

# Počítačové systémy

## 7 Mnohaúrovňová organizace počítače 1

# Mnohaúrovňová organizace počítače

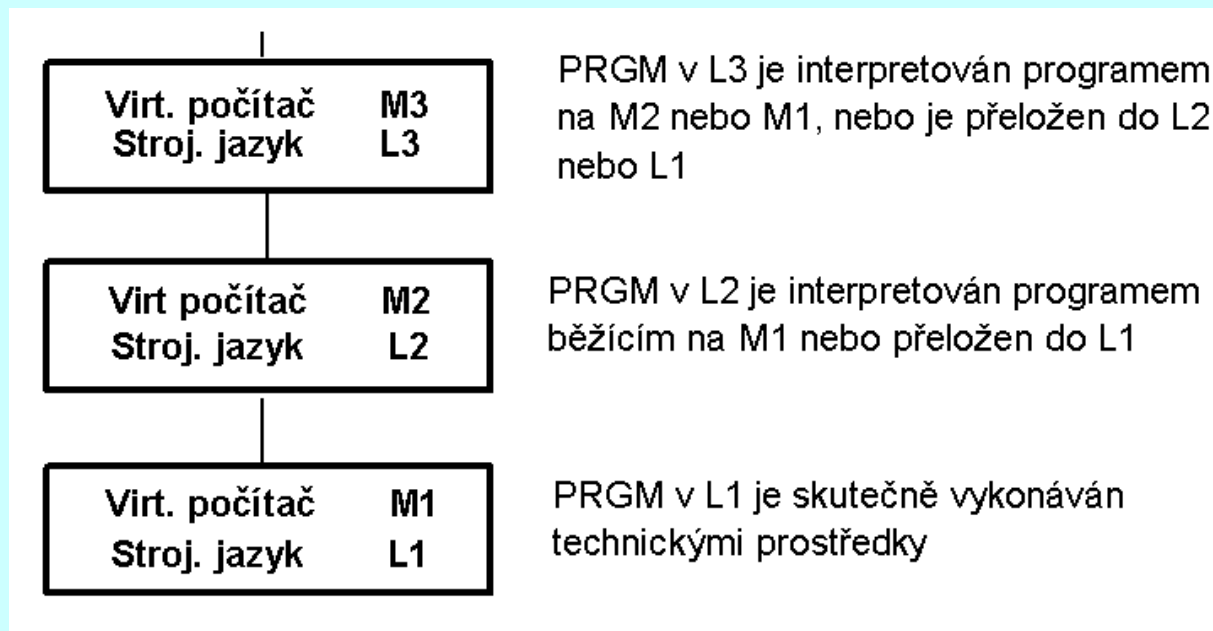
**Strojový jazyk počítače** - množ. jedn. instr. - do ní převést prog. pro výkon  
 - úroveň L1 - abeceda {0,1} , obtížná komunikace

Jazyk vyšší úrovně - vhodnější pro lidskou komunikaci L2 + další

## Vykonání programu v L2 na stroji jenž má L1

**Kompilace** - instrukce v L2 se nahradí posloupností instrukcí v L1

**Interpretace** - program v L1 zpracovává program v L2 jako data (**pomalejší**)



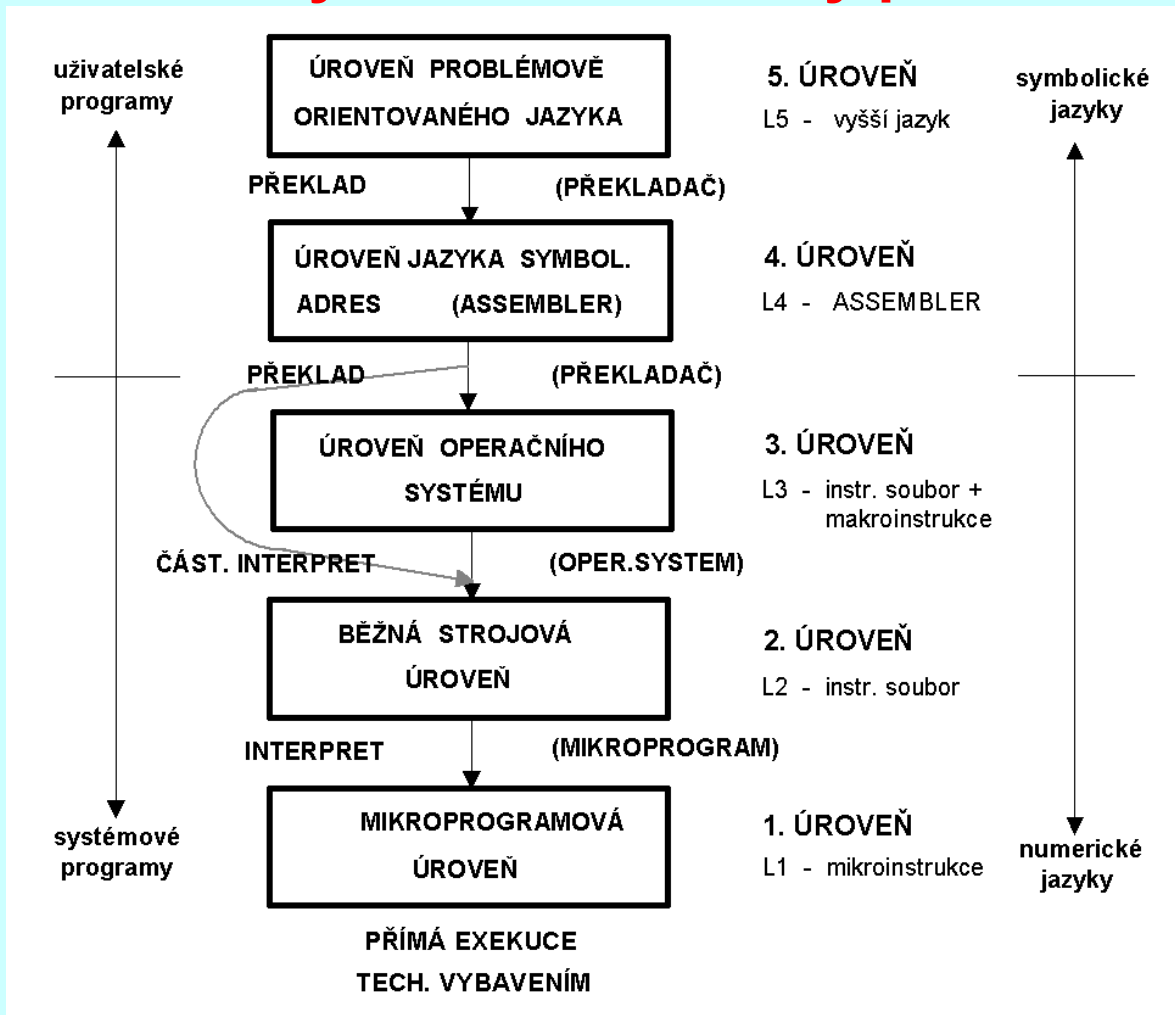
## Virtuální počítač

Na úrovni  $i$  zavádíme virtuální počítač  $M_i$  s jazykem  $L_i$ .

