Úvod do organizace počítače

Organizace procesoru

Návrh datových cest

Opakování

- Konstrukce ALU
  - Stavba bloků (návrh pomocí hradel)
  - Modulární návrh
    - Multiplexor vybírá požadovanou operaci
    - Všechny operace se vykonávají paralelně
  - Sčítačka typu carry lookahead
- Počítačová aritmetika
  - Konečná přesnost
  - Matematické zákony pro reálná čísla „neplatí“ zcela
  - Čísla integer: reprezentace – dvojkový doplněk
  - Čísla FP: IEEE 754 standard

Přehled

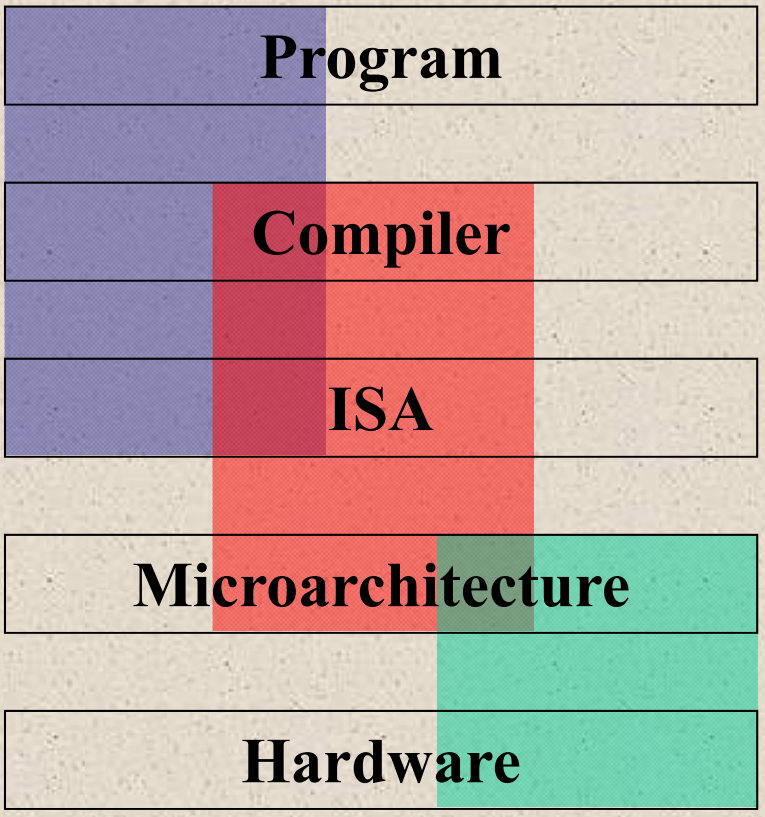
- Organizace počítače (mikroarchitektura)
- Organizace procesoru
- Datové cesty v procesoru
  - Řízení
  - Registrový soubor
- Přehled implementace procesoru
- Metodologie taktování činnosti (Processor Clocking)
- Sekvenční obvody
  - Registry typu Latch
  - Registry

