(Mikro)Procesor

CPU – Central Process Unit

(spravidla realizovaný mikroprocesorom)

CP – Central Processor

(označenie v superpočítačoch)

Procesor - časť počítača určená na vykonávanie programu. Je to logický obvod, ktorý dokáže spracovávať sadu **mikroinštrukcií** (veľmi jednoduché príkazy).
Program vytvorený z mikroinštrukcií tvorí sadu inštrukcií používaných programátorom.

Inštrukcia – z hľadiska programátora najmenší elementárny úkon vykonateľný procesorom.

Kód inštrukcie – binárna reprezentácia inštrukcie uložená v pamäti.
Skladá sa obyčajne z :
- **kódu operácie**
- a kódu **operandu** (informácia, kde sa nachádza operand)

Operand – pamäťové miesto, ktorého obsah sa zúčastňuje operácie (inštrukcie bez operandu, s jedným a dvomi operandami)

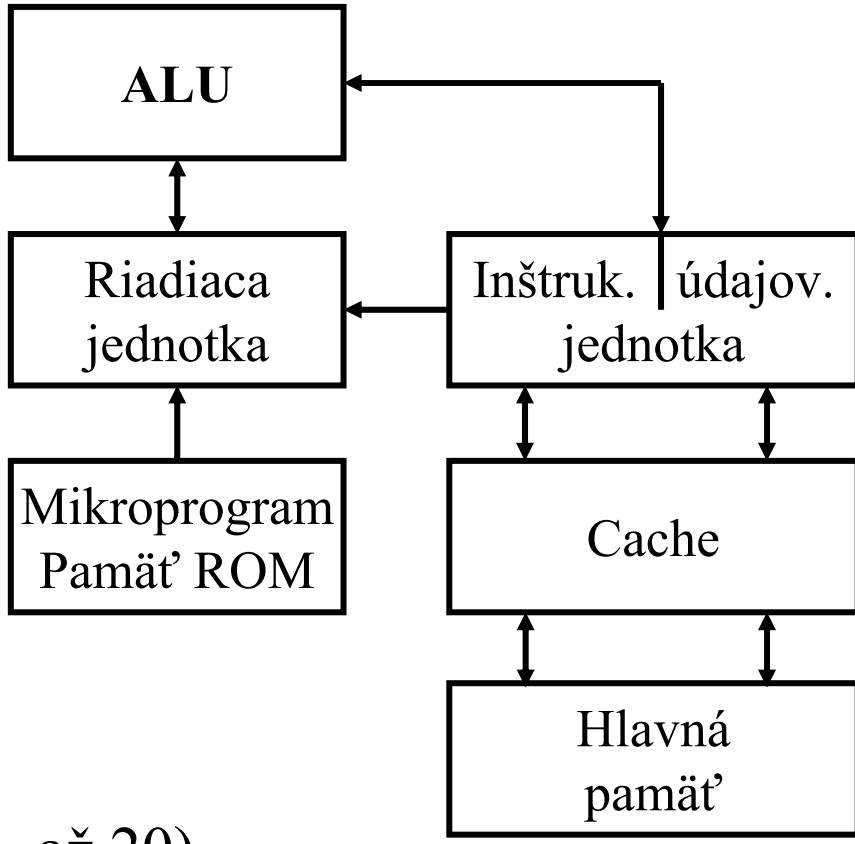
Program – postupnosť inštrukcií uložených v určitom poradí v pamäti

Poznáme dve základné koncepcie mikroprocesorov:

Complete Instruction Set Computer (CISC) Procesor má „úplnú“ sadu inštrukcií. Cieľom vývoja týchto procesorov bol stav, kedy jazyk vysokej úrovne bude priamo strojovým jazykom daného procesora. Procesor CISC => jednoduchý návrh prekladačov.
CISC procesory na konci 70 rokov boli už tak zložité, že sa objavili prvé pokusy o celkové zjednodušenie štruktúry procesorov. Vznikla tak celá nová kategória počítačov, dnes označovaná ako RISC.

Reduced Instruction Set Computer (RISC) Vychádza z predpokladu, že na vykonanie cca 80% operácií programu postačuje cca 20 % inštrukcií. Procesor má zabudované len základné mikroinštrukcie, ktoré sú jednoduchšie a ľahšie – rýchlejšie vykonateľné. Inštrukcie, ktoré nie sú v základnom inštrukčnom súbore sa jednoducho naprogramujú. Výhoda procesorov RISC, sú rýchlejšie ako CISC, paradoxne zvyšuje cenu počítača. RISC procesor predpokladá rýchlejšie zbernice, pamäte, atď.

Procesory CISC:



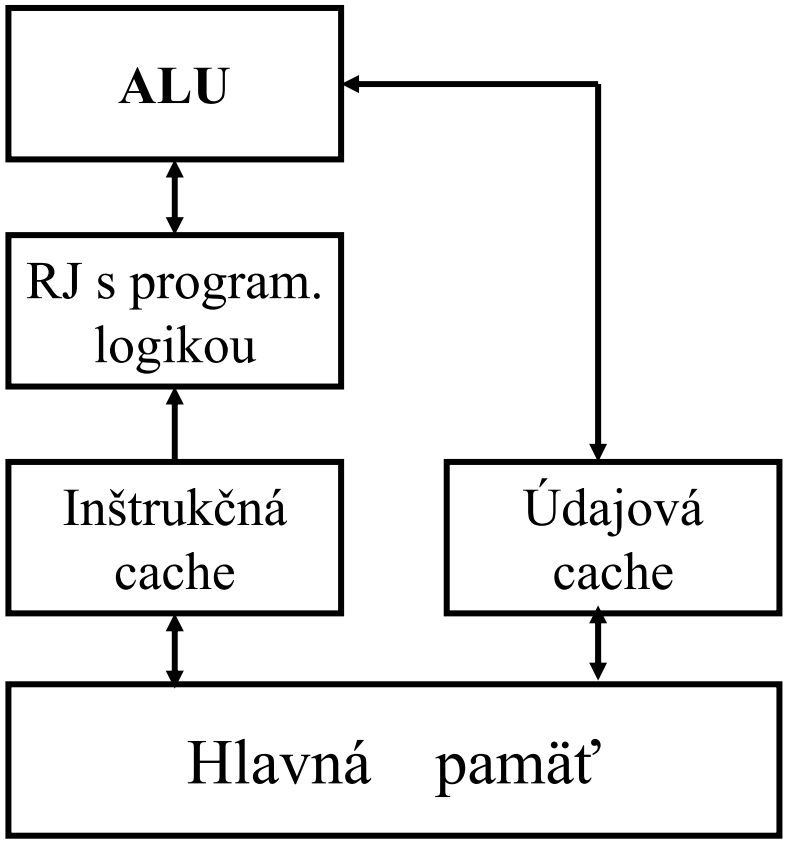
Charakteristika:

- veľa zložitých inštrukcií (120 – 350)
- malý počet vnútorných registrov
- inštrukcie s premenlivou dĺžkou
- veľký počet formátov inštrukcií (> 8)
- zložité, časovo náročné dekódovanie
- veľa adresovacích módov (>12)
- **riadenie vykonávania inštrukcie pomocou mikroprogramu**
- vykonanie inštrukcie na viacero SC (2 až 20)
- možnosť doplnenia užívateľských inštrukcií v pamäti mikroprogramu

Možnosti zvyšovania výkonnosti:

- prúdové spracovanie operandov
- prúdové pracovanie inštrukcií - predvýber inštrukcie (IF)

Procesory RISC:



Celkový čas vykonávania programu T_c možno vyjadriť vzťahom:

$$T_C = T_{SC} \sum_{i=1}^{N} C_i$$

Kde:

N – počet inštrukcií

$\boldsymbol{T_{SC}}$ – čas trvania jedného SC

C_i – „čas“ trvania i-tej inštrukcie [/]

RISC svojou podstatou zvyšuje N

C_i a T_{SC} sú pre procesory RISC kratšie.