Program v  $L_i$  je překládán nebo interpretován počítačem  $M_{i-1}$  atd.

# Mnohaúrovňová organizace počítače

## Současný mnohaúrovňový počítač



### Vývoj víceúrov. stroje

- první počítače - běžná strojová úroveň - **1.úr.**
- 50- tá léta - Wilkes – mikroprogram. - **2.úr.**
- 60- tá léta - operační systémy - **3.úr.**
- překladače, program. jazyky - **4.úr.**
- uživatelské aplikační programy - **5.úr.**

HW a SW jsou logicky **ekvivalentní** (lze je navzájem nahradit)

souboj **RISC** a **CISC** procesorů

# Mnohaúrovňová organizace počítače

## Procesy a jejich stavy

**PROCES** - probíhající program (program - pasivní, proces - aktivní)

**STAV PROCESU** - info, které při zast. procesu umožní jeho znovuspuštění

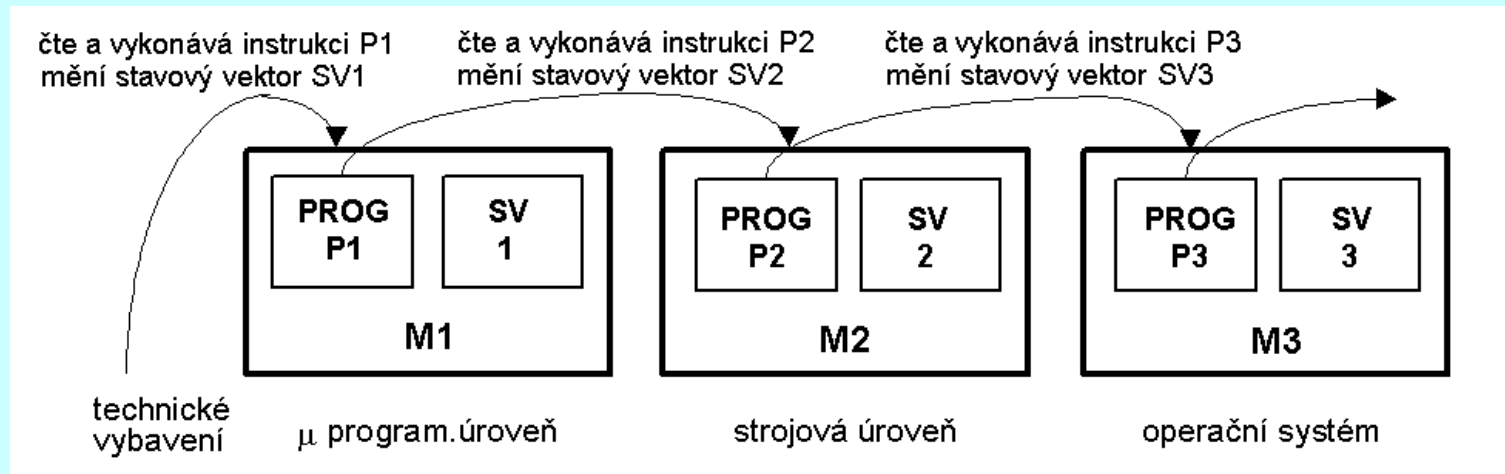
1. program
2. násled. instrukce
3. hodnoty proměnných a data
4. stavy a polohy všech I/O

**PŘEDPOKLAD:** proces P sám nemění svůj program!

**STAVOVÝ VEKTOR** - proměnné složky stavu procesu – mění HW nebo prgm.

**PROCES = PROG + STAV. VEKT.**

**ZMĚNA STAV. V.** – stav. vektor proc. P2 mění P1 -> P1 interpret programu P2



# Konvenční strojová úroveň počítače

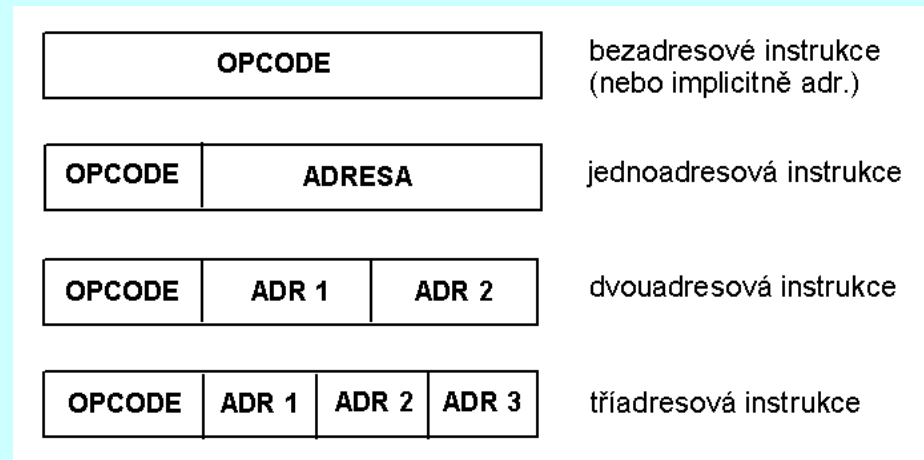
## M2 Konvenční strojová úroveň

**HW** - strukt. poč., proc., I/O kan., sběrnic, org. a příst. k pam., org.reg. a j.

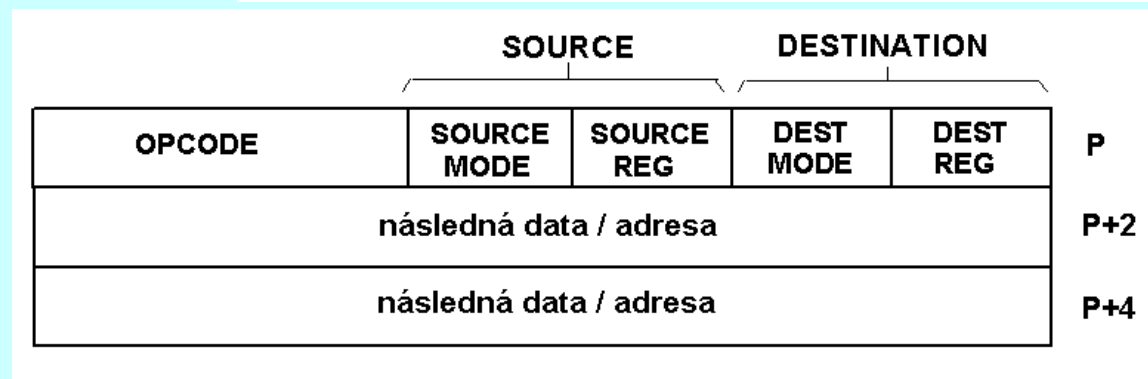
**SW** - instr.soubor, formát dat a uložit. v pam., adresování, zás.pam, reg aj.

