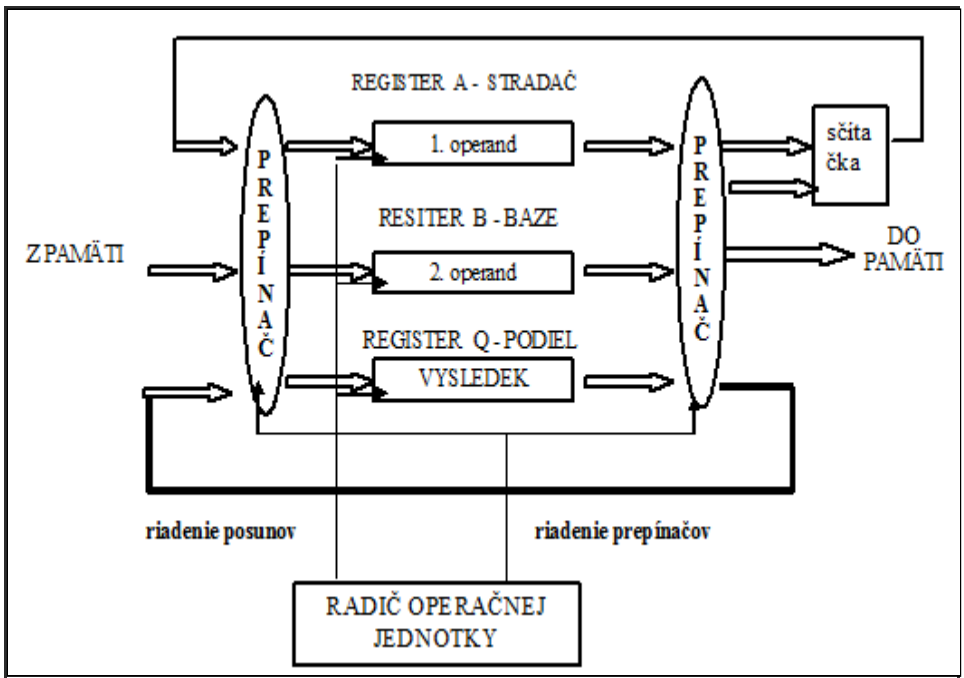
Funkcia každej pamäťovej buňky je daná nasledujúcimi predpokladmi:

- Obsahuje jedno slovo – slabiku – byte.
- Pri získaní údaja čítaním sa obsah neporuší.
- Je možné do nej zapísať novú informáciu s podmienkou, že sa zruší existujúca informácia.
- Má jednoznačne pridelenú adresu.

Požiadavky, ktoré sa v počítačovom systéme kladú na pamäte sa zatiaľ nedajú ekonomicky splniť jedinou pamäťou. Čím väčšia je kapacita pamäte, tým väčšia je totiž i jej pomerná cena. Podobne rastie cena i so skracovaním vybavovacej doby. Preto rýchle pamäte (s krátkou vybavovacou dobou) majú obvykle malú kapacitu, pomalé pamäte (s dlhšou vybavovacou dobou) majú veľkú kapacitu. Začlenenie jednotlivých úrovní pamäte je naznačené na obrázku.



Operačný podsystém vykonáva všetky aritmetické a logické transformácie. Pritom spolupracuje s pamäťou, z ktorej vyberá operandy a ukladá do nej výsledky. Operaciu, ktorá sa má vykonať, určuje riadiaca jednotka. Typické zloženie sériovej operačnej jednotky je na obrázku.



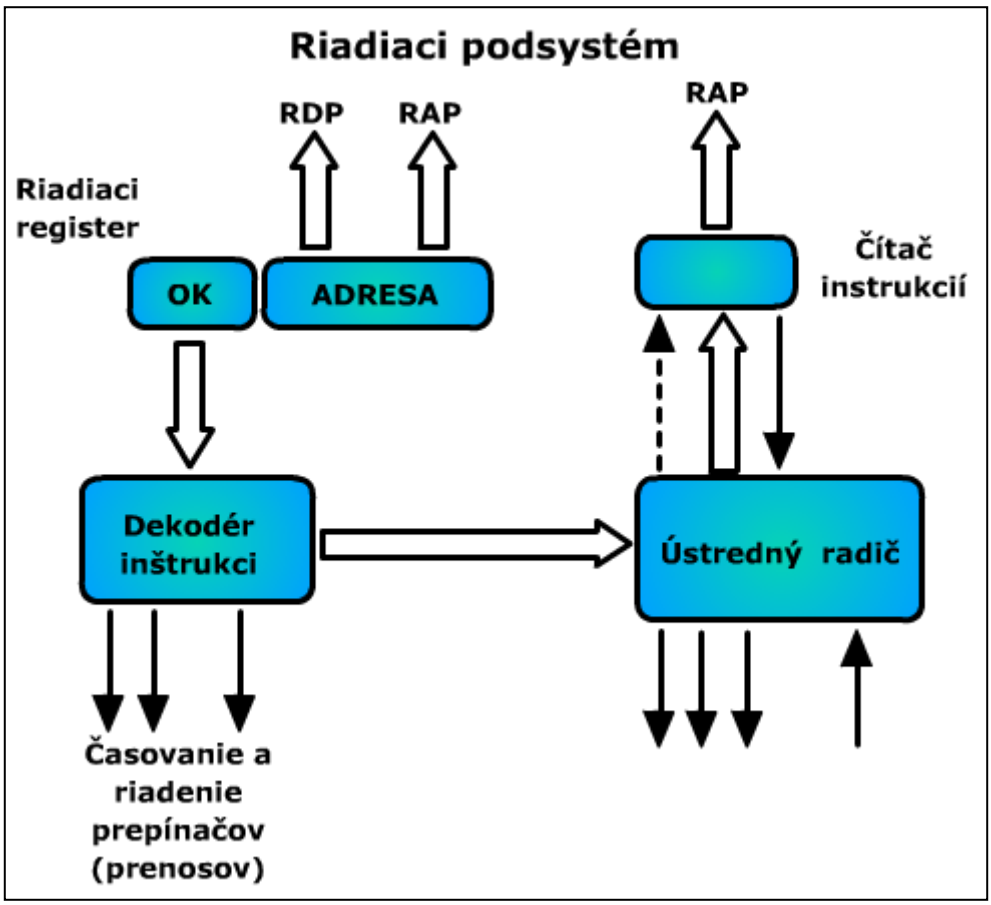
Operačný podsystém

Tri registre, každý na jedno slovo, tvoria dočasnú pamäť. Registre uchovávajú operandy, mezivýsledky a konečné výsledky. Označenie registrov vychádza z ich funkcie:

- RA - stradač (acumulator),
- RB - báza (base),
- RQ - podiel (quotient).

Operačná jednotka obsahuje ďalej sčítačku, ktorá má i prostriedky pre vytváranie doplnku, čím umožňuje odčítanie. Prepínač riadi tok informácií a umožňuje priamy zápis informácie z jedného registra do druhého, priechod informácie z dvoch registrov sčítačkou, presun údaja z pamäti do niektorého registra apod. Radič operačnej jednotky je autonómny. Dozerá na činnosť operačnej jednotky. Akonáhle sa mu odovzdá inštrukcia, ktorá sa má vykonať, riadi a časovo sleduje vykonávanie príslušného algoritmu. Okrem sériových operačných jednotiek sa samozrejme používajú sériovo-paralelné alebo paralelné, ktoré sú síce rychlejšie, ale zložitejšie a teda aj drahšie. (Pozn. Operačné jednotky súčasných mikroprocesorov obsahujú podstatne väčší počet registrov.)

Riadiaci podsystém (radič) riadi, resp. dozerá na veškerú činnosť a spoluprácu všetkých podsystémov počítača tak, aby celý systém počítača pracoval podľa zadaného programu. Dostáva inštrukcie programu, spracováva ich a transformuje ich na postupnosť príkazov pre ostatné časti počítača. Týmito príkazmi prideluje úlohy iným podsystémom. Pritom je stále informovaný o okamžitom stave všetkých podsystémov. Základné funkčné jednotky riadiaceho podsystému sú uvedené na obrázku.



Riadiaci podsystém pracuje v dvoch fázach:

výberovej a vykonávacej.