Nové – Návrh datových cest

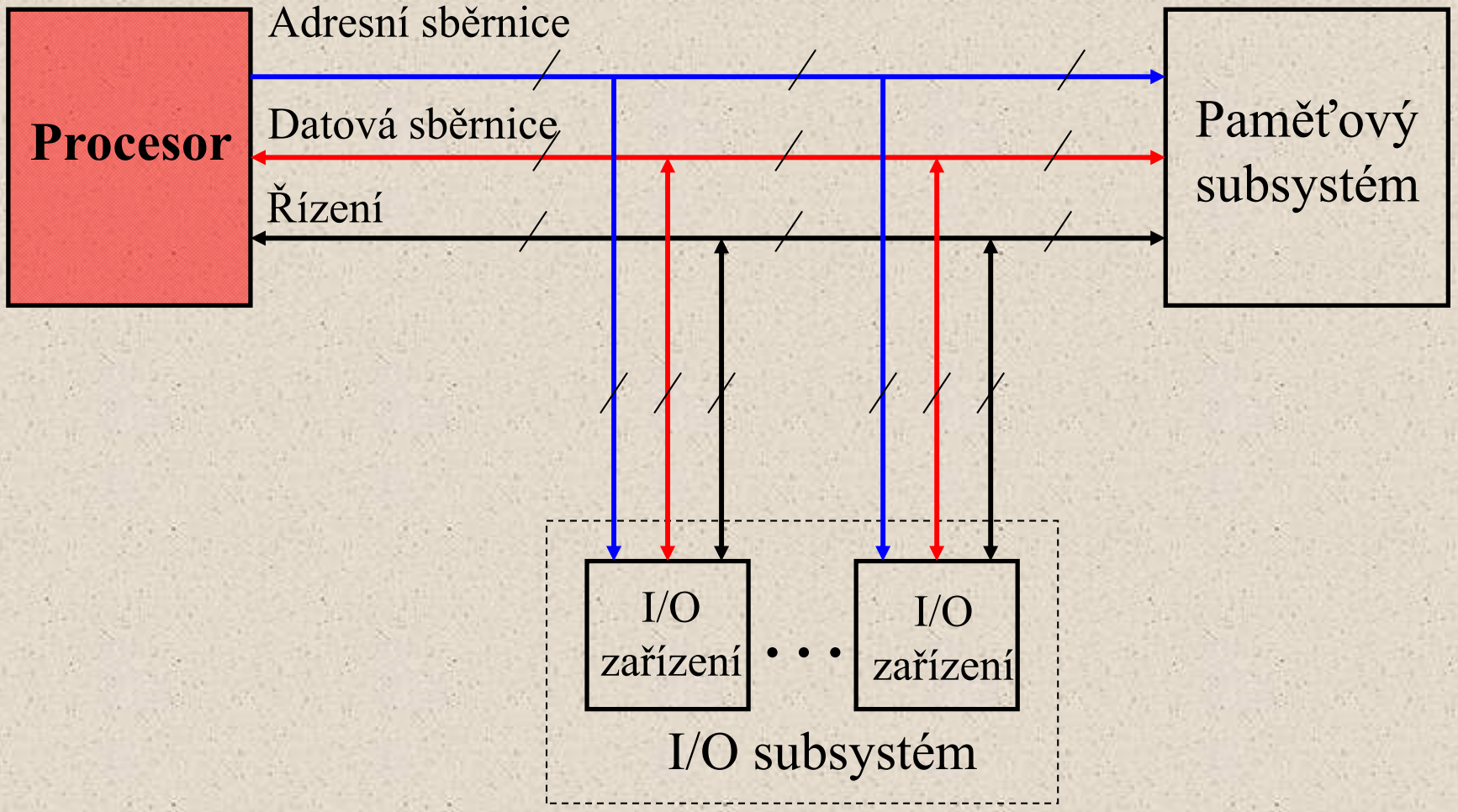
- **Datové cesty -** implementace fází fetch-decode-execute
- **Metodologie návrhu**
  - Určení tříd instrukcí a instrukčních formátů
  - Vytvoření sekcí datových cest pro každý instrukční fmt.
  - Sloučení sekcí tak, aby byly pokryty potřebné přesuny u MIPS
- *Otázka #1*: Co jsou to instrukční třídy?
- *Otázka #2*: Které komponenty jsou vhodné?