### Instrukční formát

- skládá se z polí, (po sobě jdoucí slova)
- adr. části mohou být adresami nebo odkazy na reg. (nepřímo)



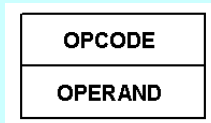
Nejpoužívanější formát je dvouadresový



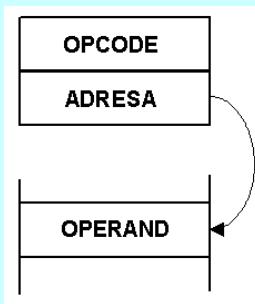
# Konvenční strojová úroveň počítače

## Adresování

### 1. Bezprostřední



### 2. Přímé adr.

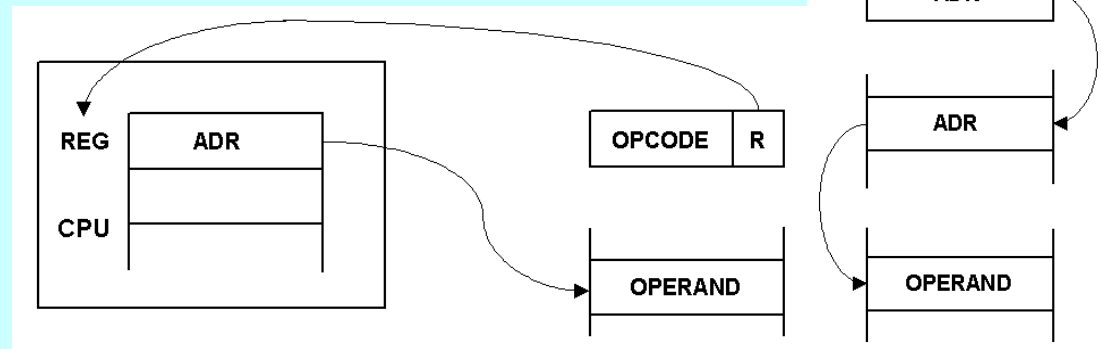


### 6. Registrové

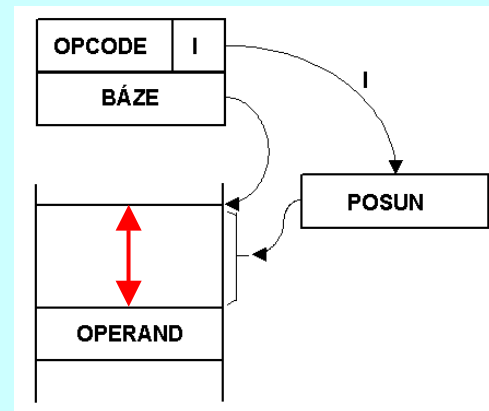


další způsoby adresování  
včetně jejich kombinací.

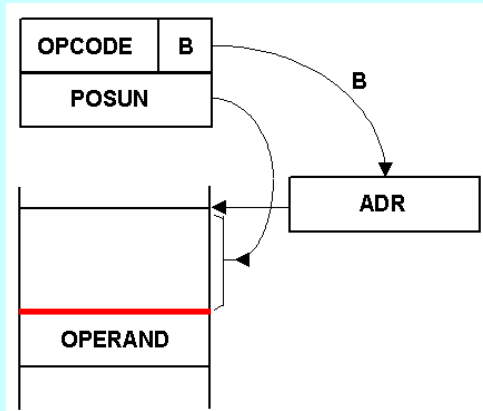
### 3. Nepřímé adr.



### 4. Indexové adr. (autoindexové)



### 5. Bázovými registry (relativní – PC)



# Konvenční strojová úroveň počítače

## Reprezentace dat

Typ dat, jejich struktura i umístění v pam. definuje instr.soubor procesoru

1. **Pevná řádková čárka** - min.zákl.pam. buňka (slovo). Data s vyšší přesn. na násl. adr. Záporná čísla v 1 nebo 2 doplňku.
2. **Pohyblivá řádková čárka** - znaménka, mantisa, exponent. Rozsah dat závisí na instr. souboru.
3. **Boolská data** - proměnné 0 nebo 1, přiřazeno buď celé pam. slovo (zbytek 0) nebo bitové přiřazení.
4. **Znaky** - v jedné paměť. buňce (slově) jeden znak (ASCII). Ostatní bity 0.
5. **Řetězce** - uchování textů - několik znaků ve slově, násl. na dalších adr. - každé slovo 1 znak + adresa pokračování
6. **Pole** - používá se báze + indexové adr. Sousední index na soused. adr.

# Konvenční strojová úroveň počítače

## Instrukční soubor

### Rozdělení instrukcí z hlediska

- **struktury** proc.: k jakým částem se instr. vztahují (adr., reg., I/O, float aj.)
- **funkčního**: jak mění stav. vektor proc. (přenos dat, výkon. oper., řídicí aj.)

### 1. Přenosy dat - vytvoření nové kopie dat v novém místě a zachování původ.

LD, ST, MOV, XCHG

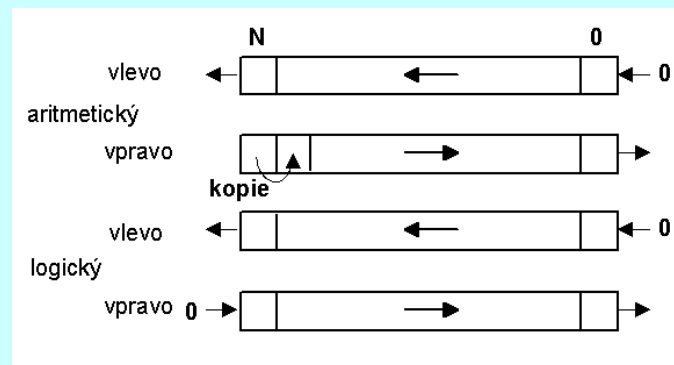
### 2. Dyadické operace

- aritmetika integer / real +, -, x, /
- logika - úplný soubor

ADD, SUB, MPY, DIV, FADD, FSUB, FMPY, FDIV, OR, AND, XOR

### 3. Monadické operace - s jedním oper.

- nulování
- inkrementy/dekrementy
- změna znaménka
- negace (bitová inverze)
- posuny
- rotace (i přes CY bit)