Požiadavka je $C_i < 1$

Charakteristika:

- redukovaný súbor inštrukcií (cca 100 a menej)
- inštrukcie s konštantnou dĺžkou
- malý počet formátov (menej ako 8) jednoduchšie dekódovanie
- malý počet adresovacích módov (3 až 5), ale pre čítanie a zápis do pamäte len dve inštrukcie (Load, Store), ostatné sú registrové
- **riadenie vykonávania inštrukcie pomocou pevnej logiky**
- vykonanie inštrukcie spravidla v 1 SC
- zložitejší kompilátor z vyšších programovacích jazykov
- veľký počet vnútorných registrov (zmenšuje potrebu zapisovať a čítať medzivýsledky z pamäte), umožňuje odovzdávanie parametrov do podprogramov logickým usporiadaním registrov do „okien“

Argumenty za CISC:

- komplexnejšie inštrukcie, zmenšujú počet čítaní inštrukcie z pamäte
- jednoduchší kompilátor z vyšších programovacích jazykov
- obchodný a reklamný faktor (procesor vykonávajúci viacej typov inštrukcií = dokonalejší)
- vygenerovaný kód programu je kratší

Argumenty za RISC:

- na čipe procesora zbytočne zaberá miesto pamäť mikroprogramu s príslušným zložitým riadením
- vysoké percento inštrukcií nevie využiť kompilátor ani programátor
- dynamické merania IBM ukázali, že
    - 50% času sa vykonávajú len 3 inštrukcie !!
    - 75% času 8 inštrukcií
    - inštrukcie, ktoré bežia 0,2% času zaberajú 60% pamäte mikroprogramu !!
- čas vývoja CISC až 5 rokov
- čas vývoja RISC 8- 20 mesiacov (uvedený RISC- prvý vyrobený čip fungoval)!!!!
- prúdové spracovanie inštrukcií (optimalizácia skokov, vyplňovanie bublín, odovzdávanie výsledkov ...)
- efektívny čas vykonania inštrukcie pri špičkových menej ako 1 hodinový cyklus
- technologický a vývojový faktor

Základná funkcia procesora je založená na opakovaní základných činností :

a) prečítanie inštrukcie (adresa inštrukcie je uložená v registri CPU v **programovom počítadle PC**), ktoré pozostáva z
   - z prečítania kódu operácie do **inštrukčného registra IR**
   - dekódovania kódu operácie, či k danému kódu netreba čítať ďalšie časti inštrukcie (operand), dokončiť čítanie inštrukcie

b) **riadiaca jednotka CPU** na základe kódu zabezpečí postupnosť **riadiacich signálov** tak, aby bola vykonaná požadovaná operácia. Výsledok operácie sa zvyčajne dočasne ukladá do vnútornej pamäte CPU – registrov CPU

c) po vykonaní operácie resp. pred jej vykonaním riadiaca jednotka pripočíta k obsahu PC hodnotu, ktorá je rovná dĺžke inštrukcie v bytoch => po vykonaní inštrukcie bude v PC **adresa nasledujúcej inštrukcie**, ktorá sa má vykonať, a činnosť prebieha znovu od bodu a).

Operácie, ktoré CPU je schopné vykonať (tvoria **inštrukčný súbor** procesora) môžeme rozdeliť na :

1. **operácie na presun údajov** (kopírovanie obsahu pamäťových miest registrov, pamäťových buniek a registrov I/O zariadení)

2. **operácie aritmeticko-logické** (zmena obsahu pamäťovej bunky)

3. **operácie riadenia toku programu** (skokové inštrukcie, inštrukcie vetvenia programu, volanie a návrat z podprogramu) – zmena obsahu programového počítadla

4. Výrobcovia procesorov doplňujú inštrukčnú sadu o inštrukcie pre prehrávanie videa, zvuku, grafiky.

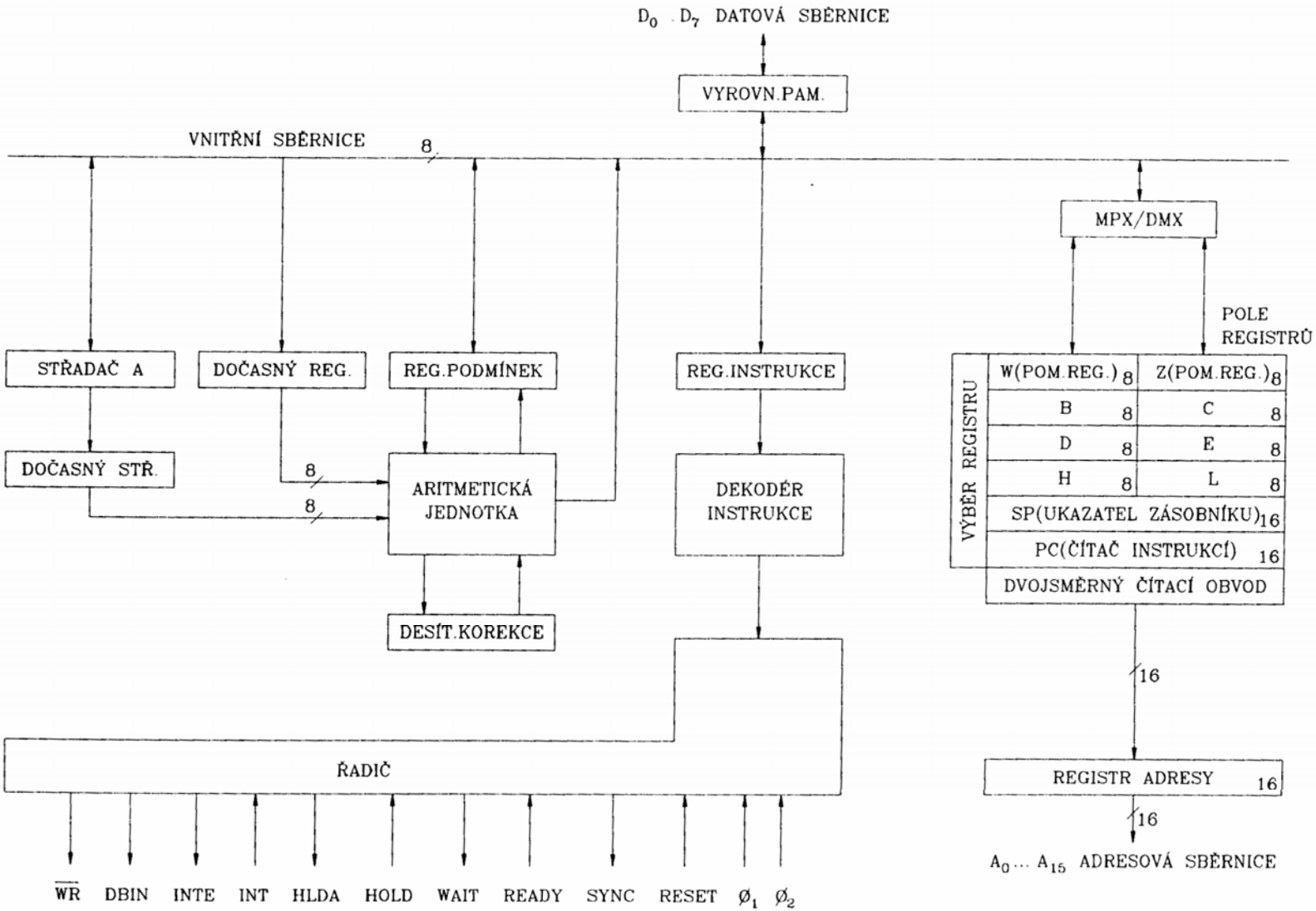
Na to aby CPU mohlo vykonávať program musí mať:

- **registre** na dočasné uchovávanie informácií (väčšina z nich je prístupná programátorovi) (**akumulátor**, **registre pre všeobecné použitie, stavový register**)
- **programové počítadlo,**
- **inštrukčný register,**
- **smerník zásobníka**
- **aritmeticko-logickú jednotku**
- **riadiacu jednotku**
- **jednotku na zápis a čítanie** informácie v pamäti a I/O priestore