Výkon procesoru

CPU time = IC * CPI * Cycle time



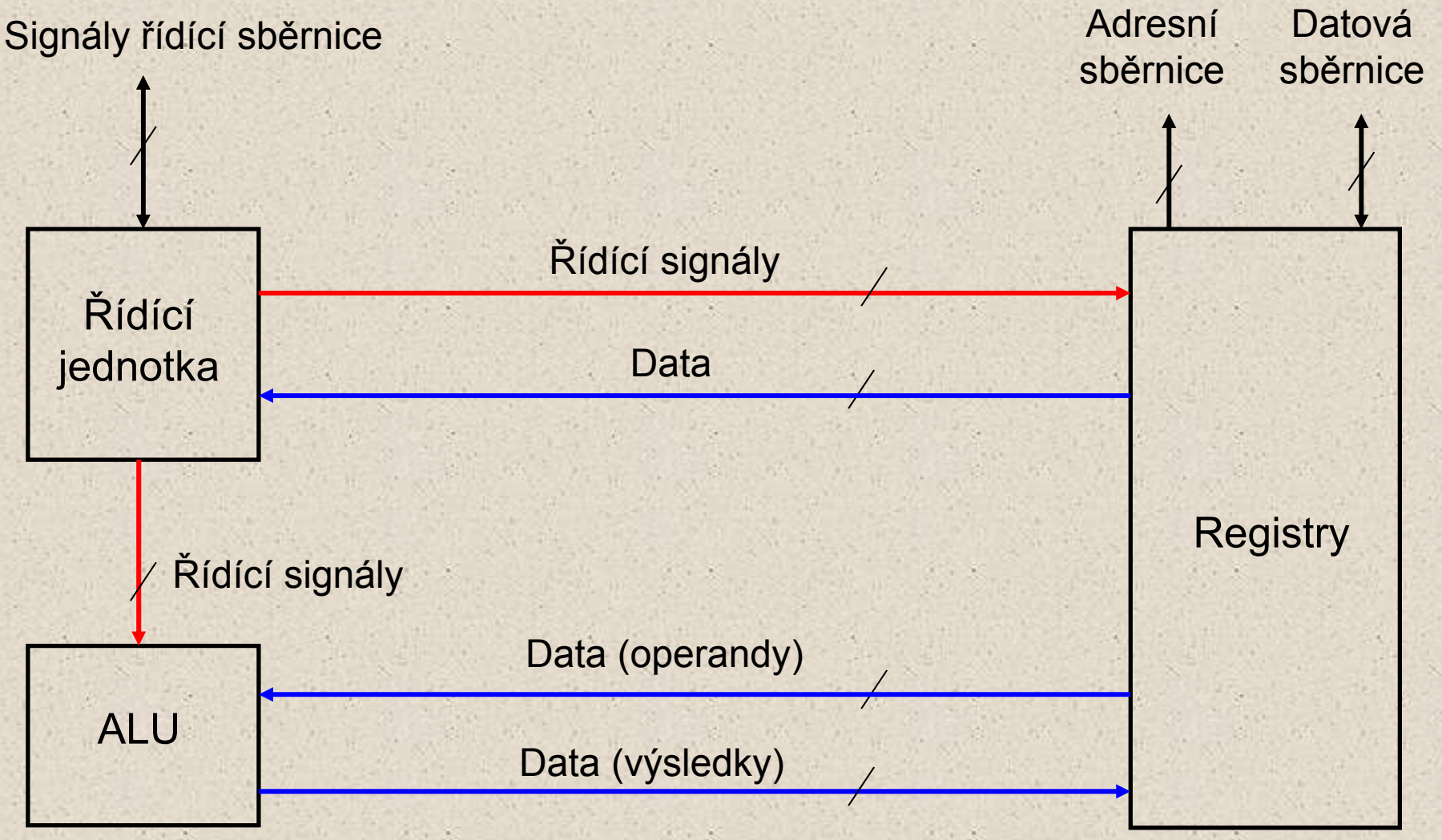
Organizace počítače



Procesor

- Procesor (CPU)
  - Aktivní část počítače
  - Vykonává všechnu „práci“
    - Manipulace s daty
    - Rozhodování
- Datové cesty
  - Hardware, který vykonává všechny potřebné operace
  - ALU + registry + interní sběrnice
  - ***Výkonné prvky***
- Řízení
  - Hardware, který řídí, co se má provádět a kam se co má přesouvat