1.3 Zobrazenie informácií v počítač

Informácie, ktoré číslicové počítače spracovávajú, musia byť kódované a s ohľadom na technickú realizáciu je nutné k ich vyjadreniu nájsť lacné a spoľahlivé prvky. Z tohoto hľadiska sú zatiaľ stále najvhodnejšie prvky s dvomi jednoznačne rozlíšiteľnými stavmi. Tak sa dostávame k dvojkovej číselnej sústave, ktorá je dominujúcou sústavou strojového spracovania informácií. Dvojková číslica (0 alebo 1) je bit (elementárna informácia). Znaky *(*alfabetické a numerické) sa vyjadrujú väčšinou osembitovou skupinou. Pre tieto niekoľkobitové skupiny sa používá termín slabika alebo byte. Najbližšia väčšia skupina s pevne stanoveným počtom bitov je slovo. Dĺžka slov býva väčšinou 4 slabiky, tj. 32 bitov. Ak je v počítači pre všetky operácie slovo uložené vždy v jednej bunke pamäte, potom ide o počítač slovne orientovaný. Pokiaľ slovo môže byť uložené v niekoľkých bunkách (po slabikách), potom počítač nazývame slabikovo orientovaný.

1.3.1 Číselné sústavy a kódy

Z číselných soustav je nejbežnejšia dvojková, ktorá ako z hľadiska technického, tak i programového vybavenia stroja má značné výhody. Z hľadiska technického sú to napr. jednoduché algoritmy aritmetických a logických operácií a ich realizácia, v porovnaní s inými sústavami menší požadovaný rozsah pamäte ap. Z programového hľadiska je treba oceniť možnosť delenia a násobenia mocninami dvoch prostým posunom čísla. Hlavnou nevýhodou dvojkovej sústavy je potreba prevodu pri vstupe a výstupe z desiatkovej sústavy do dvojkovej a naopak. Desiatková sústava má prednosť práve v tom, že uvedené prevody odpadajú. Preto sa u väčšiny počítačov jednotlivé desiatkové číslice kódujú do dvojkovej sústavy tzv. dvojkovodesiatkovým kódom (BCD).

V rade počítačov sa používa dvojkovo kódovaná desiatková sústava. Každá desiatková číslica sa pritom musí vyjadriť. dvojkovo. Je zrejmé, že musíme použiť pre každú číslicu v desiatkovej sústave nejmenej 4 číslice dvojkovej sústavy, ale môže ich byť aj viacej.

Voľba konkrétneho kódu závisí na jeho použití. Iné požiadavky sa kladú na kódy, s ktorými sa majú vykonávať aritmetické operácie a iné na kódy, používané pri prenose informácií apod.

Pre aritmetické operácie sú najvhodnejšie kódy, ktoré vyhovujú tzv. **Aikenovým podmienkám**:

1. Každé miesto v kóde má mať určitú váhu.

2. Súčet váh miest, v ktorých je dvojková číslica kódu rovná 1, má dať hodnotu priradenej desiatkovej číslici alebo aspoň, väčšej desiatkovej číslici má odpovedať väčšie dvojkové číslo príslušného váhového kódu.

3. Vzťah medi párnymi a nepárnymi kódmi a priradenými desiatkovými číslicami môže byť síce ľubovolný, ale u zvoleného kódu má byť nemenný.

4. Desiatkovým doplnkom desiatkových číslic majú zodpovedať doplnkové kódy, ktoré vznikli inverziou jednotlivých bitov pôvodných kódov.

U kódov určených prr prenos ap. nás zaujíma predevšetkým bezpečnosť kódov voči poruchám, možnosť zaistenia vzniknutých chýb a príp. odstranenia. V následujúcej tabuľke sú uvedené niektoré známé kódy, z ktorých však všetky nespĺňajú vyššie uvedené podmienky.

	8421	2421	(8421)+3	5421	3n+2	5043210 Bikvinárne
0	0000	0000	0011	0000	00010	0100001
1	0001	0001	0100	0001	00101	0100010
2	0010	0010	0101	0010	01000	0100100
3	0011	0011	0110	0011	01011	0101000
4	0100	0100	0111	0100	01110	0110000
5	0101	1011	1000	1000	10001	1000001
6	0110	1100	1001	1001	10100	1000010
7	0111	1101	1010	1010	10111	1000100
8	1000	1110	1011	1011	11010	1001000
9	1001	1111	1100	1100	11101	1010000

Tabuľka dvojkových kódov pre desiatkové číslice

Z viacej možných spôsobov zabezpečenia informácie (kódov) sa stručne zmienime len o jednej, tzv, kontrole paritou. Podstata zabezpečenia spočíva v tom, že v najjednoduchšom prípade sa k informácii, ktorá je v **n**-rádoch, pridružuje informácia **n(n+1)**-tom ráde, ktorou sa dopĺňa počet jednotkových bitov na párny a nepárny počet (párna alebo nepárna parita). V nasledujúcej tabuľke sú naznačené **n**=5 bitové informácie s párnou paritou v 6. ráde.

n	**n+1**
10110	1
01001	0
01111	0
01011	1
01001	0

Príklad zabezpečenia informácie párnou paritou

Pri prenose informácie sa po každom prepise kontroluje, či počet jedničiek v informácii je párny. V prípade lichého počtu sa signalizuje chyba. Je potrebné pripomenúť, že paritný bit sa k informácii pridáva pri tvorbe informácie, napr. na pamäťové médium.

1.3.2 Kódovanie v dvojkovej sústave

Spôsob zobrazenia čísiel v počítači nazývame číselný kód. Ukážeme si tri spôsoby kódovania čísiel v dvojkovej sústave. Je zrejmé, že kladné a záporné čísla sú zobrazené v počítači rovnako, až na znamienko. Aby počítač pracujúci v dvojkovej sústave poznal znamienko čísla, musíme tieto znamienka vyjadriť znakom, ktorý počítač dokáže rozlíšiť, tj. v našom prípade nulou alebo jedničkou (pozn., ak pracujeme s inou číselnou sústavou, kóduje sa znamienko zvláštnym znakom). Z dôvodu zjednodušenia budeme uvažovať o zobrazení čísla s pevnou rádovou čiarkou (s čiarkou pred najvyšším rádom, čo značí |x| <1).

1.3.2.1 Priamy kód

Kladné číslo bude mať v priamom kóde pred pevnou rádovou čiarkou, tzn. v znamienkovom ráde, nulu, číslo záporné jedničku. Čísla kladné sú teda vyjádrené v rozmedzí od nuly do jednej, čísla záporné od jednej do dvoch, čo je možné písať:

$(x)_{priamy\ kód} = x$ pro $x >= 0$

$(x)_{priamy\ kód} = 1-x$ pro $x < 0$.

Zapíšeme teda číslo + 7/32 ako 0,00111, ale číslo -7/32 ako

1,00111 = (1-(-0.00111)).

2.3.2.2 Doplnkový kód

Doplnkový kód nezáporného čísla (kladné a nula) **x** je rovnako ako u priameho kódu rovný číslu **x**. Doplnkový kód záporného čísla **x** dostaneme tak, že pred pevnú rádovú čiarku napíšeme znamienkový znak 1, na všetkých ostatných miestach nahradíme nuly jedničkami.

Obecne môžeme písať:

$(x)_{doplnkový\ kód} = x$ pro $x >= 0$

$(x)_{doplnkový\ kód} = 2 + x$ pro $x < 0$.

Zapíšeme teda + 0,75 ako 0,110

ale číslo - 0,75 ako 1,010

pretože 2_{10} v binárnej sústave je 10_2

potom 10,000		1,001
- 0,110	alebo	+ 1
1,010		1,010

Nula má jediné vyjadrenie 0,000 ... 0.

Maximálne záporné číslo 1,000 ... 0 nemá teda kladný ekvivalent (jeho dvojkový doplnok je totiž zase 1,000 ... 0).

1.3.2.3 Inverzný kód

Inverzný kód kladného čísla x je rovný číslu x. Inverzia záporného čísla vznikne tak, že sa do znaku napíše jednička a v číslicových rádoch sa zamenia jedničky za nuly a naopak. Je teda záporné číslo v inverznom kóde v poslednom číslicovom ráde o jedničku menšie než číslo v doplnkovom kóde. Teda:

$(x)_{\text{inverzný kód}} = x$ pro $x >= 0$

$(x)_{\text{inverzný kód}} = 10 + x - 10^{-n}$ pro $x < 0$.

Príklad:

+ 0,8128 je v dvojkovej sústave 0,1101 a - 0,8125 je v inverznom kóde 1,0010

1.3.2.4 Modifikovaný doplnkový kód

Modifikovaný doplnkový kód je obecnejšou verziou doplnkového kódu.

Je definovaný

$(x)_{\text{modifikovaný doplnkový kód}} = x$ pro $x >= 0$

$(x)_{\text{modifikovaný doplnkový kód}} = z^2 + x$ pro $x < 0$.