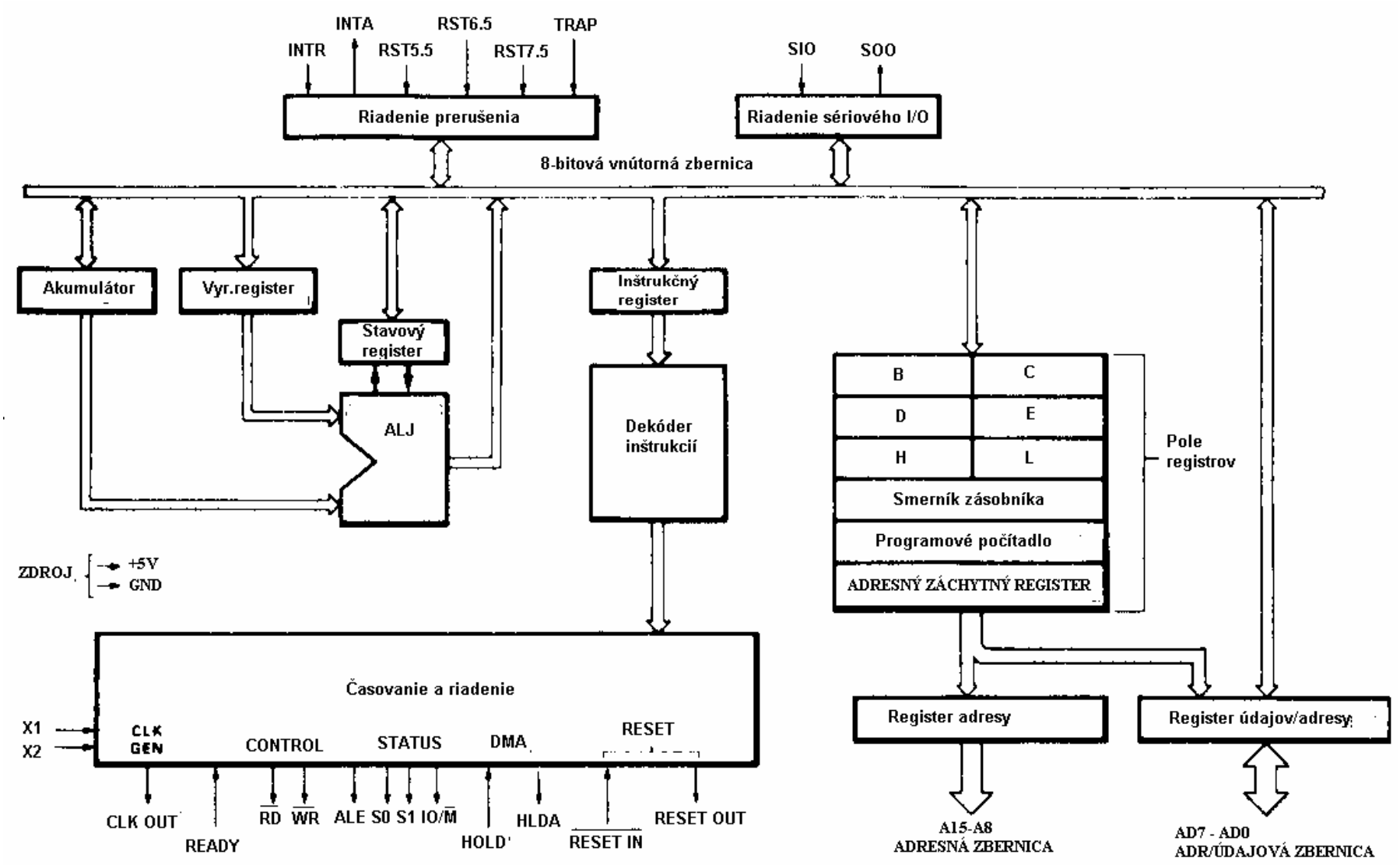
Okrem týchto základných funkcií CPU by malo vedieť:

- dostať sa do definovaného stavu (reagovať na signál **RESET**)
- dostať sa do stavu, kedy nebude generovať signály na zbernicu (**HOLD**), tento stav je dôležitý pre realizácii **DMA** a pre súčinnosť s viacerými procesormi
- **vetviť program** (vykonávať inštrukcie mimo poradia určených programom) na základe **vonkajších signálov** (maskovateľné a nemaskovateľné prerušenia)
- signalizovať svoj **stav** na zbernicu
- vedieť prispôsobiť sa pomalším zariadeniam na zbernici a pod.

Ako ilustrácia 8-bitový procesor I8080.



Ako ilustrácia 8-bitový procesor I8085.



Architektúra mikroprocesorov:
Sa rozumie schopnosť mikroprocesora spracovať postupnosť inštrukcií. Staršie procesory (8086 až 486) spracovávali inštrukcie programu postupne jednu po druhej.

Novšie procesory (Pentia a ďalej) sú vybavené **superskalárnou architektúrou.** Podstata je v súčasnom spracovávaní viacerých inštrukcií naraz. Realizuje sa to napr. zdvojením niektorých funkčných blokov procesora.

Ďalšie zrýchlenie práce procesorov sa dá dosiahnuť pomocou **pipeling** , zreťazenia vykonávania:
- aritmetických operácií, v ALU.
- inštrukcií, v inštrukčnej jednotke. Problémom tohto riešenia je vetvenie programu.

Prúdové spracovanie informácie:

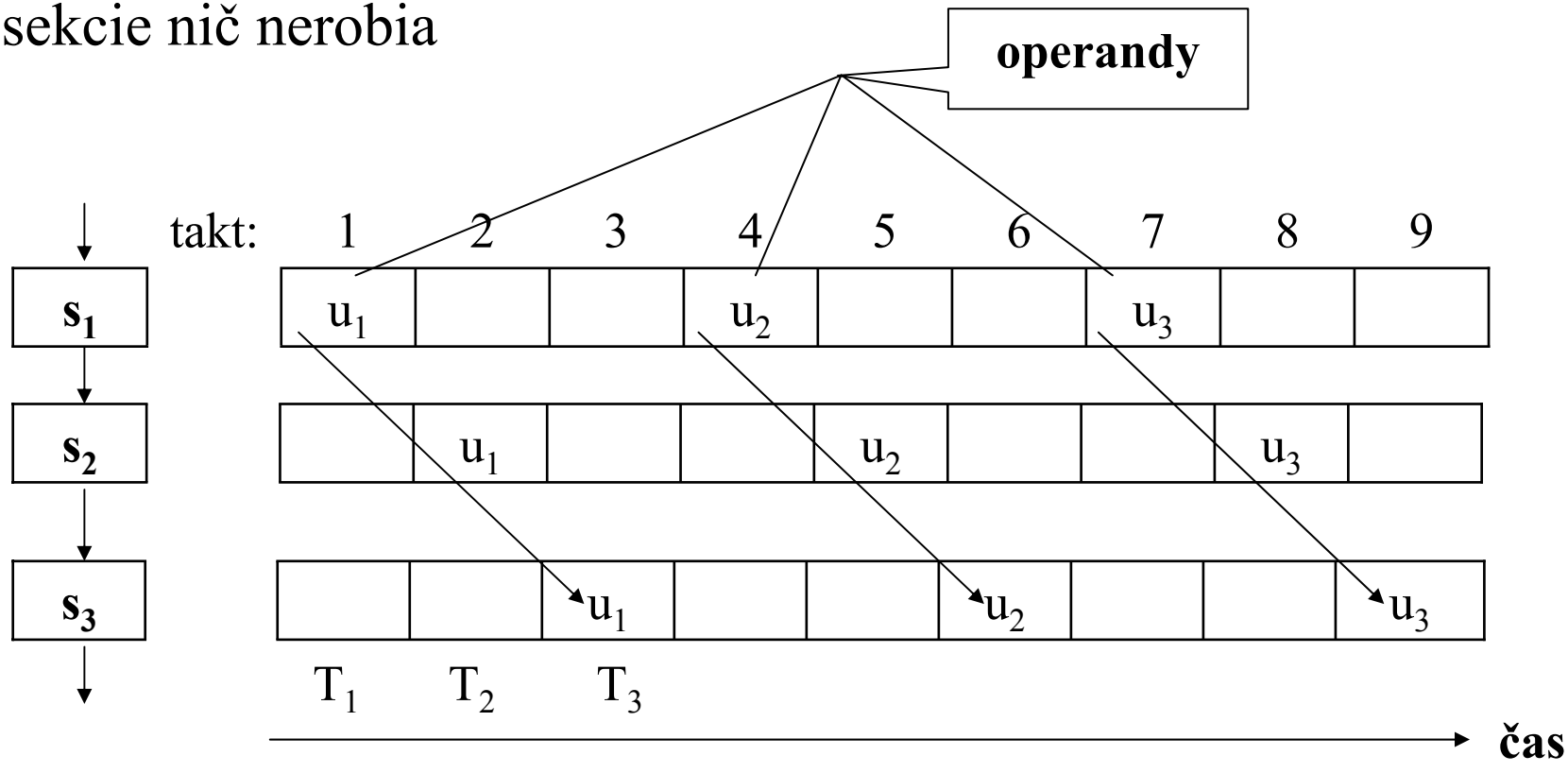
Operand (operandy) spracovania postupne prechádza transformáciami