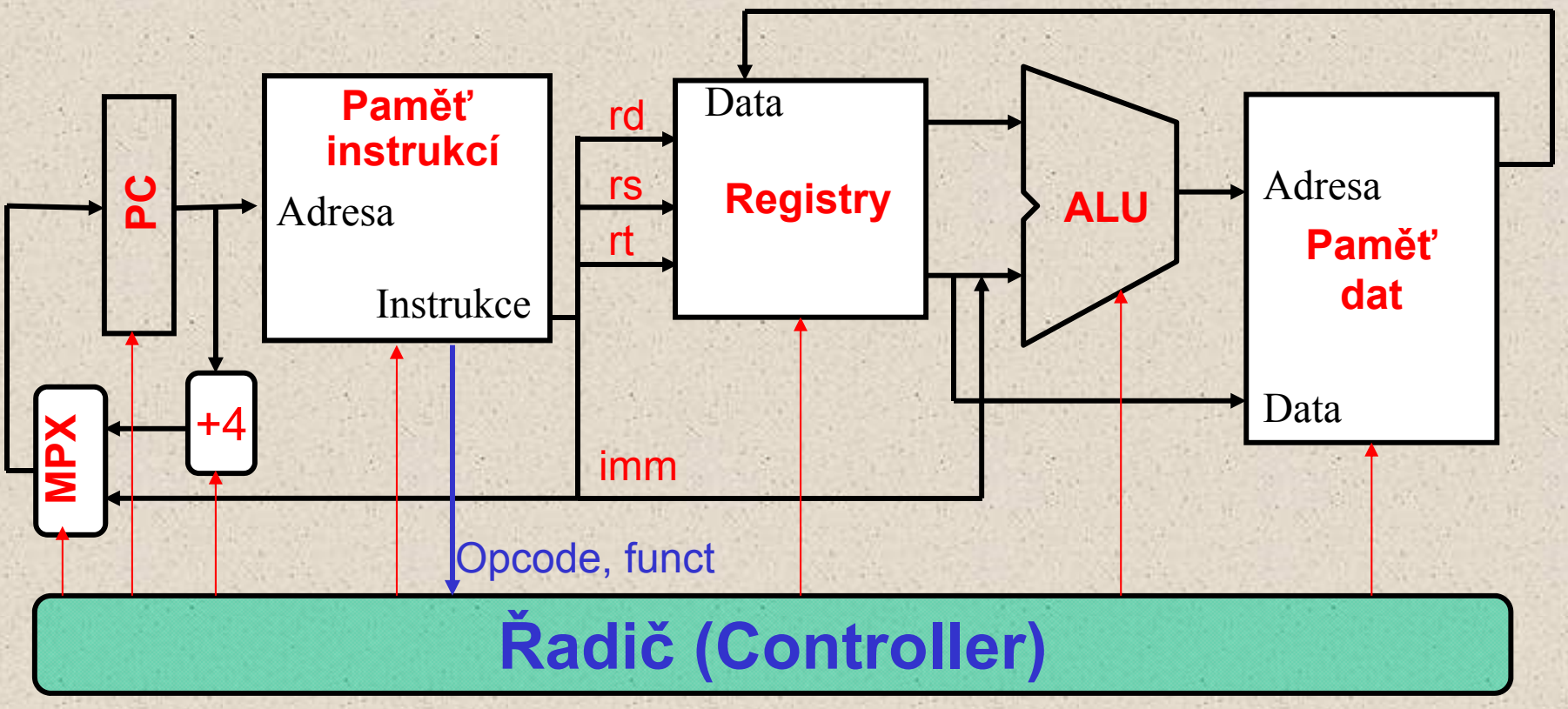
Organizace procesoru



MIPS - implementace

- ISA určuje mnoho aspektů implementace
- Strategie implementace ovlivňuje hodinovou frekvenci a CPI
- Výběr atributů u MIPS pro ilustraci implementace
  - Instrukce pro práci s pamětí
    - Load word (lw)
    - Save word (sw)
  - Aritmetika čísel integer a logické instrukce
    - add, sub, and, or, a slt
  - Instrukce větvení
    - Branch if equal (beq)
    - Jump (j)