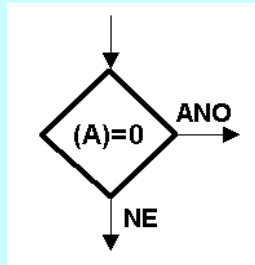


INC, DEC, NEG, SHR, SHL (log.posun), SAR, SAL (ar.posun), RAR, RAL (rotace)



# Konvenční strojová úroveň počítače

## 4. Větvení programů - skoky - podmíněné/nepodmíněné (změna PC)

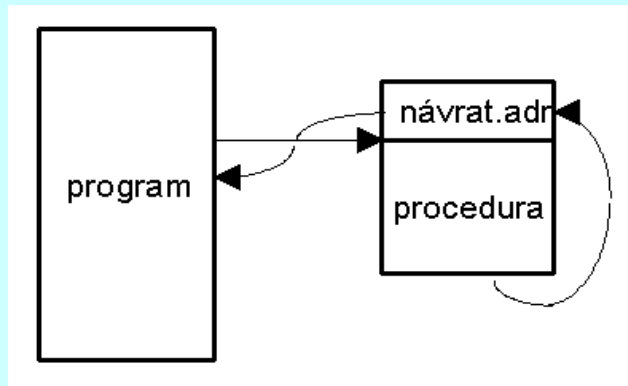


### Podle výsledku testu:

- vynechání/exekuce násl. instr. (SKIP)
- skok/pokračování (adr. skoku v instr.)

**JMP, JT, JF, BR**

## 5. Volání podprogramů - někdy i podmíněné, nutné uchovat adr. návratu



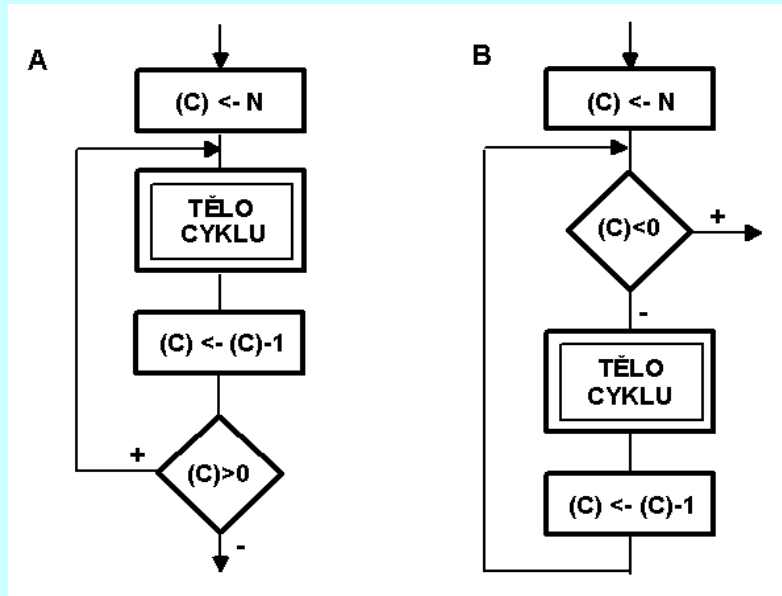
### Uchování návrat. adresy:

- **do paměti** - obvykle 1.slovo podprog. návrat jako nepřímý skok (přes 1.sl.)
- **do registru** - odpovědnost má podpr.
- **do zásobníku** - návrat prepisem PC z vrcholu zásobníku

**CALL** - volání  
**RET** - návrat

# Konvenční strojová úroveň počítače

## 6. Řízení cyklů - podmíněné skoky dané testem počtu průchodů cyklem.



Varianty:

A - ne pro  $N=0$

B - i pro  $N=0$

JCXZ, LOOP

7. Instrukce I/O – samost. adr. prostor, většinou pouze přesuny, log oper.

IN, OUT

8. Strojní instrukce - řízení procesoru, zákaz/pov. INT, stop, wait, NOP aj.

EI, DI, HLT, NOP

9. Další podle struktury procesoru a jeho funkce

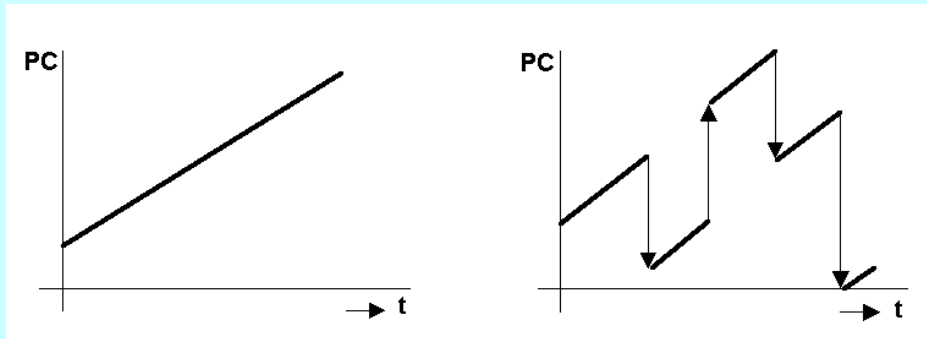
# Konvenční strojová úroveň počítače

## Řízení procesoru programem

### Předávání řízení mezi instrukcemi:

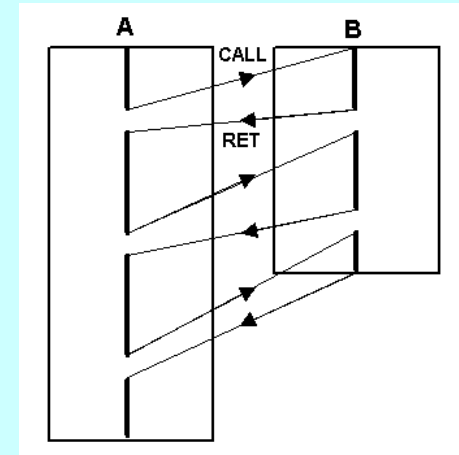
- **na instr. s následující adr.** t.j. předávání řízení je lineární vzhl. k adr.
- **narušují:** skoky, větvení, přeskoky, procedury, korutiny, přerušení

### 1. Lineární předávání řízení (skoky)



špatně struktur. prog., nečitelnost, chyby, obtížné ladění. Skoky nelze eliminovat na konv. stroj. úrovni (JMP, GOTO)

### 2. Přechody do **podprogramů**



technika strukt.program., procedura je instr. vyšší úrovně.

### 3. **Přerušení INT** - přechod na adr. vyvolaný HW i SW, návrat jako procedura **synchronní** (programem), **asynchronní** - (I/O, chyba)

**Předávání parametrů.** - **přes registry** (málo), **přes paměť** + adr. (lib. počet), **přes zásobník** (pozor na návrat)

# Mikroprogramová úroveň úroveň počítače

## M1 Mikroprogramová úroveň

**Nejnižší úroveň** z celé hierarchie – tech. vybavení reál. počítače hradla, registry, čítače, aj. Respektuje obvod. prostř., časování.