V dvojkovej sústave je $z^2 = (100)_2$

Príklad:

$(+ 11/16)_{10} = (00,1011)_2$

$(- 11/16)_{10} = (11,0101)_2$ pretože

$$\begin{array}{r} 100,0000 \\ -\ 00,1011 \\ \hline 11,0101 \end{array} \quad \text{alebo} \quad \begin{array}{r} 11,0100 \\ +\quad 1 \\ \hline 11,0101 \end{array}$$

Modifikovaný doplnkový kód nezáporného čísla **x** je rovný tomuto číslu **x**, pričom v znamienkových rádoch sú dve nuly; u záporného čísla sa pred pevnou rádovou čiarkou napíšu dve jedničky, na všetkých ostatných miestach nahradíme nuly jedničkami a jedničky nulami a potom k najnižšiemu rádu pripočítame jedničku.

2.3.2.5 Modifikovaný inverzný kód

Je definovaný

$(x)_{\text{modifikovaný inverzný kód}} = x$ pre $x >= 0$

$(x)_{\text{modifikovaný inverzný kód}} = 100 + x - 10^{-n}$ pro $x < 0$.

Modifikovaný inverzný kód nezáporného čísla **x** je rovný číslu **x**; u záporného čísla sa pred pevnú rádovú čiarku napíšu dve jedničky a v číslicových rádoch sa zamenia jedničky za nuly a naopak.

Príklad:

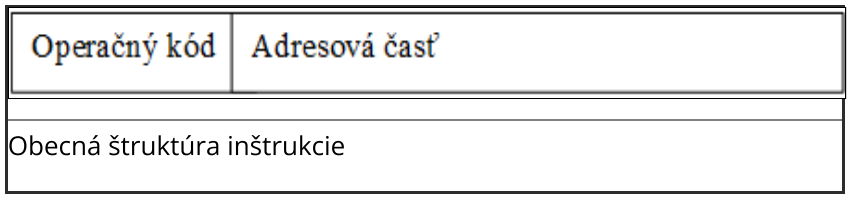
$(+ 11/16)_{10} = (00,1011)_2$

$(- 11/16)_{10} = (11,0100)_2$

1.3.3 Zobrazenie inštrukcie

Požiadavka na vykonanie operácie operačnou jednotkou, prípadne inými podsystémami počítača je zakódovaná do inštrukcie. Inštrukcia určuje, o ktorú z operácií ide (tzn. operačný kód) a kde sú uložené operandy v pamäti, registroch ap. (tzn. adresy).

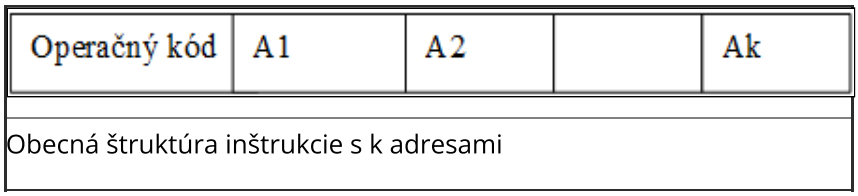
Obecná štruktúra inštrukcie je na následujúcom obrázku.



Obecná štruktúra inštrukcie

Adresová časť obsahuje toľko samostatných častí, koľko má inštrukcia adries (väčšinou 1,2 alebo 3) - hovoríme potom o jednoadresnej, dvojadresnej alebo trojadresnej inštrukcii.

Inštrukcia s k = 1,2 ..... adresami majú formát podľa následujúceho obrázka.



Obecná štruktúra inštrukcie s k adresami

Porovnanie efektívnosti inštrukcie s rôznym počtom adries budeme ilustrovať na príklade výpočtu výrazu:

Y = A * B + (C – D) * E / F

kde A,B,C.D,E,F,Y - premenné, uložené v bunkách s adresami a,b,c,d,e,f,y.

Budeme predpokladať, že výsledok ľubovolnej operácie sa ukladá v registri R základnej jednotky, a môže byť použitý v roli operandu v nasledujúcej operácii. Pre uloženie mezivýsledkov P1, P2... máme k dispozícii pracovné pamäťové bunky p1, p2...

Najprirodzenejší pre výpočet aritmetických a logických výrazov je trojadresný systém inštrukcií.

OK	A1	A2	A3

Štruktúra trojadresnej inštrukcie

Tu = operačný kód A1, A2 adresa dvoch operandov, A3 je adresa výsledku.

Inštrukcia				Význam
Násob	a	b	p1	A * B -> P1
Odčítaj	c	d	-	C - D -> R
Násob	-	e	-	R * E -> R
Del	-	f	-	R / F -> R
Sčítaj	-	p1	y	R + P1 -> Y

Pomlčka v poli adresy označuje, že táto adresa sa nepoužije aj keď v poli adries v inštrukcii sa nachádza. Pretože každá inštrukcia potrebuje tri m-bitové polia adries, tak pre adresáciu informácie je potrebné K3 = 3 x 5 = 15 adries, aj keď len 9 je ich využitých efektívne. Pri vykonaní programu sa T3 = 14 krát obraciame k pamäti:

- 5x výber inštrukcie,
- 9x čítanie a zápis operandov.

Pri použití dvojadresných inštrukcií

OK	A1	A2

Štruktúra dvojadresnej inštrukcie

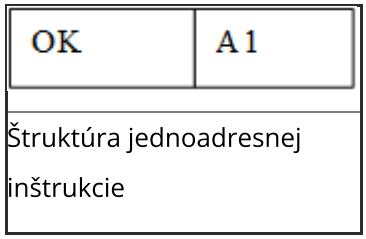
je proces výpočtu výrazu popísaný nasledujúcim programom.

Inštrukcia			Význam
Násob	a	b	A*B -> R
Zapíš	p1	-	R->P1
Odčítaj	c	d	R-D -> R
Násob	-	e	R*E -> R
Del	-	f	R/F -> R
Sčítaj	P1	y	R+P1 -> Y

Tu Zapíš je operácia zápisu výsledku R do pamäte. Program je zostavený zo šiestich inštrukcií, zaberajúcich K2 = 2x6 = 12 adries a 9 z nich je využitých efektívne. Pri vykonávaní programu sa T2 = 15x obraciame k pamäti:

- 6x výber inštrukcie,
- 9x čítanie / zápis operandov.

Pri práci s jednoadresnou inštrukciou



Štruktúra jednoadresnej inštrukcie

program výpočtu výrazu bude:

Inštrukcia **Význam**

Inštrukcia		Význam
Ulož	a	A -> R
Násob	b	R * B -> R
Zapíš	p1	R -> P1
Ulož	c	C -> R
Odčítaj	d	R - D -> R
Násob	e	R * E -> R
Del	f	R / F -> R
Sčítaj	p1	R + P1-> R
Zapíš	y	R -> Y

Program je zostavený z 9 inštrukcií zaberajúcich 9 adries. Pri vykonávaní programu T1 = 18x obracia k pamäti:

- 9x výber inštrukcie,
- 9x čítanie / zápis operandov.

Počet adries inštrukcií k	1	2	3
Počet adries v programe K	9	12	15
Počet prístupov k pamäti T	18	15	14

Z tabuľky je vidieť, že s väčším počtom adries v inštrukcii sa zväčšujú požiadavky na veľkosť pamäte pre adresu, ale znižuje sa počet prístupov k pamäti, tj. znižuje sa doba výpočtov. So zvyšovaním počtu operandov vo výraze odpadá nutnosť každú operáciu vykonávať tromi adresami; výsledok operácie môže byť uložený v procesore a môže byť použitý ako operand v následujúcej operácii. Vo vedecko-technických úlohách, na jednu operáciu, prisluší v priemere 1,2 až 1,4 operandov, uložených v operačnej pamäti a v úlohách spracovania hromadných údajov 1,6 až

1,8 operandov. Preto pre vedecko-technické úlohy sú výhodné 1-adresné inštrukcie a pre spracovanie údajov 2-adresné inštrukcie. Z toho dôvodu univerzálne počítače nejviac používajú 2-adresný systém inštrukcií a u mini a mikropočítačov 1-adresný systém inštrukcií.