Přehled implementace



° Datové cesty umožňují přesuny obsahu registrů při provádění instrukcí

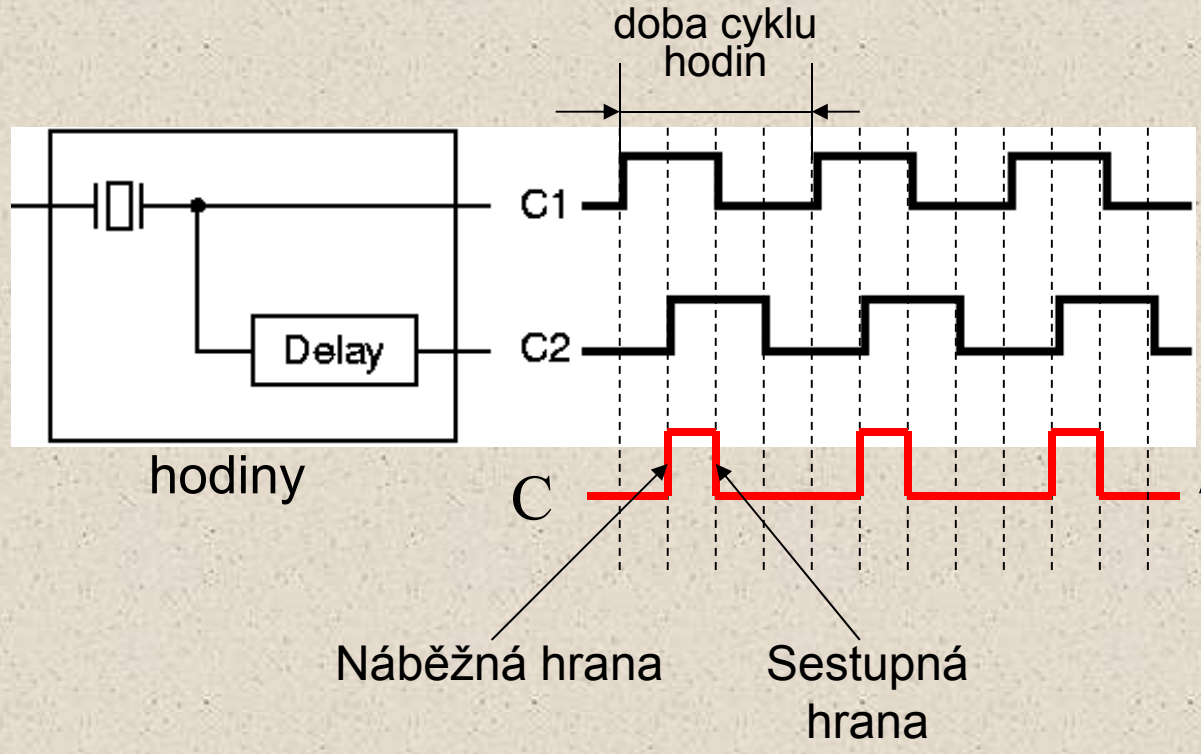
° Řízení zajišťuje provedení správných přesunů

Logika a taktování

- Kombinační prvky
  - Výstupy závisí pouze na aktuálních vstupech
  - Příklad: ALU (sčítačky, multiplexery, posuvové jednotky)
- Sekvenční prvky
  - Obsahují stav (state)
  - Výstupy závisí na vstupech a na stavu
  - Vstupy: data a hodiny (clock)
  - Paměť, registry
- Signály v aserci: logická jednička (vysoká úroveň)