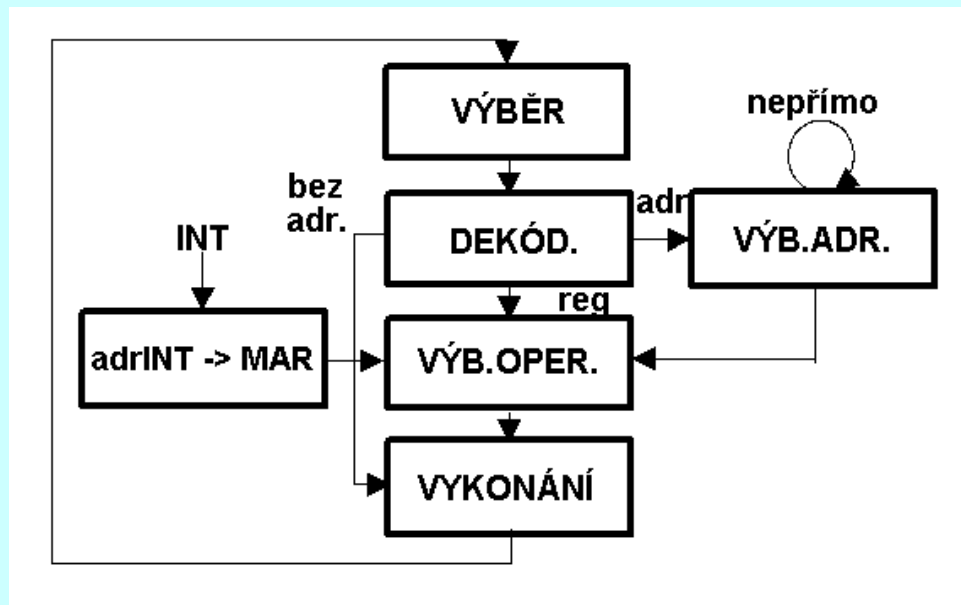
**Úroveň** - **hostitelská** = **mikroprogramová** - na této běží interpreter  
- **cílová** = **konv. stroj.** - programy jsou interpretovány

Interpretace instrukcí M2 mikroprogramovou úr. M1 představuje ovládání tech. prostředků řadičem - **2 složky**:

- **programová** - představuje **cykly**: výběr instrukce a provedení ucelené sekvence operací (**syntaktická činn.**)
- **mikroprogramová** - představuje **stavy** (subcykly) - provedení jednotlivých operací mikrooperacemi.

# Mikroprogramová úroveň úroveň počítače

## Základní cykl instrukce



### Typy strojních cyklů:

instrukční cykl, čtení z paměti, zápis do paměti, cykl přerušení, vnitřní cykl procesoru

### Prostředky mikroprogramové úrovně:

1. **technické** - komponenty, z nichž je počítač složen
2. **programové** - množina  $\mu$  instrukcí pro řízení tech.prostř.

# Mikroprogramová úroveň úroveň počítače

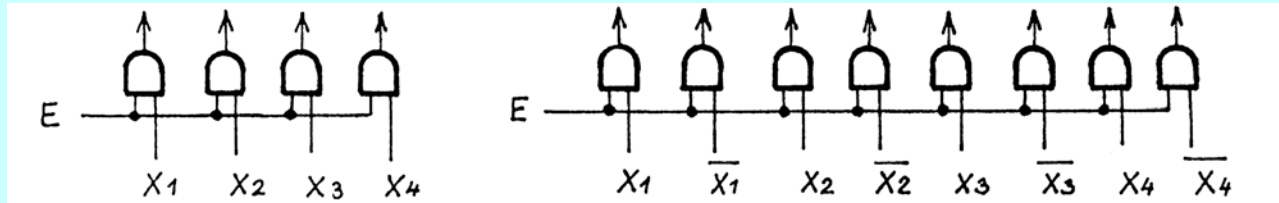
## Komponenty a bloky mikroprog. úrovně - tech. prostředky

**Registry** - paměť informace, rychlá obsluha. Dělení - **statické/dynamické**

**Hradla** - řídí **přenos** po sběrnicích, zápisy do registrů aj.

**Sběrnice** - paralelní přenos info mezi reg., **jednosměrná/obousměrná**.

jedno-  
fázová



para-  
fázová

**Dekodéry** - dekódování stavů, adres, tj. informací v kódu

**Čítače** - čítání v kódu

**Multiplexery** - logické přepínače: podle adr. z n - vstupů na jeden výstup

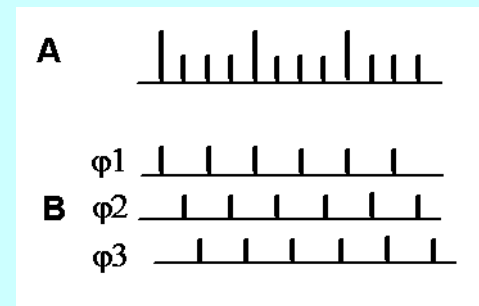
**Paměťové registry** (porty) - adresový (**MAR**) a datový (**MBR**) - vyr. reg.

**Operační jednotka** - ALU - 2 vst.reg., 1 výst.reg.

**Hodiny** - pulsy konst. frekv., synchronizace  
fce procesoru na  $\mu$ prog. úr.

Interval pulsů - **stavy** (fáze, subcykly)