Spracovanie v k-funkčných blokoch:

- pri postupnom spracovávaní (k-1) blokov nič nerobí

Pr.: uvažujme 3 funkčné bloky (sekcie)

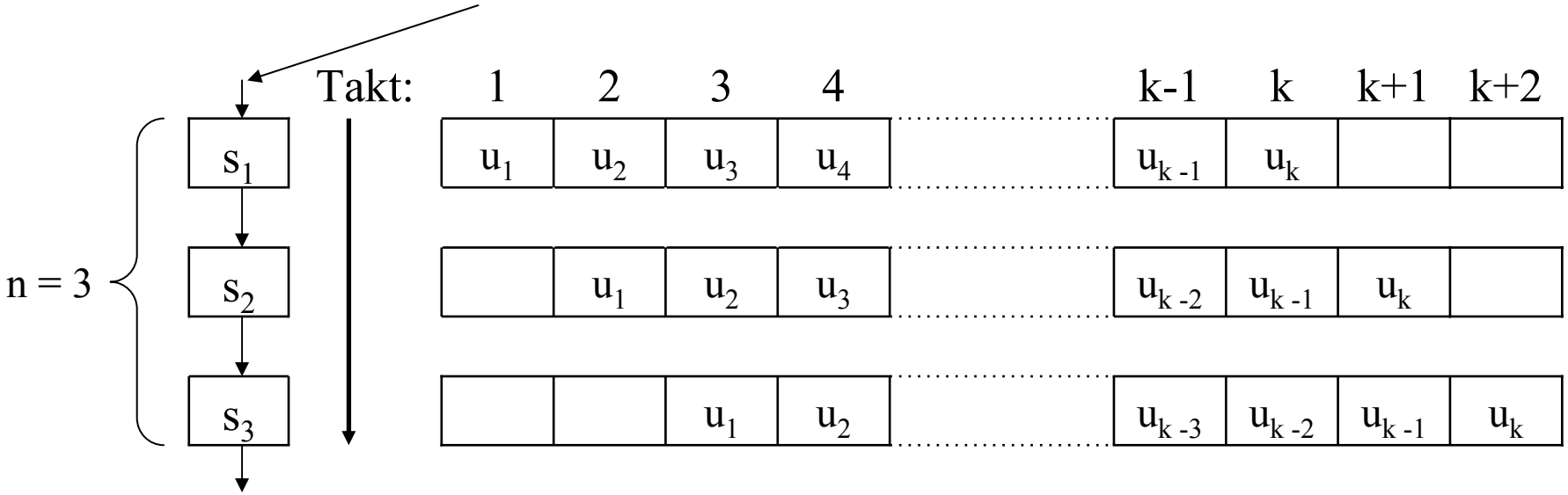
- u_i sú jednotlivé operácie (úlohy)
- kým nie je dokončená predchádzajúca úloha, nezačne nasledujúca
- vždy 2 sekcie nič nerobia



Čas vykonania k úloh (nech T_1 trvá v s_1, T_2 trvá v s_2 a T_3 trvá v s_3), bude

$$T_C = k.(T_1 + T_2 + T_3)$$

Prúdové spracovanie operandov (pipelining)



Čas vykonania *k* úloh bude v tomto prípade $T_C = (k+2)T_{\max}$

kde $T_{\max} = \max\{T_1, T_2, T_3\} + \Delta T$

ΔT je čas potrebný na zápis a čítanie výsledku mikroopeácie z vyrovnávacieho registra.

Napr.: $T_2 > T_1 = T_3$

Pri určitom zjednodušení môže byť koeficient priepustnosti prúdovej jednotky oproti sekvenčne pracujúcej jednotke

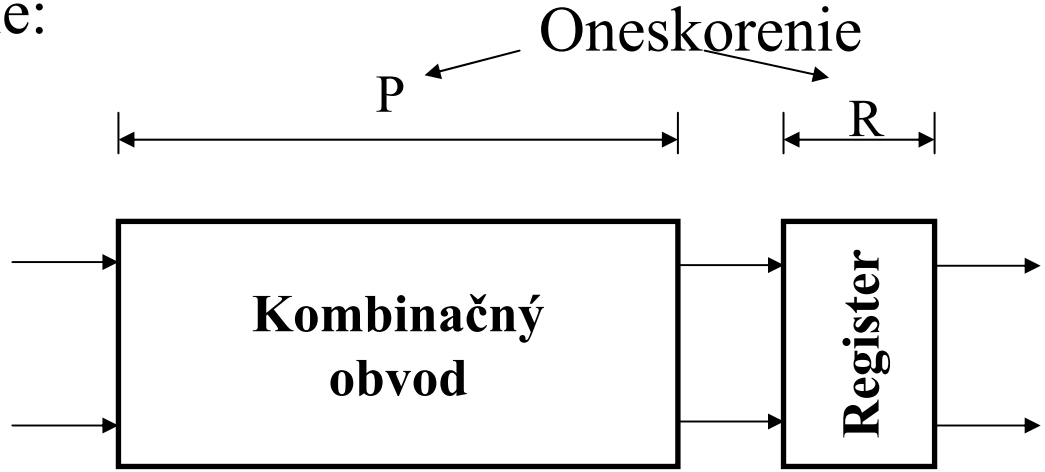
$$q = \frac{n.k}{n+k-1} \quad pre \quad k >> n \quad q \cong n$$

kde n je počet sekcií

Podmienky realizácie prúdového spracovania :