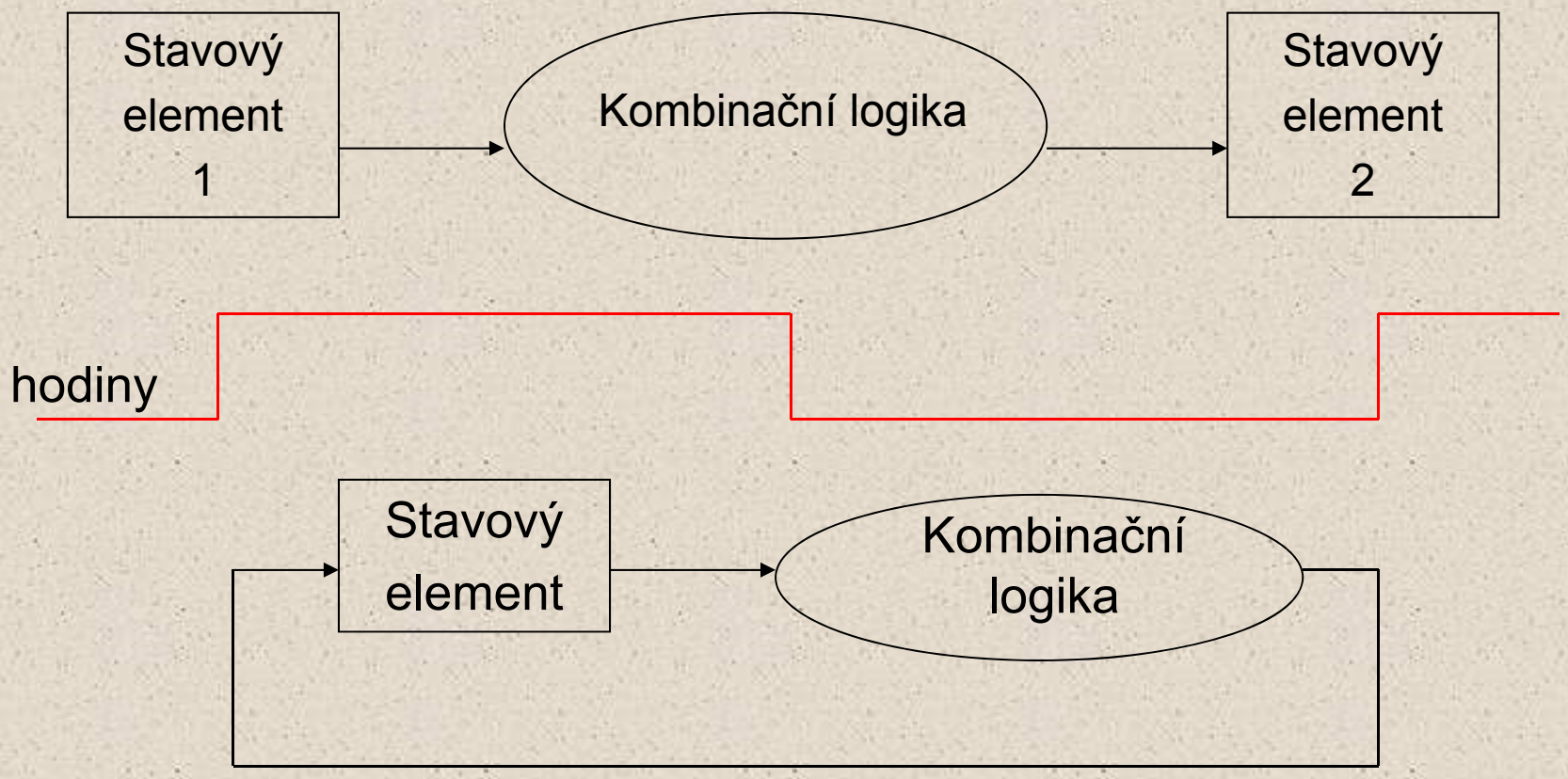
Metodologie taktování

- Určuje pořadí událostí
  - Určuje, kdy lze signály číst nebo zapisovat
- Hodiny: obvod, který generuje posloupnost pulsů