- **cykly** - několik stavů

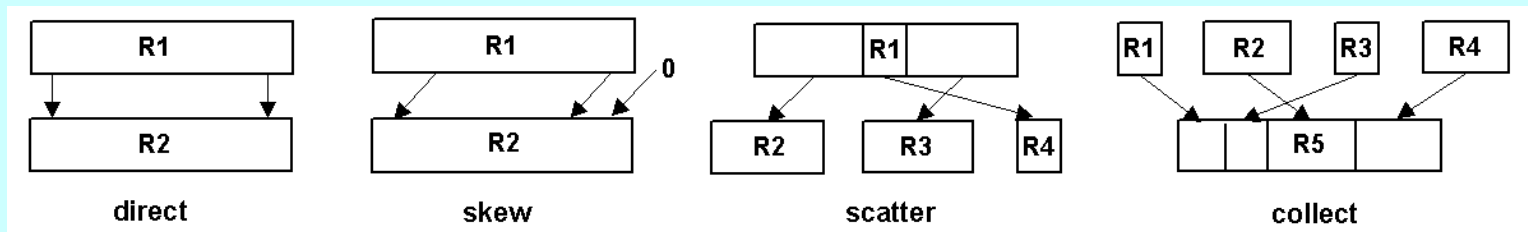


# Mikroprogramová úroveň úroveň počítače

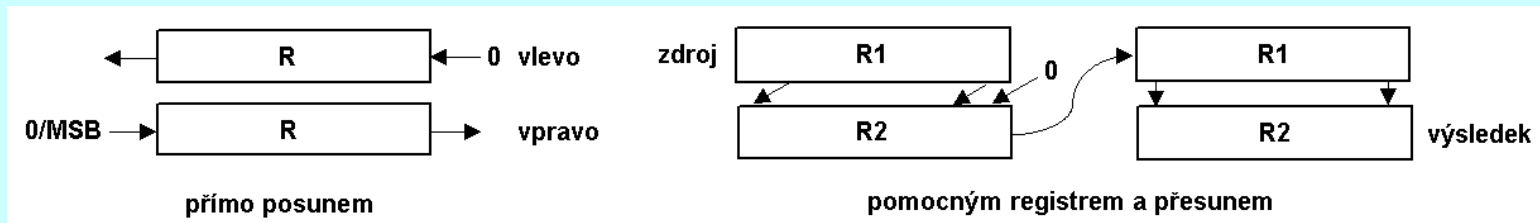
## Základní operace mikroprog. úr. – programové prostředky

Instrukce konv.stroj.úr. jsou sestaveny z mikrooperací -> mikroinstrukce

### a. Přenosy mezi registry



### b. Posuny registrů



c. **Čtení a zápis OP** - **čtení** do MBR z adresy definované MAR  
 - **zápis** na adresu paměti def. MAR dat z MBR

d. **Testování bitů** - slouží jako řídicí mechanismus větvení programů

**způsoby:** - výběr mikroprog. cesty podle testu spec. reg. (instrukce)  
 - pole  $\mu$  instrukce s dalším reg. definuje násl.adr.





# Mikroprogramová úroveň úroveň počítače

## Příklad instrukce ADD STACK

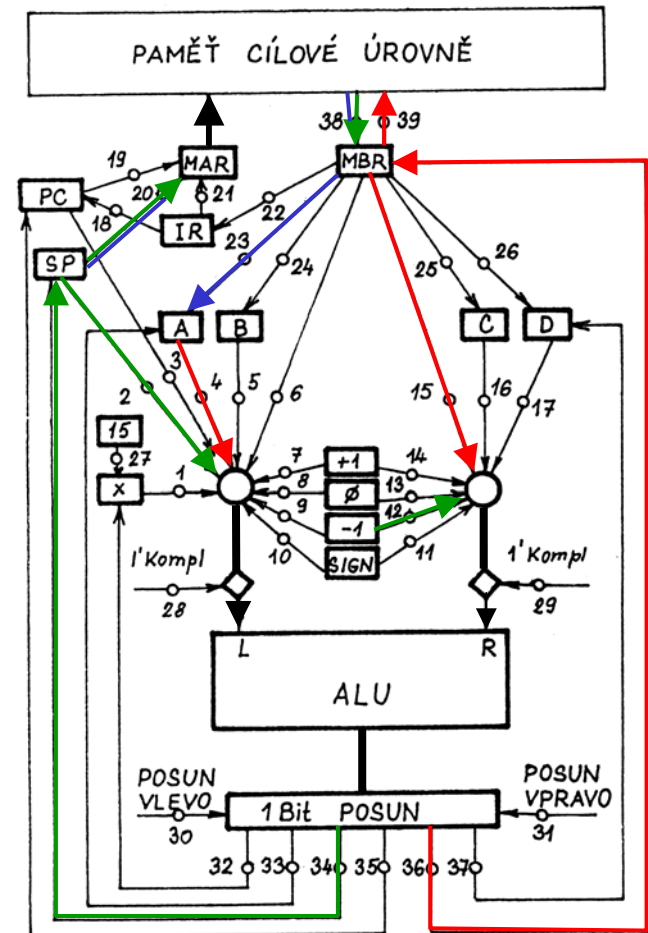
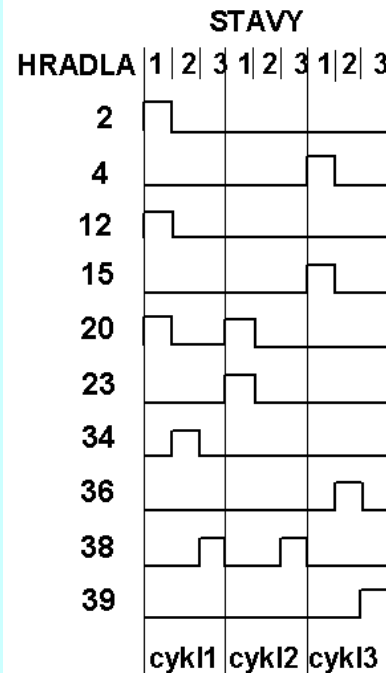
- 1 POP OPER 1
- 2 POP OPER 2
- 3 ADD OPER 1, OPER 2
- 4 PUSH výsledek

CYKL	STAV		
	1	2	3
1	2, 12, 20	34	38
2	20, 23		38
3	4, 15	36	39

Průběh spínání hradlových cest

Některé cesty nesmějí být otevřeny současně -> dělení na stavy.

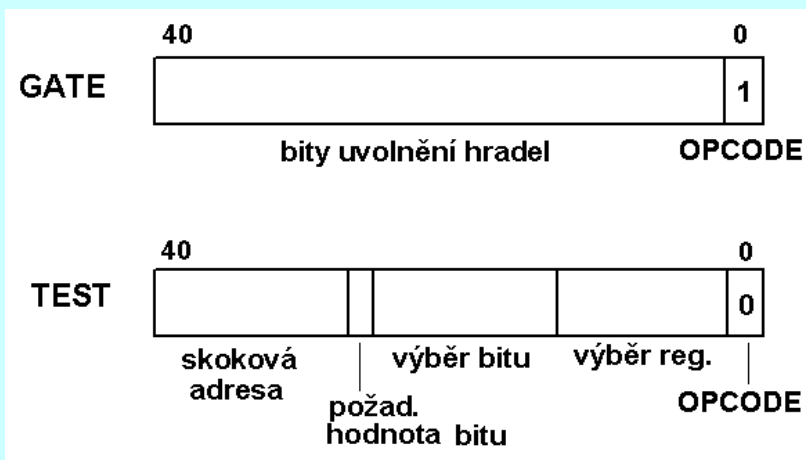
Cesta otevřena po celý stav.