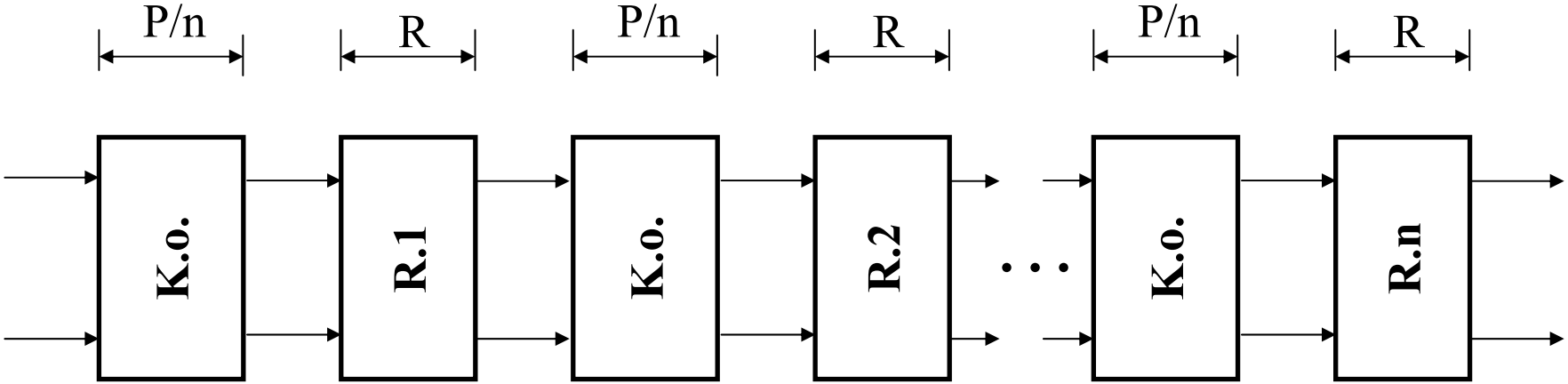
- činnosť sekcií musí byť **synchronizovaná**
- každá sekcia musí obsahovať **vyrovnávací register**, cez ktorý odovzdáva medzivýsledok ďalšej sekcii
- **čas vykonania** mikrooperácie v sekciách je **rovnaký** (čas vykonania mikrooperácie sa rovná času vykonania mikrooperácie v najpomalšej sekcii)

Realizácia jednej sekcie:



Za čas P + R spracuje jednu úlohu.

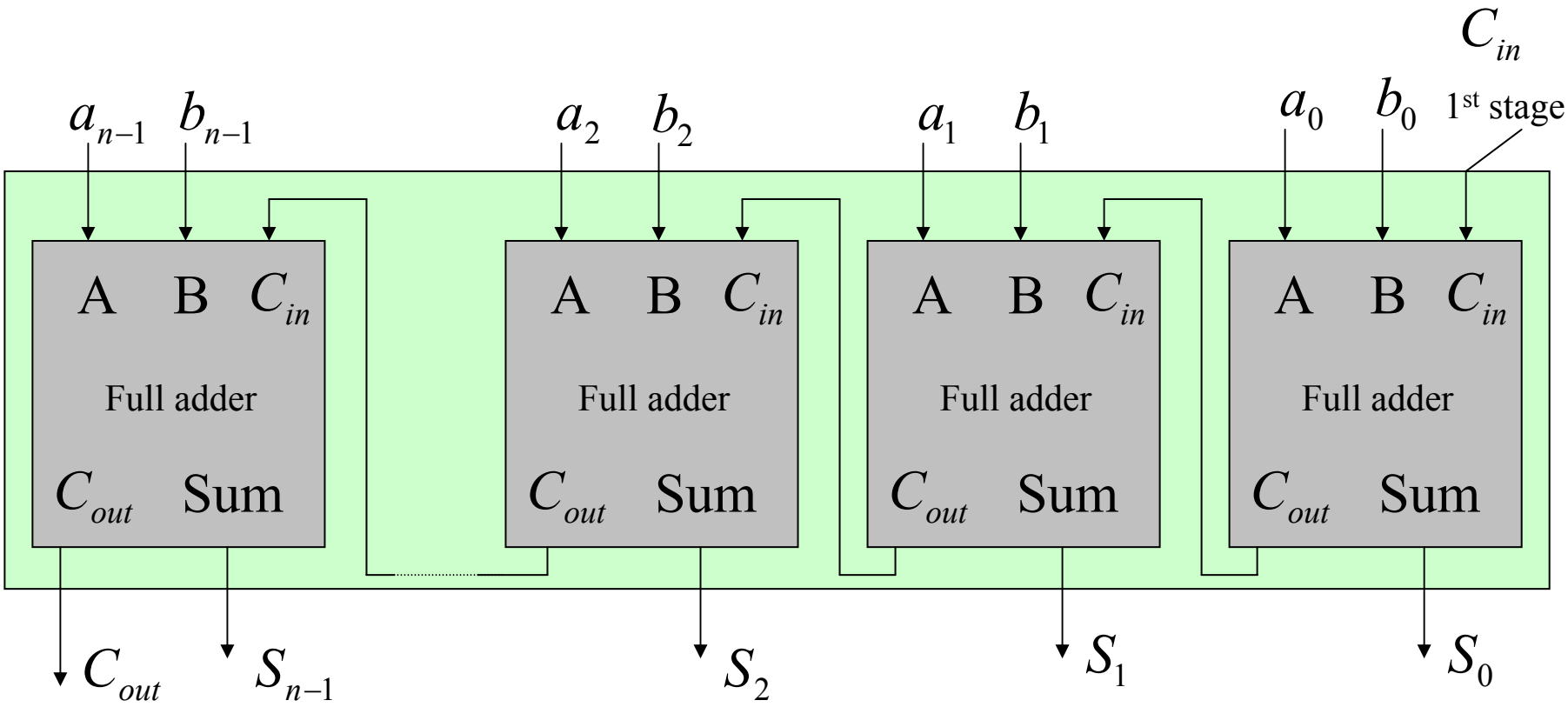
Realizácia obvodu s *n* sekciami:



Existuje optimálna (rozumná) voľba počtu sekcií ? Áno. *Zložitý vzťah.*

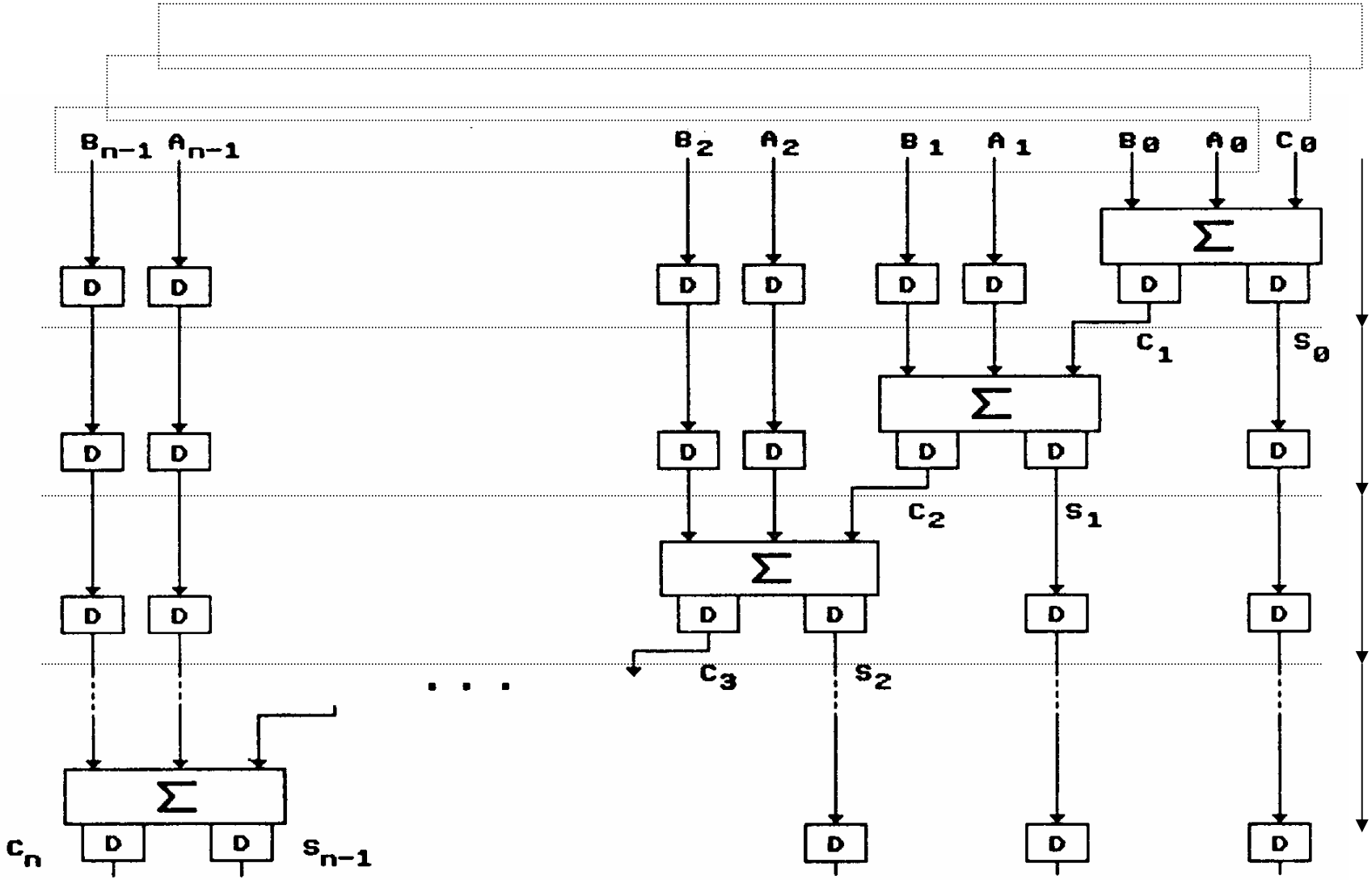
Na sčítanie dvoch n-bitových čísel sa dajú použiť **paralelná** alebo **sériová** sčítačka, pričom sa vo všetkých stupňoch používa úplná sčítačka (realizácia prenosu z predchádzajúceho sčítania)