1.3.4 Obrazenie abecedno-číslicových znakov

Súčasné počítače (až na výnimky) môžu pracovať s úplnou abecedou písmen, číslic a rôznych symbolov, (tzn. 26 písmen mezinárodnej abecedy, 10 číslic a interpunkčné znamienka, panelové znaky pre niektoré vstupné a výstupné zariadenia). Základná abeceda hlavne v slovanských zemiach býva doplnená písmenami národnej abecedy. V pamäťových médiách samočinného počítača sa spravidla uchováva až dvakrát toľko číselnej informácie ako alfabetickej informácie. Peoto je žiadúce, aby vybraný kód bol efektívnejší pre vyjadrovanie číslic. Túto podmienku spĺňa napr. osembitový kód EBCDIC (Extended Binary Coded Decimal Interctange Code), ktorý umožňuje vyjadriť 206 rôznych znakov. Každý znak teda predstavuje skupina osmich bitov. Výhoda osemibitového kódu spočívá v tom, že umožňuje v jednej osemibitovej skupine uchovávať kódy dvoch desiatkových číslic. Tieto číslice vstupujú do stroja zakódované osmimi bitmi; štyrmi bitmi ľavé časti tvoria tzv. zónu, ktorá svojou hodnotou (1111) určuje, že znak je číslica. Znaky sa dajú zreťaziť a v poslednej osembitovej kombinácii sa miesto zóny udáva znamienko čísla. Desiatkové číslice 0, 1, 2, ... 9 sú zakódované v priamom dvojkovom kóde 8421. Uvedený tvar je "rozvinutý" a používá sa pre styk s perifémymi zariadeniami. Číslicová informácia sa pred uložením do pamäte preskupý do zhusteného tvaru. Zhustený tvar sa uplatňuje pri vnútorných operáciách s desiatkovými číslami.

	0-	1-	2-	3-	4-	5-	6-	7-
-0	NUL	DLE	SP	0	@	P	`	p
-1	SOH	DC1	!	1	A	Q	a	q
-2	STX	DC2	"	2	B	R	b	r
-3	ETX	DC3	#	3	C	S	c	s
-4	EOT	DC4	$	4	D	T	d	t
-5	ENQ	NAK	%	5	E	U	e	u
-6	ACK	SYN	&	6	F	V	f	v
-7	BEL	ETB	'	7	G	W	g	w
-8	BS	CAN	(	8	H	X	h	x
-9	HT	EM	)	9	I	Y	I	y
-A	LF	SUB	*	:	J	Z	j	z
-B	VT	ESC	+	;	K	[	k	{
-C	FF	FS	,	<	L	\	l	|
-D	CR	GS	-	=	M	]	m	}
-E	SO	RS	.	>	N	^	n	~
-F	SI	US	/	?	O	_	o	DEL

Znaková sada ASCII

1.3.5 Pevná a pohyblivá rádová čiarka

Väčšina moderných počítačov umožňuje spracovávať čísla v dvojakom tvare, a to v pevnej a pohyblivej rádovej čiarke, viz nasledujúci obrázok.

Obecné usporiadanie čísla v pevnej rádovej čiarke

Obecné usporiadanie čísla v pohyblivej rádovej čiarke

Pre zobrazenie čísiel v pevnej rádovej čiarke je charakteristické pevné umiestnenie rádovej čiarky na určité miesto. Spravidla sa umiestňuje pred najvyšším rádovým miestom. Môžeme teda zobraziť čísla v intervale <0,1>.

Zápis čísla v pohyblivej rádovej čiarke má dve časti: mantisu a exponent.

Obecný zápis je $x = m.z^{e}$,

teda kde

x - zobrazované číslo,

m – mantisa,

e – exponent,

z - základ použitej číselnej sústavy.

Je možné tak zobraziť čísla v rozsahu $<z^{-(emax+m)}; z^{emax}>$

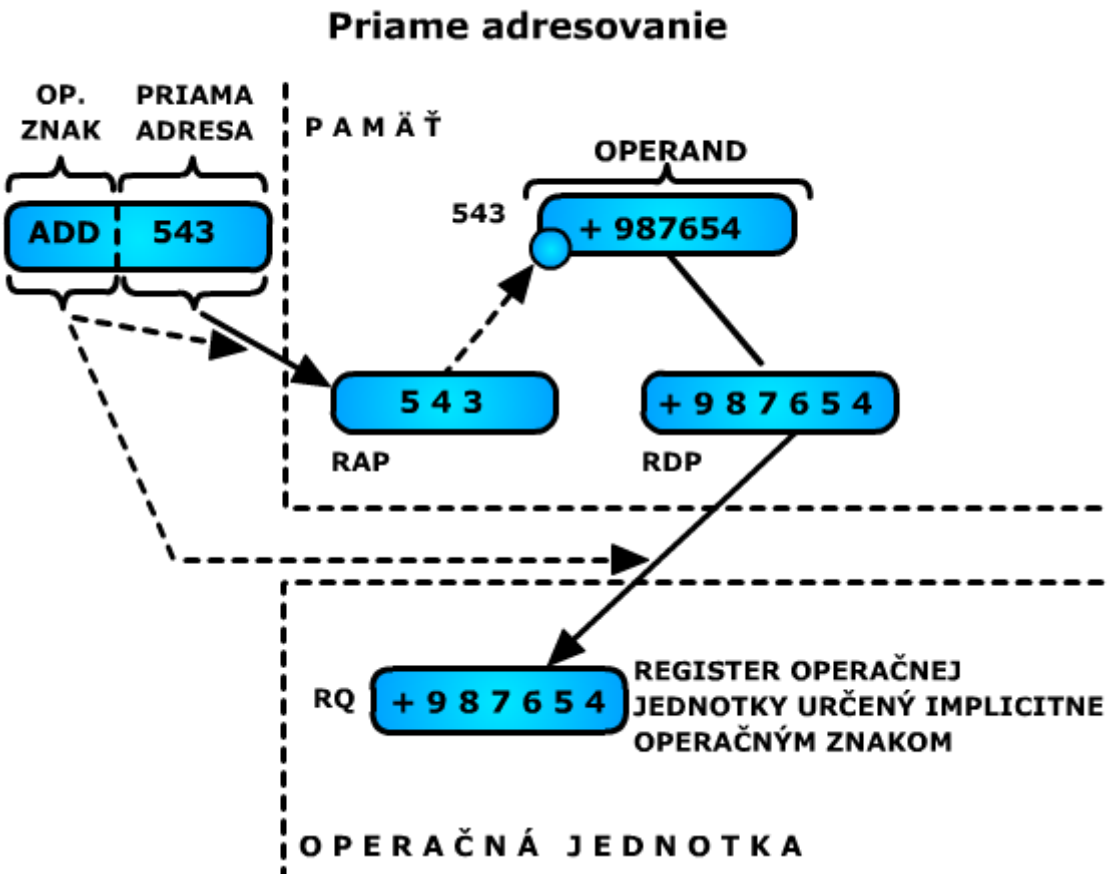
kde emax je maximálna hodnota exponentu, ktorá je daná počtom rádových miest exponentu v danom zobrazení, m je počet radových miest mantisy. Rozsah zobrazenia informácie v pevnej rádovej čiarke úzko súvisí s dĺžkou slova počítača.

1.4 Metódy určovania operandov

V číslicových počítačoch sa stretávame s rôznymi typmi určovania operandov. Sú to:

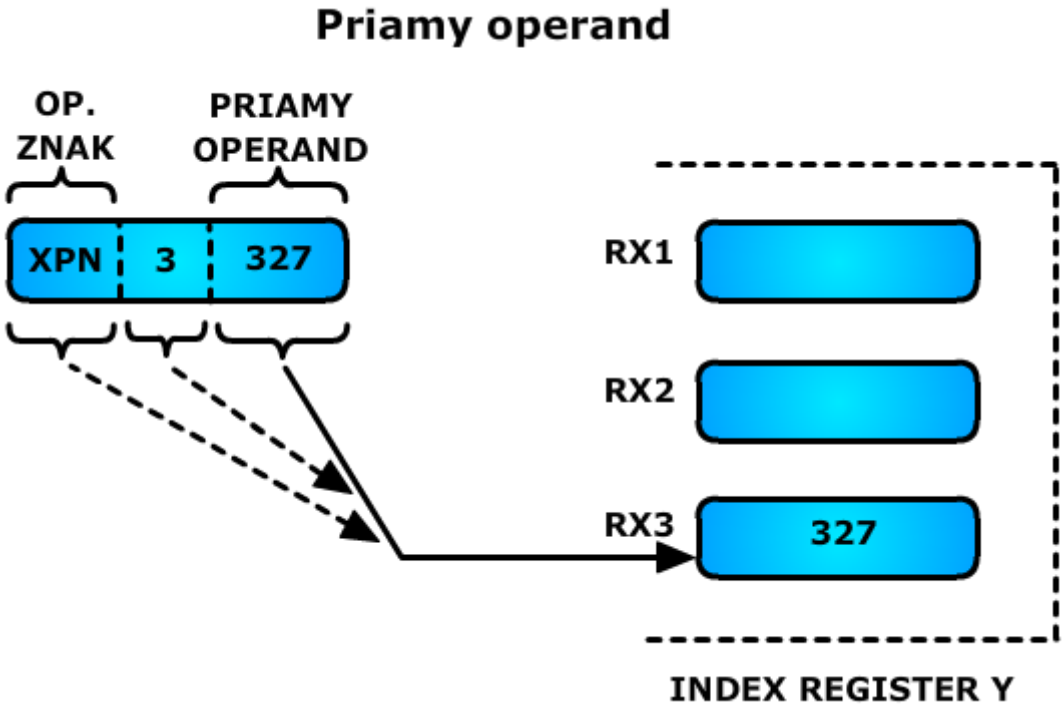
- Priame adresovanie.
- Priamy operand.
- Nepriame adresovanie.
- Implicitné adresovanie.
- Implicitný operand.
- Modifikované adresovanie.
- Relatívne adresovanie.
- Asociatívne adresovanie.