# Mikroprogramová úroveň úroveň počítače

## Realizace $\mu$ P úrovně (příklad)

Pro realizaci  $\mu$ P úrovně stačí dvě **mikroinstrukce**



uvolnění  
hradlované  
cesty

při shodě  
skok na adr.  
jinak NEXT

## Struktura mikroprog. řadiče

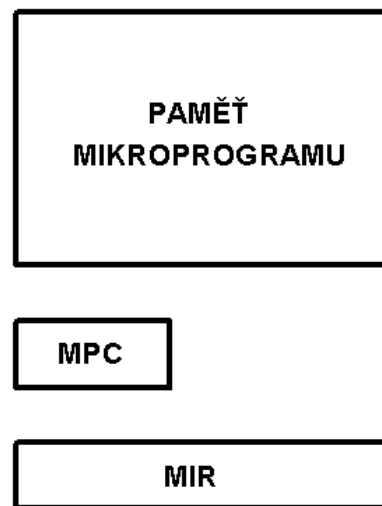
MPC - mikroprogramový čítač  
MIR - mikroprog.instrukční reg.

je-li v MIR instrukce

GATE – uvolňuje hradl.cesty

TEST – test bitu a nahr.MPC

Každá mikroinstr. vyžaduje pouze 1 cykl



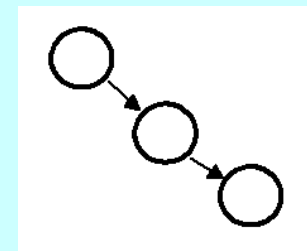
# Mikroprogramová úroveň úroveň počítače

## Základní koncepce řízení

1. **Direktivní** - otevřená smyčka
2. **Zpětnovazební** - uzavřená smyčka

## Direktivní řízení

- působí nezávisle na průběhu řízeného děje
- stačí generovat neměnné impulzní sledy
- **autonomní** sekv. log. obvod
- výstupní písmeno....**mikroinstrukce**

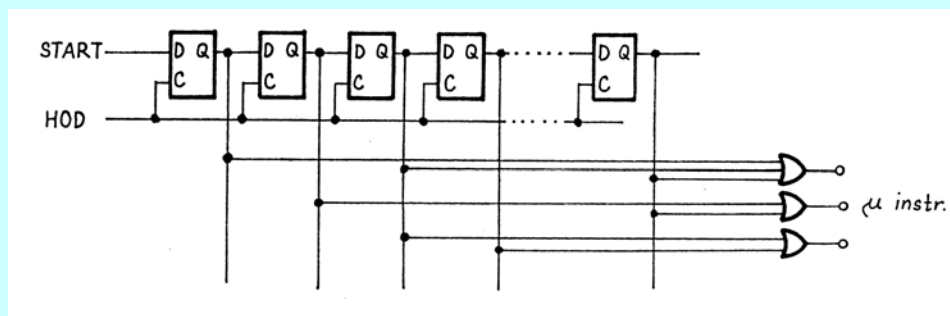


**Mikroinstrukce:** pokyn k elementárnímu úkonu tech. prostředků

Příklad: užití řetězce: násobení .....**přičti** -> **posuň** -> **přičti** ->.....

**Realizace:** podle kódování vnitřních stavů

1. **Unitární kód - kód 1 z n**  
řadič s řetězcí - řetězec  
zpožd. členů - pos. reg.  
s jednou 1  
pam. člen = 1 vnitř. stav.  
budí 1  $\mu$  instrukci.

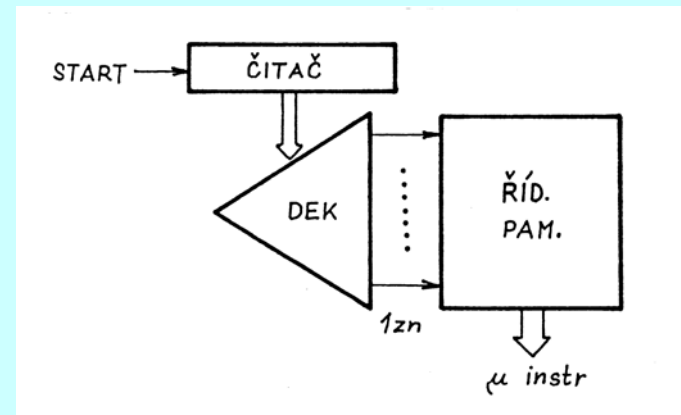




# Mikroprogramová úroveň úroveň počítače

## 2. Binární kód, řetězový kód

Čítač v příslušném kódu.  
 Nutný dekódér na kód 1z n,  
 který řídí paměť  
 Náhrada 1 řetězce



## 3. Obecný sekvenční obvod - běžný návrh dle teorie automatů

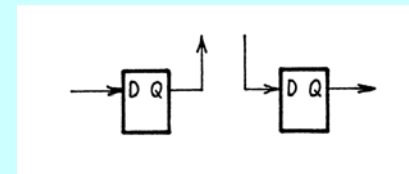
### Varianty direktivního řízení

**centralizované** - jediný automat řídí veškeré prostředky

**decentralizované** - hlavní řídicí automat spouští v určitém kroku lokální automat pro dílčí činnost

Informace o skončení lokálního automatu

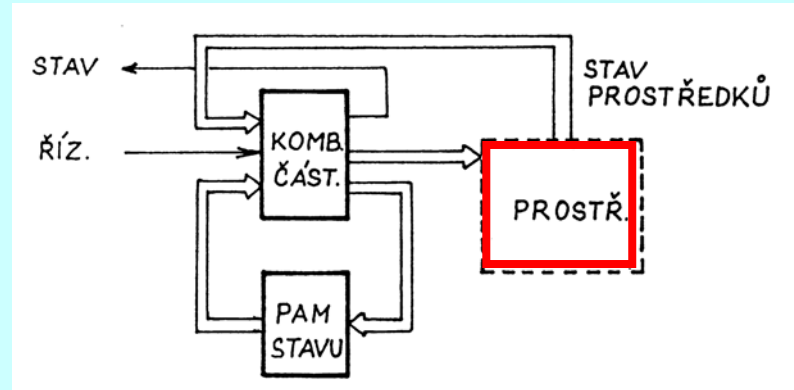
-> **rozpojení řetězce**



# Mikroprogramová úroveň úroveň počítače

## Zpětnovazební řízení

Využívá informaci o stavu řízených prostředků

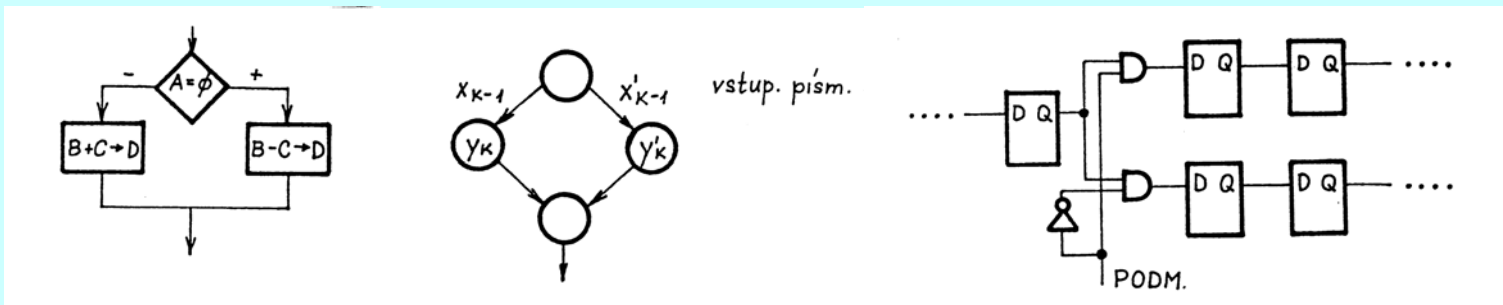


Pro návrh je nutné znát:

Soubor  $\mu$ instrukcí, jejich sekvence a přechody mezi nimi, podm. přechody podle stavu prostředků atd. -> nutné znát  $\mu$  programy

## Zápis mikroprogramu

Používá se zadání vývojovým diagramem (graf přechodů pro automat)



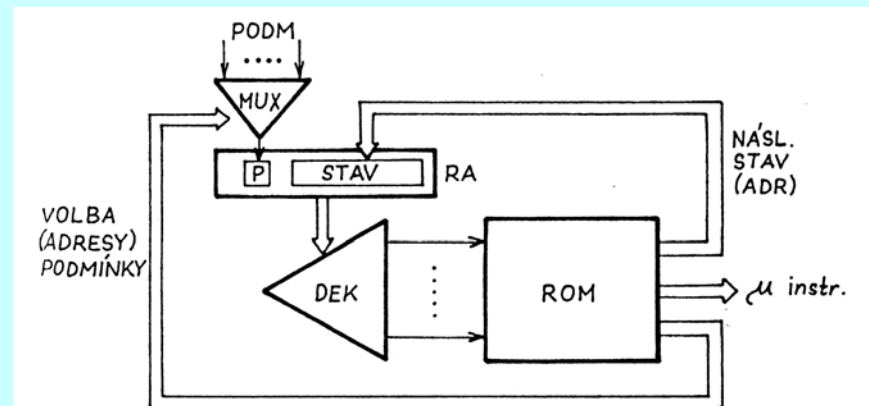
obvykle pouze jediná podmínka

větvení podmínkou (v čítači – předvolba)

# Mikroprogramová úroveň úroveň počítače

## Klasický mikroprogramový řadič

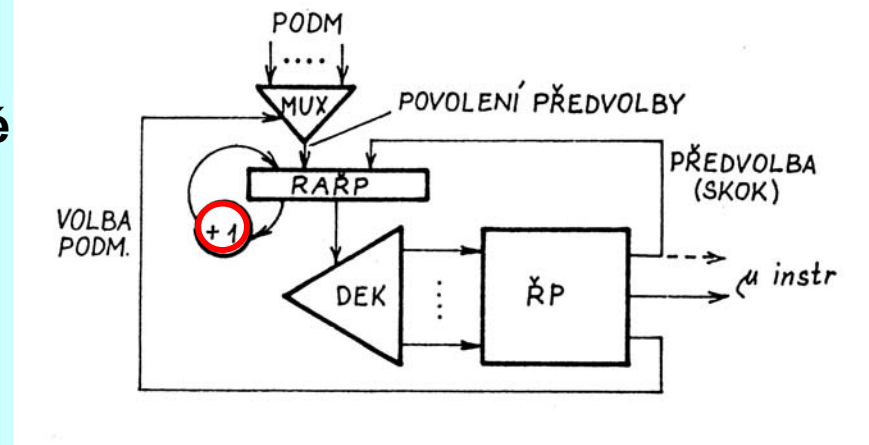
Z množiny vstupních podmínek se vybírá jediná. Vykonávaná mikroinstrukce volí podmínku podmíněného přechodu.



## Modifikace mikroprog. řadiče

Místo registru RA - čítač - souvislé  $\mu$  programy, úspora ROM podm/nepodm skoky - předvolba

Při skoku prostředky čekají



# Mikroprogramová úroveň úroveň počítače

## Variety řídicí paměti

Délka  $\mu$  instrukce  $\wedge$  počtu řídicích signálů (cca 100) => řídká  $\mu$  instrukce

### Horizontální $\mu$ programování

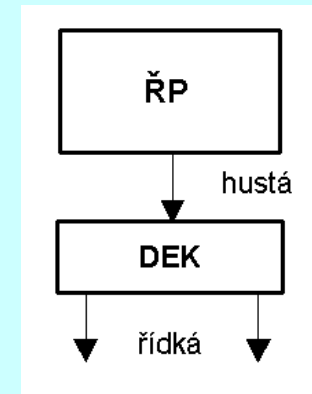
- řídká  $\mu$  instrukce pamatovaná přímo v ŘP.
- délka slova 100 - i více bitů

### Vertikální $\mu$ programování

- $\mu$  instrukce pamatovaná ve zhuštěné formě,
- délka slova ŘP 10-20 bitů, následuje dekód.

### Diagonální $\mu$ programování

- $\mu$  instr. v ŘP rozdělena do polí, obsah některých polí se dekóduje.
- délka slova cca 30 bitů.



## Význam a variety $\mu$ programování

1. řadič řešen jako **centrální** - ovládá veškeré prostř., přiřazení není určeno
2. pracovní registry jako **zápisník** - přiřazení se průběžně mění

### Výkonání instrukce

stroj.instr. svým OPCODE nastaví **poč.adr. RAŘP**, t.j. poč. adr.  $\mu$  programu



# Mikroprogramová úroveň úroveň počítače

## Specifika mikroprogramových automatů

1. bohatá vstupní abeceda (10b -1024 písmen/podmínek)
2. malý počet větvení (v zásadě pouze 2 následné cesty)
3. velký počet vnitřních stavů

**Realizace** - komb. část - ROM (bez optimalizace), větvení max 2 podmínky

**Firmware** - základní naprogramování řadiče + dodávané doplňky

**Dynamické  $\mu$  programování** - ŘP nejen ROM ale i RAM - lze do ŘP zapisovat

## Výhody $\mu$ programování

1. vytvoření „**libovolného**“ strojového kódu nad prostředky.
2. vytvoření variant strojového kódu šitých na míru pro určité **aplikace**
3. **doplnitelnost** strojového kódu - **nový firmware**, zdokonalování starých
4. **emulace** - hostitelský počítač vytváří strojový kód cílového
5. zápis **uživatelských  $\mu$ programů** - doplnění stroj. kódu (ŘP - ROM, RAM)