Inkrementácia: nastavíme carry bit a pričítame nuly



Paralelná sčítačka

Príklad : prúdovo pracujúca sčítačka



Prúdové spracovanie inštrukcií :

Inštrukcia (strojová inštrukcia) je najmenší programový element vykonateľný procesorom.
Inštrukcia sa delí na strojové cykly, samostatné elementárne úkony vykonateľné jednotkami procesora, z ktorých každá trvá násobky hodinových cyklov.
Každú inštrukciu je možné formálne rozdeliť na dielčie elementárne úkony.

Elementárne úkony pri vykonaní inštrukcie :

Typ inštrukcie: aritmetická alebo logická

Operandy dva: - 1. Register procesora

- 2. Hlavná pamäť

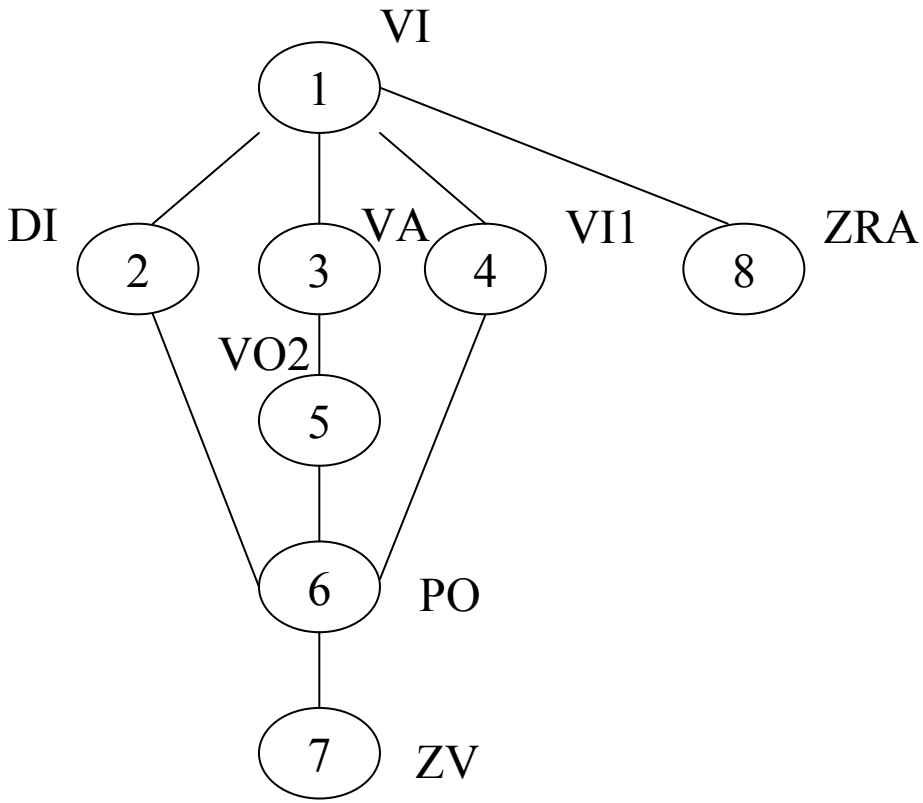
1. **V**ýber kódu **I**nštrukcie z pamäte (VI)
2. **D**ekódovanie **I**nštrukcie (DI)
3. **V**ýpočet reálnej **A**dresy operandu inštrukcie (operand 2) (VA)
4. **V**ýber **O**perandu z registra (**1**. operand) (VO1)
5. **V**ýber **O**perandu z pamäte (**2**. operand) (VO2)
6. **V**ykonanie **O**perácie (P**O**)
7. **Z**ápis **V**ýsledku (ZV)
8. **Z**väčšenie obsahu inštrukčného registra – **R**egistra **A**dries(ZRA)

Čas

Niektoré z úkonov sú navzájom nezávislé (vykonateľné paralelne), potom je možné usporiadať mikrooperácie nasledovne:

V novom usporiadaní bude 5 fáz

1. Výber kódu inštrukcie z pamäte (VI)
2. Dekódovanie inštrukcie (DI) (predtým DI, VA, VO1, ZRA)
3. Výber operandu z pamäte (2. operand) (VO) (predtým VO2)
4. Vykonanie operácie (PO)
5. Zápis výsledku (ZV)



Postupné vykonávanie inštrukcií:

1. instr. | VI | DI | VO | PO | ZV |
2. instr. | VI | DI | VO | PO | ZV |
3. instr. | VI | DI | VO | PO | ZV |

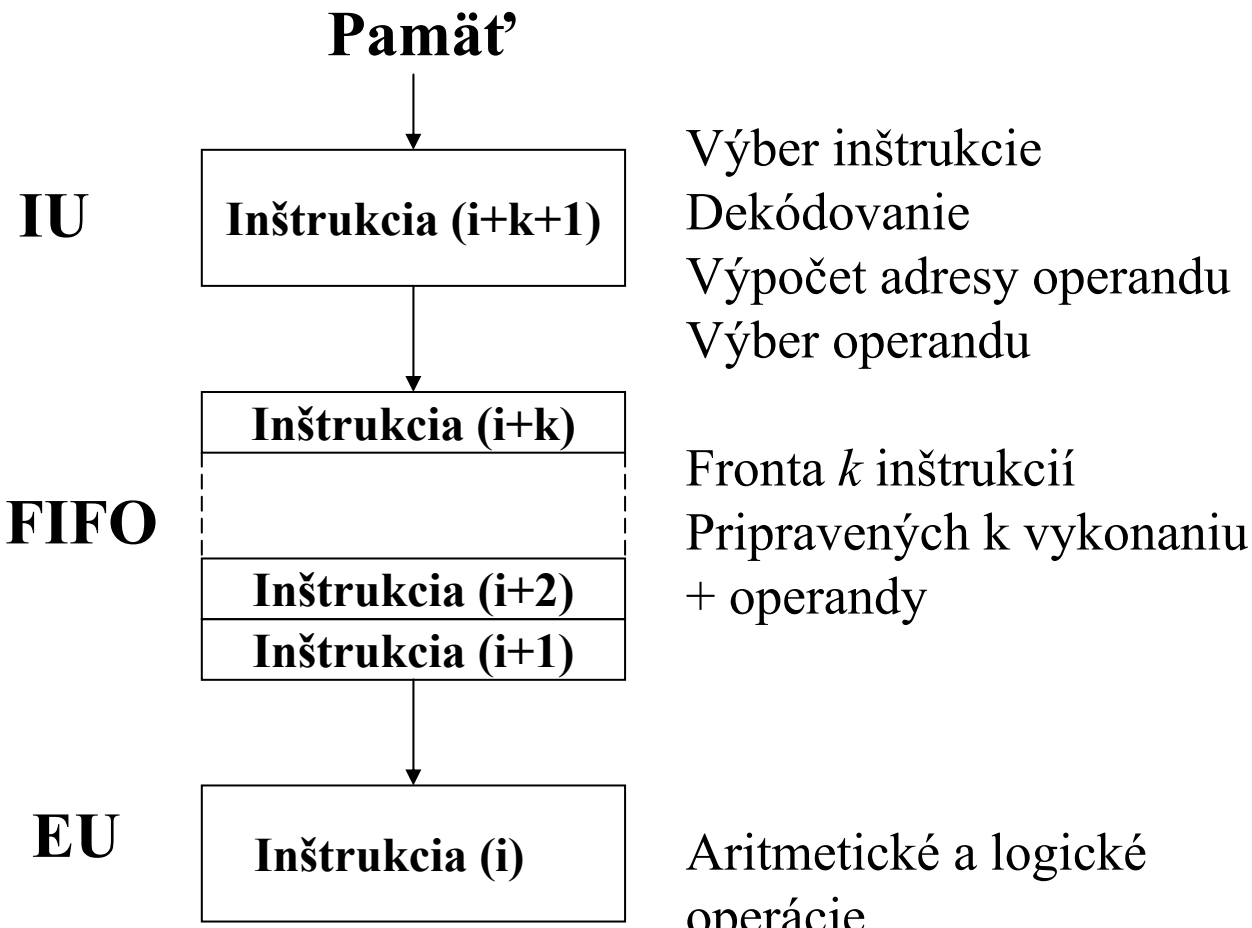
Vykonávanie inštrukcií s prekrytím (predvýber inštrukcie)