1.4.1 Priame adresovanie



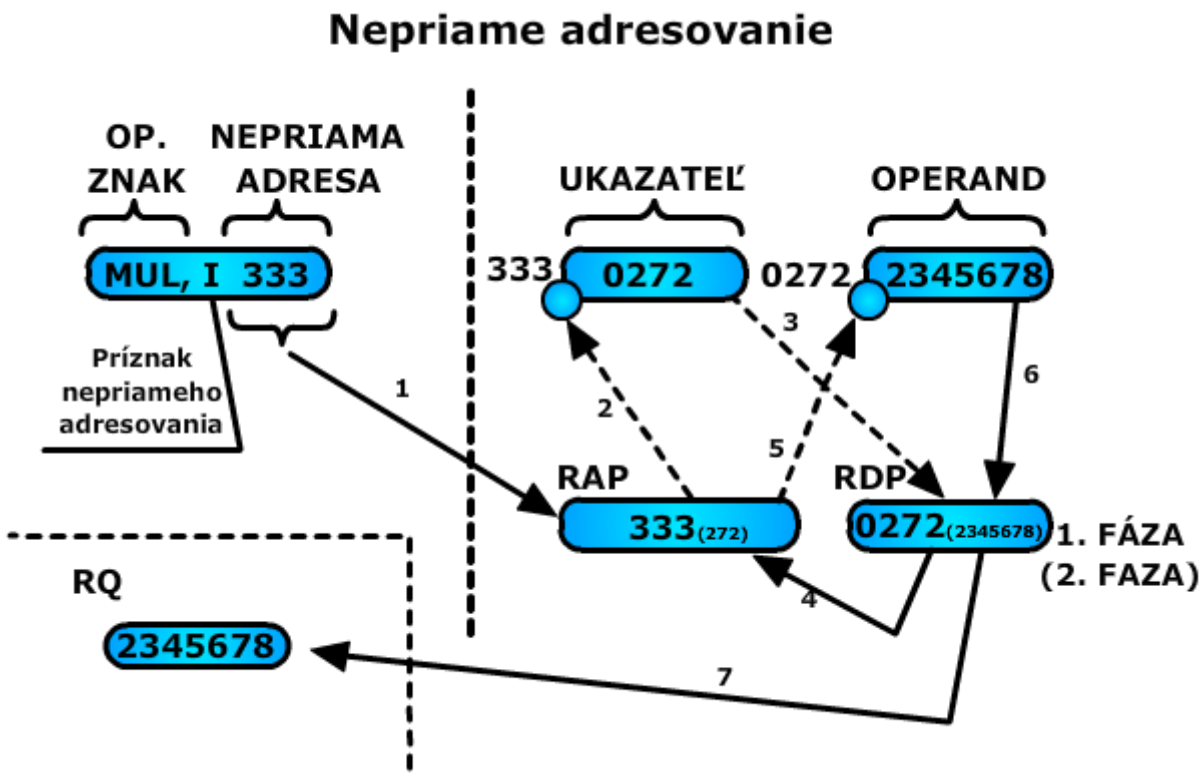
1.4.2 Priamy operand

V tomto prípade je (obvykle na mieste adresy) v inštrukcii zapísaný vlastný operand. Rozlíšenie, či ide o adresu alebo operand, musí byť dané operačným kódom. Vykonanie inštrukcie na nasledujúcom obrázku sa presunie priamy operand 327 do indexregistra číslo 3.



1.4.3 Nepriame adresovanie

Adresová časť inštrukcie udáva adresu, na ktorú je zaznamenaná adresa vlastného operandu. V porovnaní s priamym adresovaním to vyžaduje prídavný cyklus pamäte, čo je nevýhodné. Uplatní sa len vtedy, ak rad inštrukcií odkazuje na rovnaký operand, ktorého umiestnenie v pamäti sa však počas spracovania programu mení.



1.4.4 Implicitné adresovánie

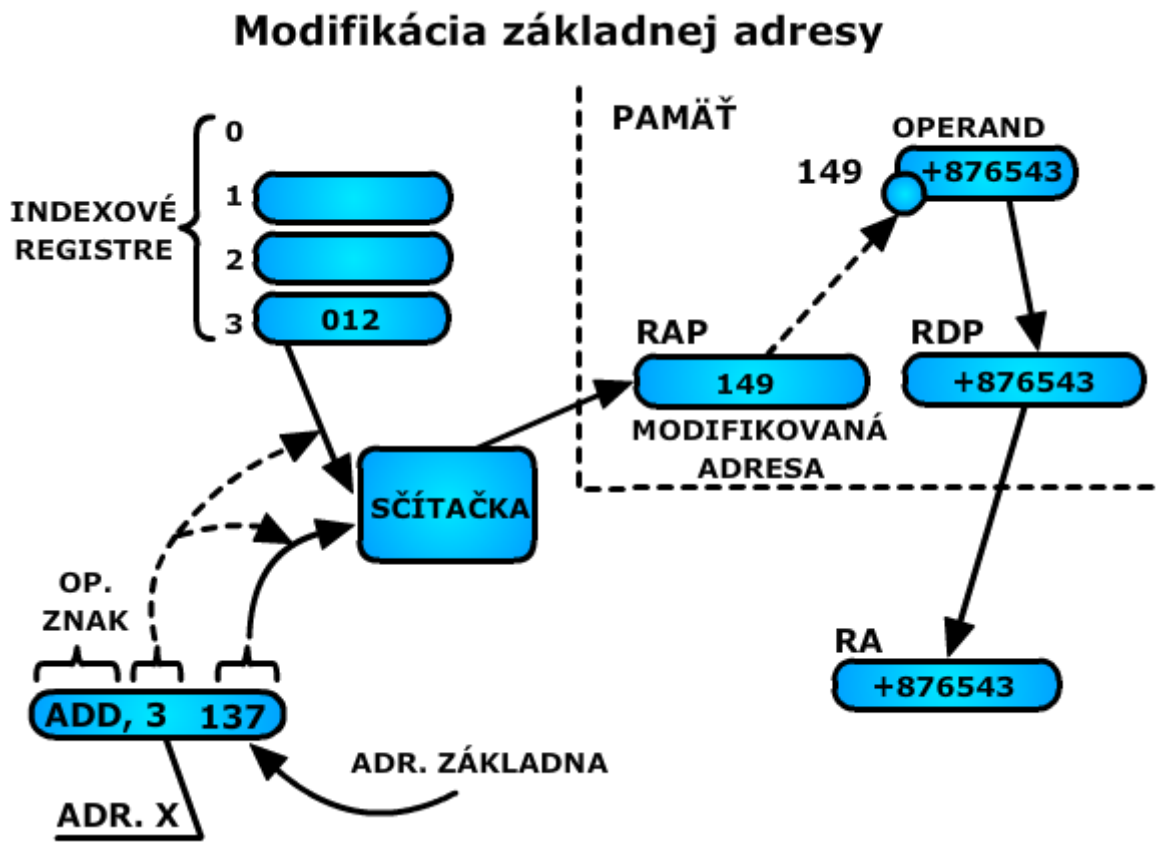
S implicitným adresovaním sa stretávame u všetkých jednoadresových počítačov, ktoré vyžadujú, aby pre všetky operácie pracujúce s dvomi operandami bolo umiestnenie jedného z nich dané vlastnou operáciou. Miesto pre výsledok je vo väčšine prípadov určené takisto implicitne. Príkladom môže byť inštrukcia prenosu medzi dvomi registrami, kedy adresy obidvoch registrov sú implicitne určené operačným kódom.

1.4.5 Implicitný operand

Ide o prípad, kedy operand je určený operačným kódom, napr. kód operácie „posuň rádovú čiarku o jeden bit“ v sebe zahŕňa operáciu násobenia dvomi (u dvojkového počítača s pevnou rádovou čiarkou).