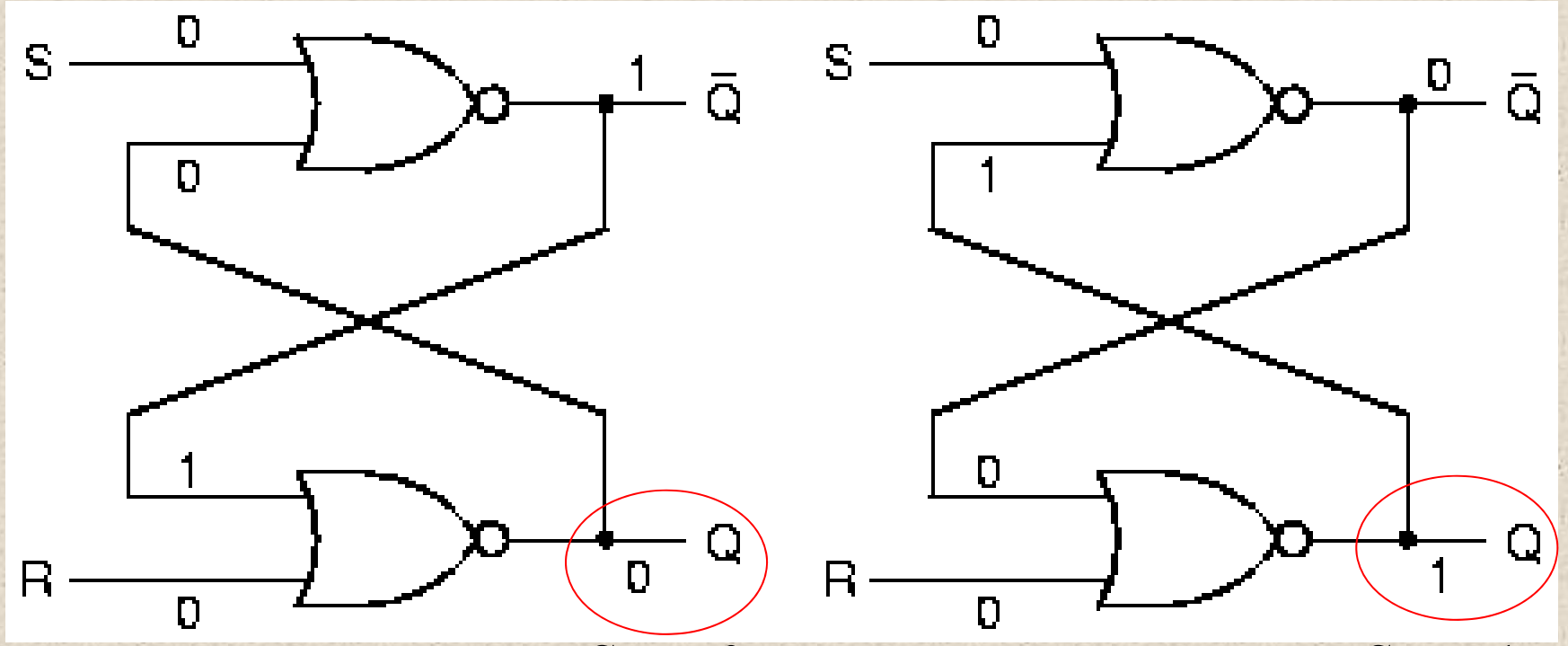


Taktování hranou hodin

- Aktivní je buď náběžná nebo sestupná hrana hodin
- Ke změnám stavu dochází pouze na aktivní hraně hodin