1. instr. | VI | DI | VO | PO | ZV |
2. instr. | VI | DI | VO | PO | ZV |
3. instr. | VI | DI | VO | PO | ZV |

Formálne sa rozdelí vykonanie inštrukcie
- na predvýber (**FETCH**) (VI alebo VI+DI+VO) a
- vykonanie inštrukcie (**EXECUTE**) (DI+VO+PO+ZV alebo PO+ZV)

Príklad organizácie CPU (I8086,..) na sekcie

- **predvýber** inštrukcie (výber inštrukcie, dekódovanie kódu inštrukcie, výpočet reálnej adresy operandu, výber operandu z pamäte)
- uloženie dekódovaných inštrukcií do **inštrukčnej fronty IF** (FIFO)
- vykonávacia jednotka **vykonáva** inštrukcie z inštrukčnej fronty



Problémy pri použití inštrukčnej fronty (skokové inštrukcie)

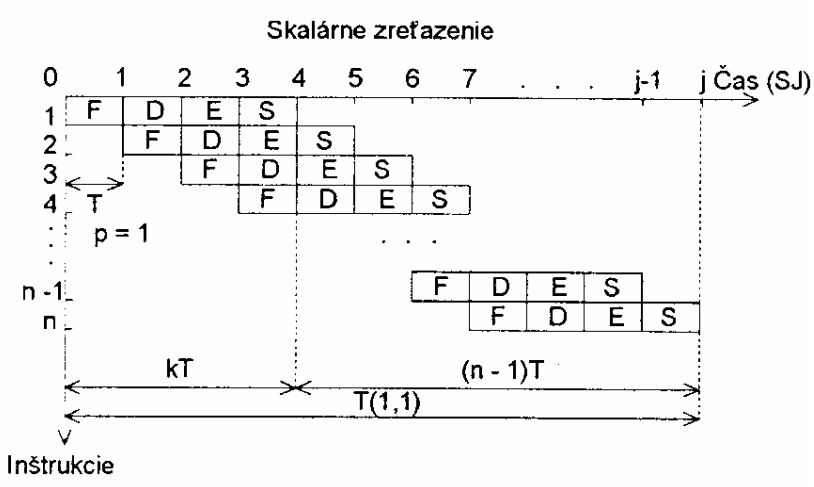
-**nepodmienený skok** môže riešiť jednotka IU (začne plniť frontu od adresy skoku)

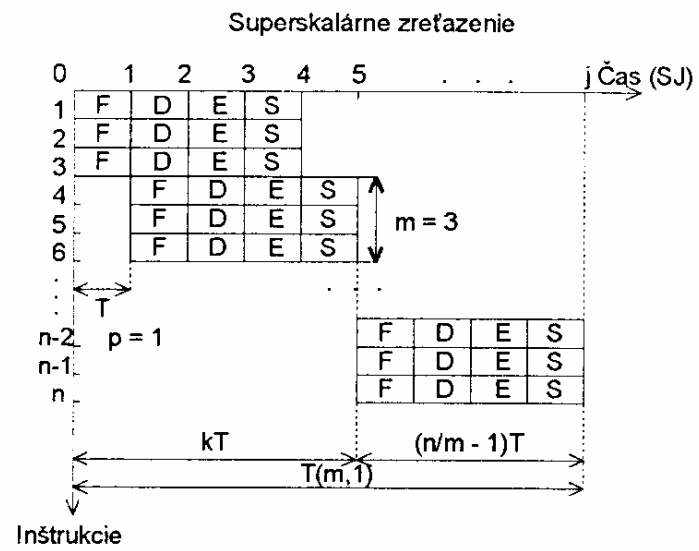
- v prípade **podmieneného skoku**, sú 3 riešenia
  - naplňovanie IF sa pozastaví po vyhodnotenie podmienky
  - v prípade, že podmienený skok vedie k zmene inštrukčného toku (vyprázdni sa inštrukčná fronta (je určitá pravdepodobnosť, že táto situácia nenastane), ale v prípade vyprázdnenia vykonávacia jednotka stojí (čaká na naplnenie FIFO)
  - v dokonalejších systémoch sa napĺňajú dve IF (hlavná a pomocná)

Zväčšenie počtu sekcií vedie k prúdovému spracovaniu inštrukcií (všetky sekcie sú aktívne v každom takte)

1.instr.	VI	DI	VO	PO	ZV							
2.instr.		VI	DI	VO	PO	ZV						
3.instr.			VI	DI	VO	PO	ZV					
4.instr.				VI	DI	VO	PO	ZV				
5.instr.					VI	DI	VO	PO	ZV			

Skalárne a superskalárne zreťazenie inštrukcií (CPU musí mať m- rovnakých jednotiek pre každú fázu vykonania inštrukcie):





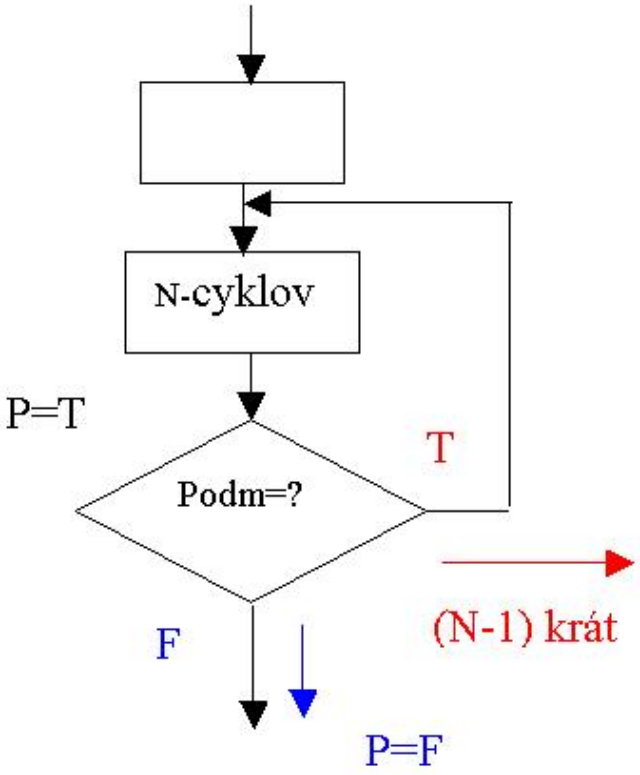
Problémy pri prúdovom spracovaní inštrukcií :
- podmienené skoky (štatisticky 15 až 25 % všetkých vykonávaných inštrukcií)
- závislosť inštrukcií na výsledku predchádzajúcej inštrukcie
- vykonanie pomalej inštrukcie (napr. čítanie z pamäte)