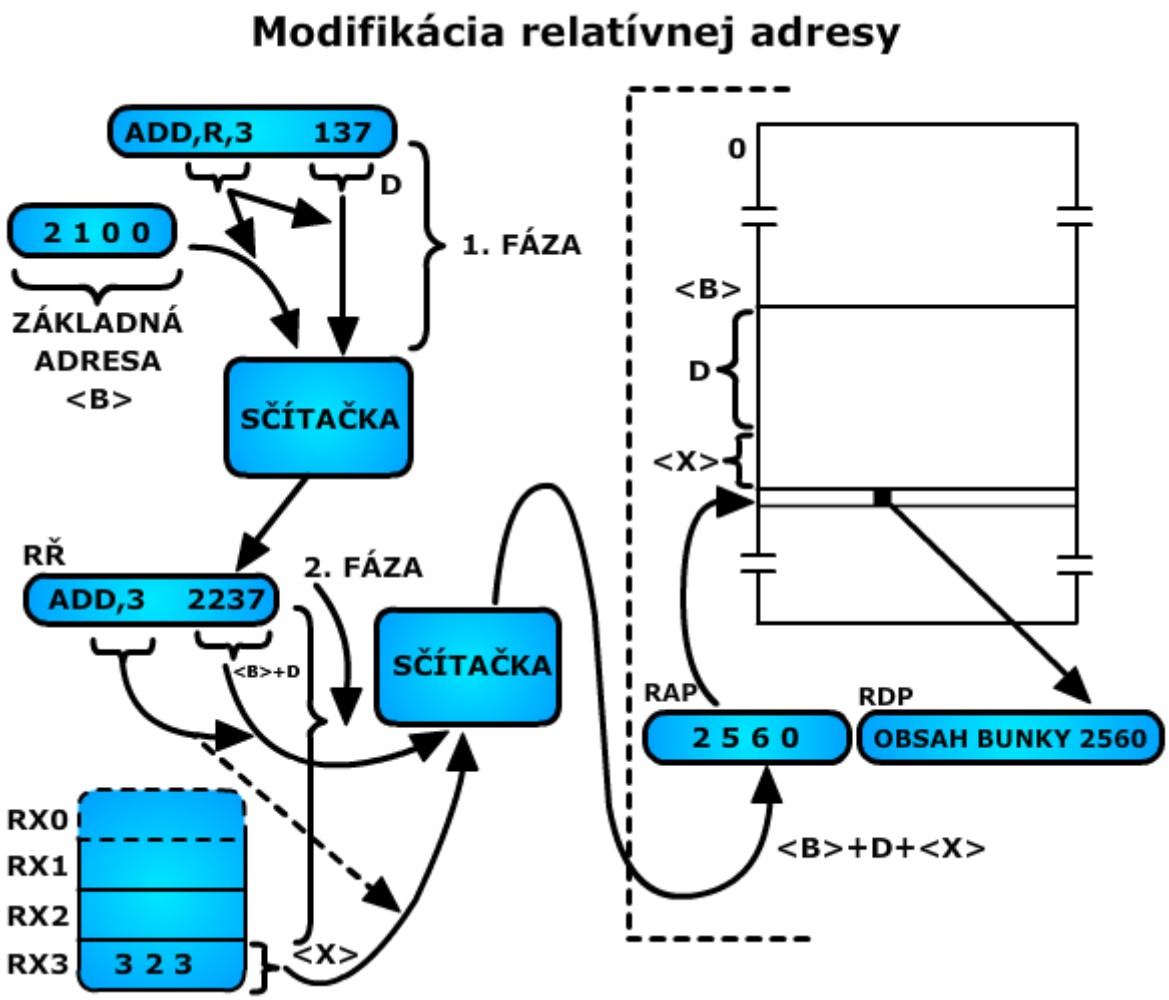
1.4.6 Modifikované adresovanie

Adresová časť inštrukcie sa pred vložením do adresového registra pamäte RAP modifikuje konštantou uloženou v indexregistri, ktorého číslo je dáné príznakom inštrukcie.



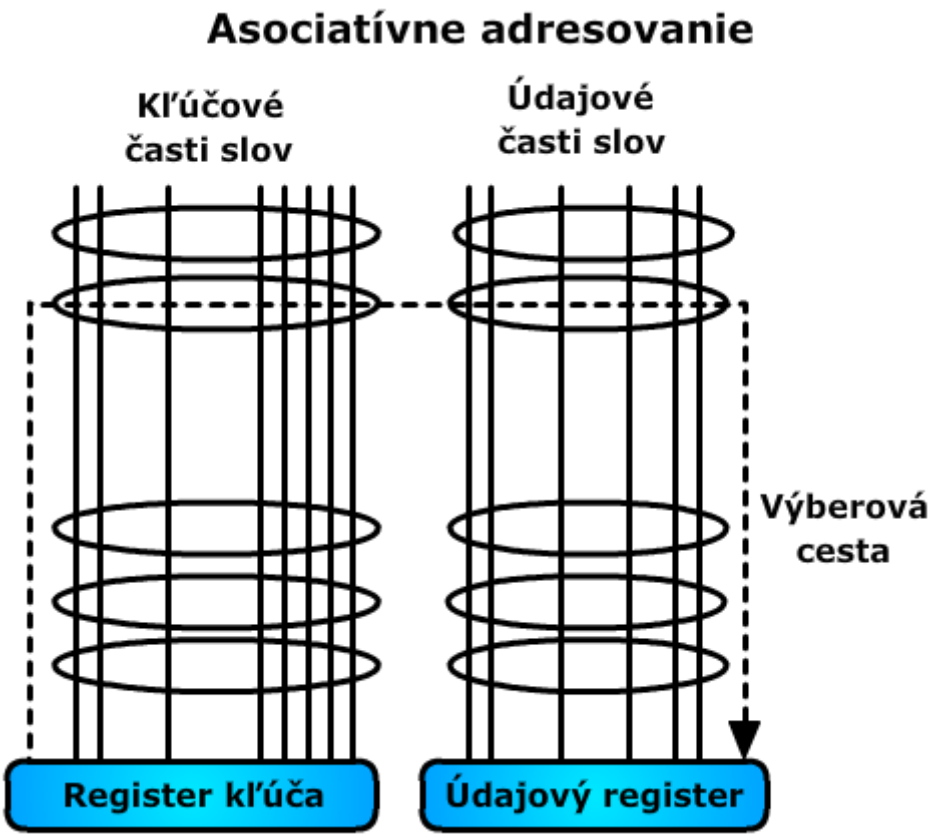
1.4.7 Relatívne adresovanie

Adresová časť inštrukcie neurčuje pri relatívnom adresovaní priamo miesto v pamäti, ale adresu uloženú k nejakej inej adrese, označenej ako uložená adresa (bázová adresa).



1.4.8 Výber podľa obsahu (asociatívny výber)

Metódy adresovania, ktoré sme prebrali, sa týkajú pamätí, ktoré sú adresované pozične, tj. každé pamäťové miesto má svoju adresu. U asociatívnych pamätí sa vyhľadáva informácia s určitou vlastnosťou. Táto vlastnosť býva určená tzv. klúčom, čo je určitá podmnožina bitov slova obsahujúceho uvedenú vzorku jedničiek a núl. Zostatkové bity sú nezaujímavé a sú pri hľadaní maskované. Teda každá bunka pamäte má údajovú časť (uložená informácia) a klúčovú časť. Obsah registra klúča sa naraz zrovná so všetkými uloženými klúčmi. V tej bunke, kde nastane zhoda klúčov, vyšle sa obsah údajovej časti bunky do údajového registra asociatívnej pamäte.

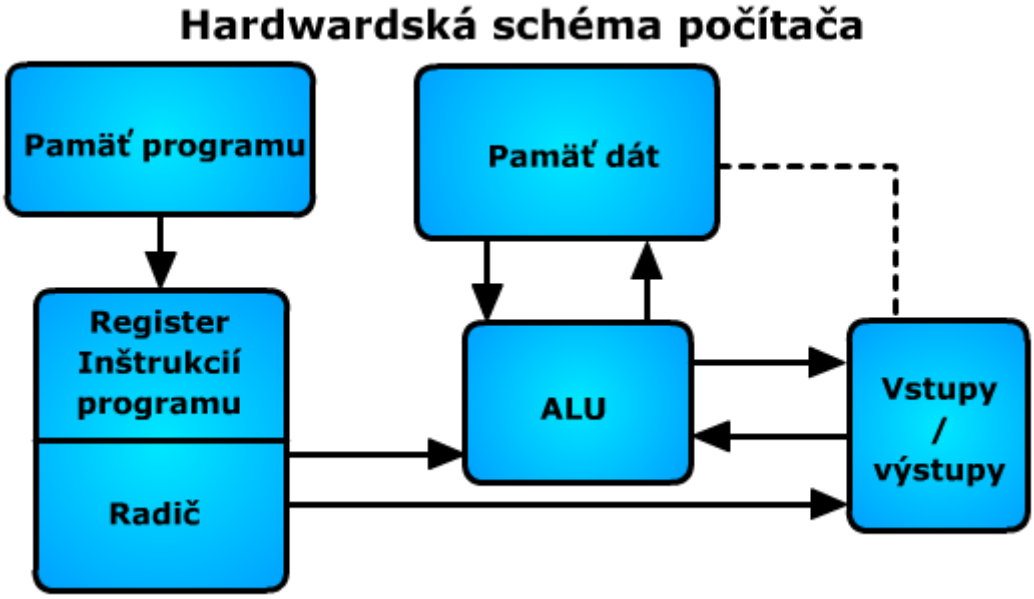


1.5 Odlišné schémy počítačov

Základné odlišnosti dnešných počítačov od von Neumannovej schémy:

- Podlľa von Neumannovej schémy počítač pracuje vždy nad jedným programom. Toto vedie k veľmi špatnému využitiu strojového času. Je teda obvyklé, že počítač spracováva paralelne viacej programov súčasne - tzv. multitasking.
- Počítač môže disponovať i viacej ako jedným procesorom.
- Počítač podľa von Neumannovej schémy pracoval len v tzv. diskrétnom režime.
- Existujú vstupno / výstupné zariadenia (I/O devices), ktoré umožňujú ako vstup, tak výstup údajov (programu).
- Program sa do pamäte nemusí zaviesť celý, ale je možné zaviesť len jeho časť a ostatné časti zavádzať až v prípade potreby.

Jednou z najpodobnejších architektúr k von Neumanovej je hardwardská schéma počítača, líšiaca sa oddelenou pamäťou pre program a pre údaje (používajú ju niektoré jednočipové mikropočítače).



V poslednej dobe sa uplatňujú viacprocesorové počítače tzn. počítače s niekoľkými CPU. Delia sa podľa toho, či majú zdieľanú pamäť na :

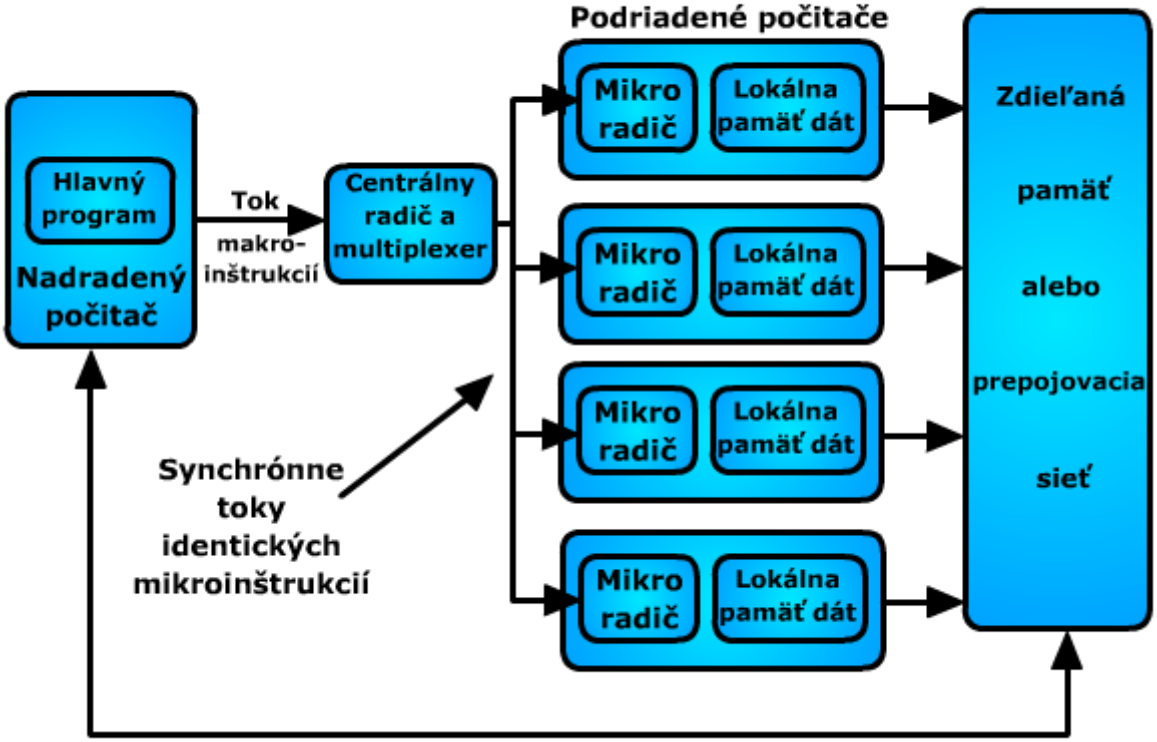
- multiprocessors (multiprocesory) majú zdieľanú pamäť,
- multicomputers (multipočítače) nemajú zdieľanú pamäť, procesory komunikujú napríklad pomocou mechanizmu zasielania správ.

Podľa počtu instrukčných a údajových prúdov sa počítače delia na:

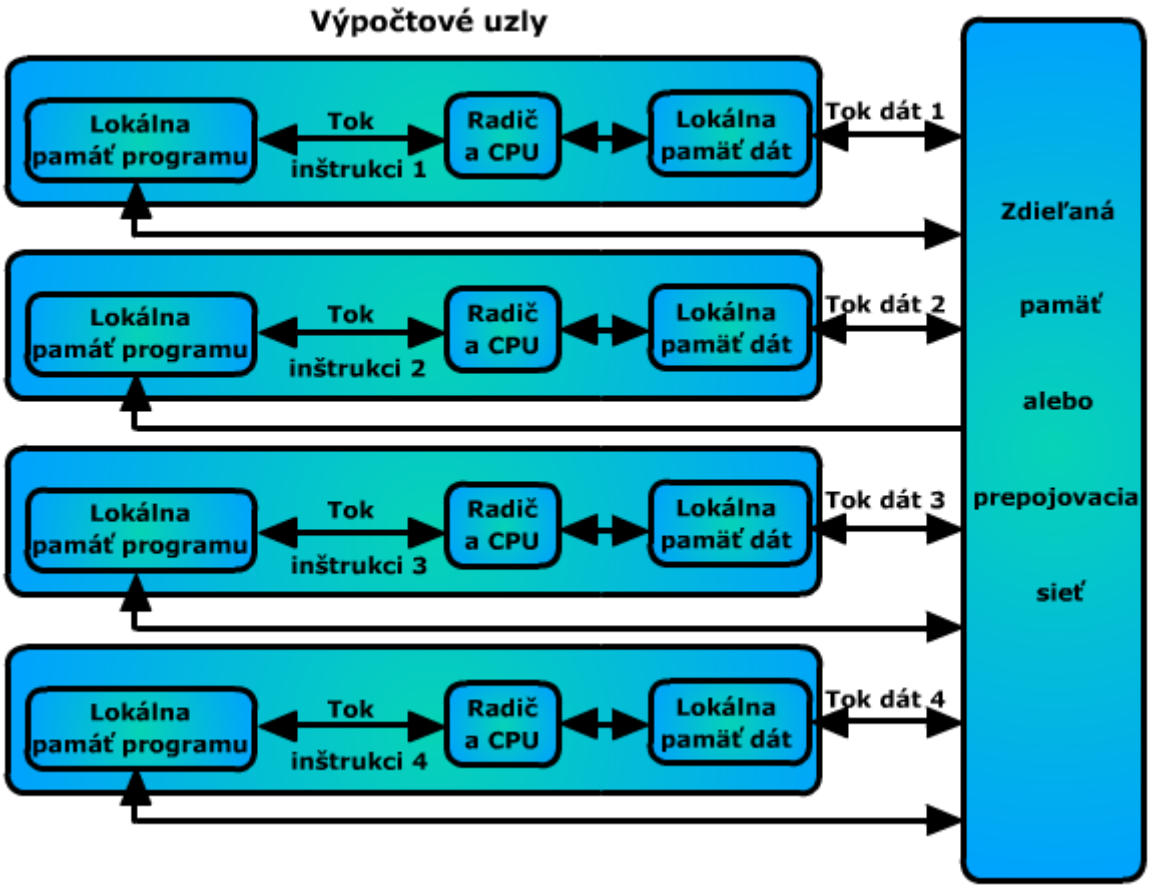
- SISD (single instruction, single data) - bežné jednoprocesorové počítače,
- SIMD (single instruction, multiple data) - jedna inštrukcia sa vykonáva na väčšom množstve údajov (napríklad sčítanie dvoch vektorov) - tzv. vektorové počítače, niektoré superpočítače sú SIMD,
- MISD (multiple instruction, single data) – niekedy označované ako zreťazené (pipeline) procesory,
- MIMD (multiple instruction, multiple data) - viacprocesorové systémy; podľa toho či majú alebo nemajú zdieľanú pamäť sa rozdelujú na multiprocesory (so zdieľanou pamäťou) a multipočítače (multicomputers - spolupracujúce počítače prepojené sieťou).

Príklady konkrétnej architektúry počítačov SIMD a MIMD so 4 výpočtovými uzlami sú naznačené na nasledujúcich obrázkoch. SIMD a MIMD sa typicky líšia v zložitosti a výkone výpočtových uzlov. Uzly v MIMD sú výkonné samostatné počítače, ale SIMD uzly sú obvykle omnoho jednoduchšie, čo je dôsledkom ich podriadenosti v systéme.

Architektúra SIMD so 4 výpočtovými uzlami



Architektúra MIMD so 4 výpočtovými uzlami



Viacprocesorové počítače sa ďalej delia na:

- Asymmetric multiprocessing - len jeden procesor môže pracovať so systémovými údajovými štruktúrami. Výhody: jednoduchší – nie je potrebné, aby OS umožňoval zdieľanie svojich vnútorných údajových štruktúr. Nevýhody: nižšia pružnosť, v niektorých prípadoch nižšia výkonnosť.
- Symmetric multiprocessing - so systémovými údajovými štruktúrami môže pracovať viacej procesorov.

- Masívny multiprocessing - desiatky procesorov.

1.6 História počítačov

Počítača sa rozdelujú do tzv. **generácií**, kde každá generácia je charakteristická svoju konfiguráciou, rýchlosťou počítača a základným stavebným prvkom. Generácie počítačov:

Generácie	Rok	Konfigurácia	Rýchlosť (operácií/s)	Súčiastky
0.	1945	Velký počet skríň	Jednotky	Relé
1.	1950	Desiatky skríň	100 - 1000	Elektrónky
2.	1958	do 10 skríň	Tisíce	Tranzistory
3.	1964	do 5 skríň	Desaťtisíce	Integrované obvody
3.$^1/_2$	1972	1 skriňa	Stotisíce	Integrované obvody (LSI)
4.	1981	1 skriňa	desiatky milionov	Integrované obvody (VLSI)

0. generácia:

Prvé počítače boli postavené na elektromagnetických relé. Ich nespoľahlivosť a veľké časové oneskorenie spôsobené mechanickou zotrvačnosťou neumožnily vytvárať spolahlivé a rýchle výpočtové systémy.

1. generácia:

Prvá generácia počítačov prichádza s objavem elektrónky, ktorej vynálezcom bol Lee De Forest a ktorá dovoľuje odstranenie pomalých a nespoľahlivých mechanických relé. Tieto počítače sú vybudované prakticky podľa von Neumannovej schémy a je pre ne charakteristický **diskrétny režim práce**. Pri tomto spracovaní je do pamäte počítača zavedený vždy jeden program a údaje, s ktorými pracuje. Potom je spustený výpočet, v ktorého priebehu už nie je možné s počítačom interaktívne komunikovať. Po skončení výpočtu musí operátor do počítača zaviesť ďalší program a jeho údaje. Diskrétny režim práce sa v budúcnosti ukazuje ako nevhodný, pretože veľmi plytvá strojovým časom. Dôvodom tohoto javu je "pomalý" operátor, ktorý zavádza do počítača spracovávané programy a údaje. V tomto okamžiku počítač nepracuje a čaká na operátora.

V tejto dobe neexistujú vyššie programovacie jazyky, z čoho vyplýva vysoká náročnosť pri vytváraní nových programov. Neexistujú ani operačné systémy.

2. generácia:

Druhá generácia počítačov nastupuje s tranzistorom, ktorého objevitelom bol John Barden a ktorý dovolil vďaka svojími vlastnosťami zmenšenie rozmerov celého počítača, zvýšenie jeho rýchlosti a spoľahlivosti a zníženie energetických nárokov počítača. Pre túto generáciu je charakteristický **dávkový režim práce**. Pri dávkovom režime práce je snaha nahradiť pomalého operátora tým, že jednotlivé programy a údaje, ktoré sa budú spracovávať, sú umiestnené do tzv. dávky a celá táto dávka je daná počítaču na spracovanie. Počítač po skončení jedného programu okamžite z dávky zavádza ďalší program a pokračuje v práci.

V tejto generácii počítačov taktiež začínajú vznikať operačné systémy a prvé programovacie jazyky, ako sú COBOL a FORTRAN.

3. generácia:

Počítače tretej a vyšších generácií sú vybudované na integrovaných obvodoch, ktoré na svojich čipoch integrujú veľké množstvo tranzistorov. U tejto generácie sa začína objavovať paralelné spracovanie viacej programov, ktoré má opäť za úlohu zvýšiť využitie strojového času počítača. Je totiž charakteristické, že jeden program pri svojej práci buď intenzívne využíva CPU (vykonáva zložitý výpočet), alebo napr. skôr využíva V/V zariadenia (zavádza údaje do operačnej pamäte, príp. vykonáva tlač výstupných údajov). Takéto programy potom môžu pracovať na počítači spoločne, čím sa lepšie využije kapacita počítača.

S postupným vývojom integrovaných obvodov sa neustále zvyšuje stupeň integrácie (počet integrovaných členov na čipe integrovaného obvodu). Podľa počtu takto integrovaných súčiastok je možné rozlíšiť následujúce stupne integrácie:

Označenie	Anglický názov	Slovenský názov	Počet logických členov
SSI	Small Scale Integration	Malá integrácia	10
MSI	Middle Sclae Integration	Stredná integrácia	10 - 100
LSI	Large Scale Integration	Vysoká integrácia	1000 - 10000
VLSI	Very Large Scale Integration	Veľmi vysoká integrácia	10000 a viac

Integrované obvody je možné vyrábať pomocou rôznych technológií, z ktorých každá má svoj základný stavebný prvok a vďaka nemu poskytuje špecifické vlastnosti:

- **TTL** (Transistor Transistor Logic): rýchla, ale drahá technológia. Jej základným stavebným prvkom je bipolárny tranzistor. Jej nevýhodou je velká spotreba elektrickej energie a z toho vyplývajúce velké zahrievanie sa takýchto obvodov.
- **PMOS** (Positive Metal Oxid Semiconductor): technológia používajúca unipolárny tranzistor MOS s pozitívnym vodivostným kanálom. Vďaka tomu, že MOS tranzistory sú riadené elektrickým poľom a nie elektrickým prúdom ako u technológie TTL, redukuje nároky na spotrebu elektrickej energie. Jedná sa však o pomalú a dnes nepoužívanú technológiu.
- **NMOS** (Negative Metal Oxid Semiconductor): technológia, ktorá využívá ako základný stavebný prvok unipolárny tranzistor MOS s negatívnym vodivostným kanálom. Táto technológia sa používala zhruba do začiatku 80. rokov. Jedná sa o lacnejšiu a efektívnejšiu technológiu ako TTL a rýchlejšiu ako PMOS.
- **CMOS** (Complementary Metal Oxid Semiconductor): technológia spojujúca v jednom návrhu prvky tranzistorov PMOS i NMOS. Tieto obvody majú malú spotrebu a táto technológia je používaná pre výrobu veľkej časti dnešných moderných integrovaných obvodov.
- **BiCMOS** (Bipolar Positive Metal Oxid Semiconductor): nová technológia spájajúca na jednom čipe prvky bipolárnej technológie i technológie CMOS. Používaná je hlavne firmou Intel k výrobe mikroprocesorov.

1.7 Kategórie počítačov

Ďalší typ klasifikácie počítačov vychádza z výkonu počítačov a z ich aplikačného nasadenia. Definujme si tieto štyri kategórie počítačov:

- **Mikropočítače.**
- **Minipočítače.**
- **Strediskové počítače.**
- **Superpočítače.**

Všetky tieto kategórie existujú v jednom čase (súčasnej dobe) vedľa seba. Mikropočítače sú určené pre každodenné používanie jednému používatelovi. Niekedy sa táto kategória nazývá tiež "osobné počítače". Mikropočítače existujú vďaka mikroprocesorom, ktorých nízká cena dovoľuje široké použitie. Minipočítač zdieľa väčšinou viacej používatelov prostredníctvom viacej terminálov, alebo slúži ako komunikačný uzol počítačovej siete apod. Strediskový počítač (mainframe) svojím vysokým výkonom slúži k vedeckotechnickým výpočtom a veľkým počtom V/V zariadení pre spracovanie hromadných údajov. Typickými aplikáciami superpočítačov sú armáda, meteorológia, seizmológia, naftový priemysel, atómová fyzika apod.