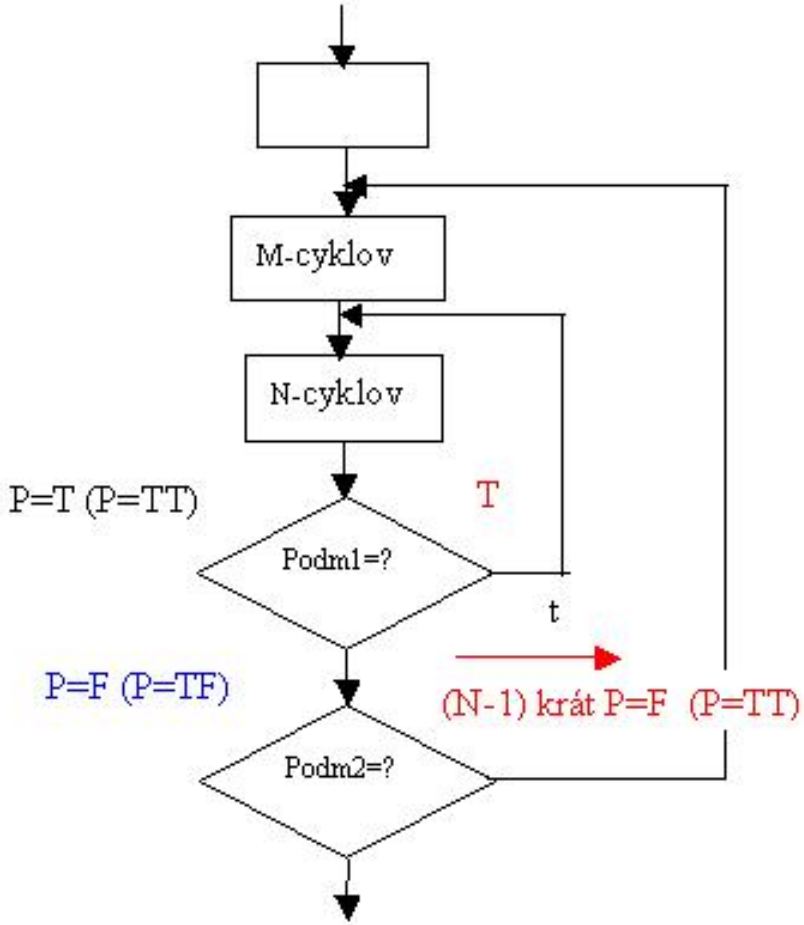
Riešenie podmienených skokov :

1. Bit predikcie skoku

Skokové inštrukcie obsahujú bit predikcie skoku, môže byť

- **statický** (nastaví kompilátor (programátor) pravdepodobnejšiu hodnotu)
- **dynamický** – nastavuje sa počas behu programu
- nevýhoda jedného bitu, nevie predvídať prechod cyklom (najmä vnorených cyklov) preto sa používajú **dva bity** (stav sa zmení po 2 neúspešných naplneniach IF)





Využitie bitu predikcie skoku

a. (1 bit a jednoduchý cyklus)
b. (2 bity pre vnorený cyklus)

2. **Oneskorenie inštrukcie skoku**

pri podmienenom skoku (ak dôjde k zmene toku programu) sa vykoná vždy niekoľko inštrukcií za inštrukciou podmieneného skoku

Potom je potrebné využiť miesto za inštrukciou podmieneného skoku (dať prázdne inštrukcie) tzv. bubliny (z hľadiska praktického je to isté ako zrušenie IF)

Dobré kompilátory dajú za podmienený skok inštrukcie napr. z tela cyklu, ktoré sa vykonajú vždy (dobrý kompilátor ich využije na 90%)

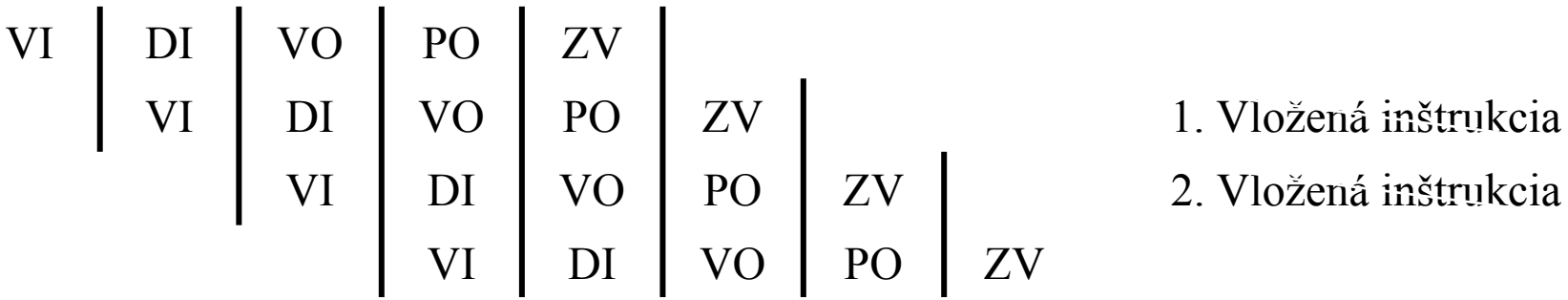
3. **Asociatívna pamäť skokov**

Obsahuje *n* posledných skokových inštrukcií (netreba čakať na čítanie inštrukcie z pamäte, ktorá nasleduje za skokovou inštrukciou)

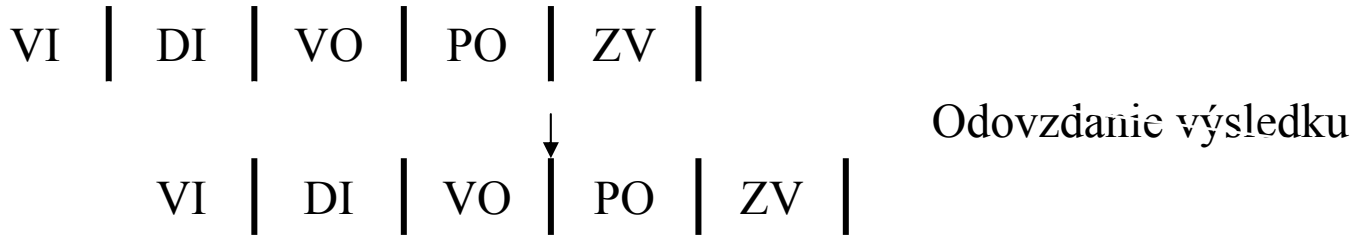
Je to vlastne cache (napr. MC 88110 obsahuje 32 inštrukcií, vždy 2 inštrukcie, ktoré nasledujú po skoku)

Riešenie odovzdávania výsledkov :

1. Vloženie prázdnych inštrukcií, ak je rozpoznané, že je potrebné mať k dispozícii výsledok predchádzajúcej operácie



2. Blokovanie sekcií, „vykonávanie“ druhej inštrukcie zistí, že jej chýba, operand, tak „počká“.



Vylepšenie odovzdanie medzivýsledku cez registre .

Literatúra:

[1] Strelec,J. Líška, M..: Architektúra procesorú RISC, Grada...,

[2] Jelšina, M.: Architektúry počítačových systémov, ...

[3] Hlavička, J.:Architektura počítačú. ČVUT 1998

[4] Krajčovič, T.: Počítače. STU FEI 2000