SR Latch s hradly NOR

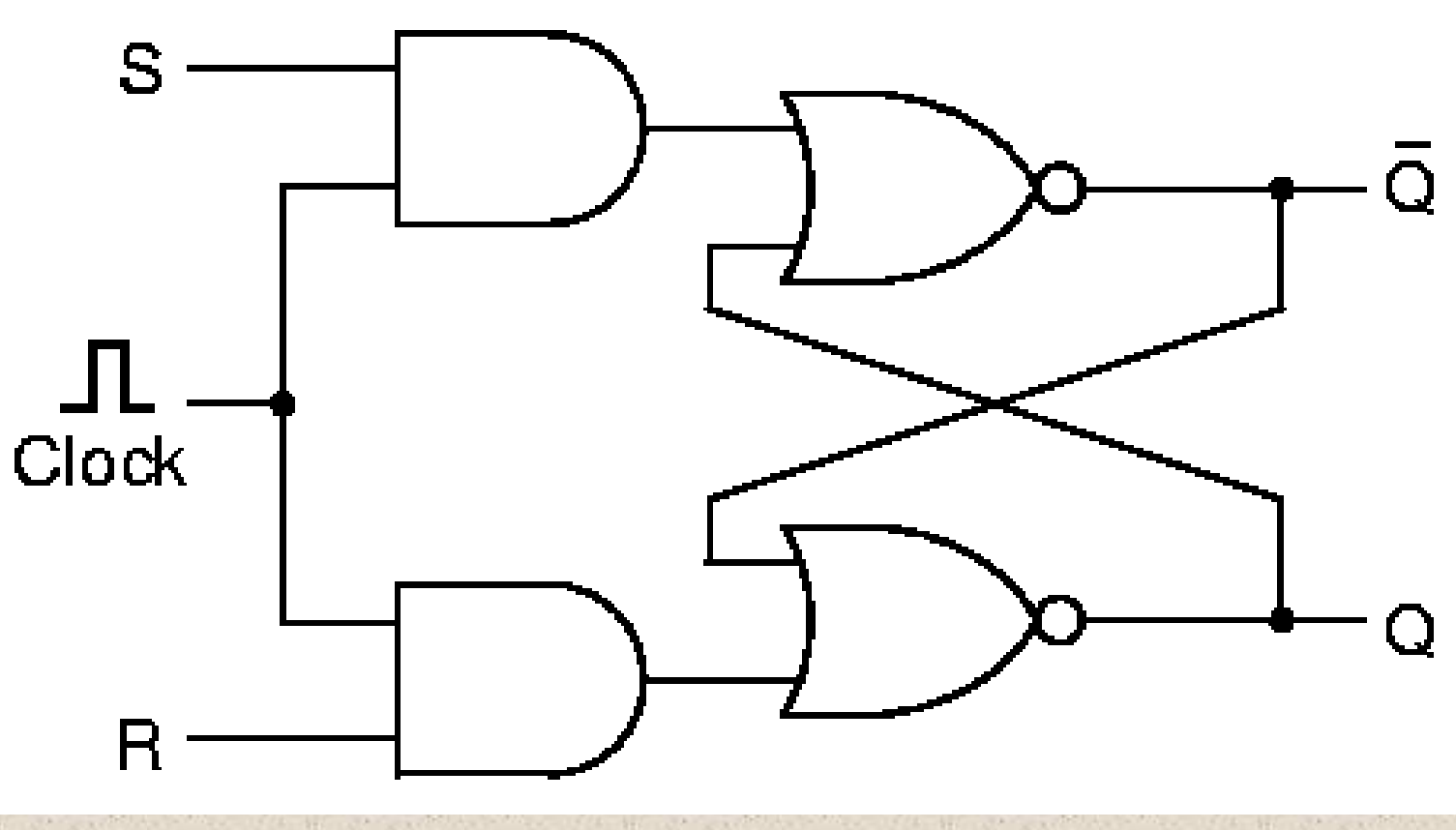


State 0 State 1

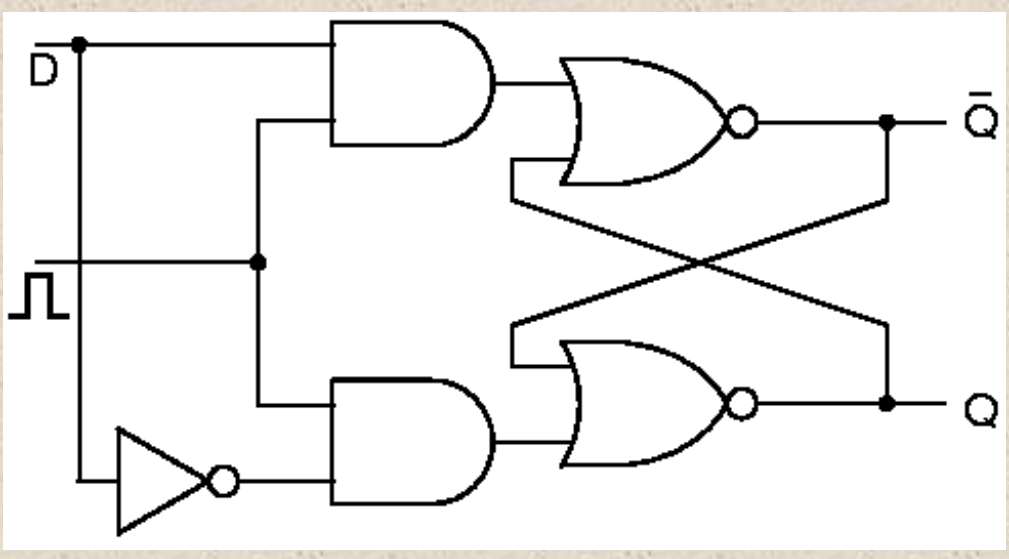
Vstupy { S - set; R - reset

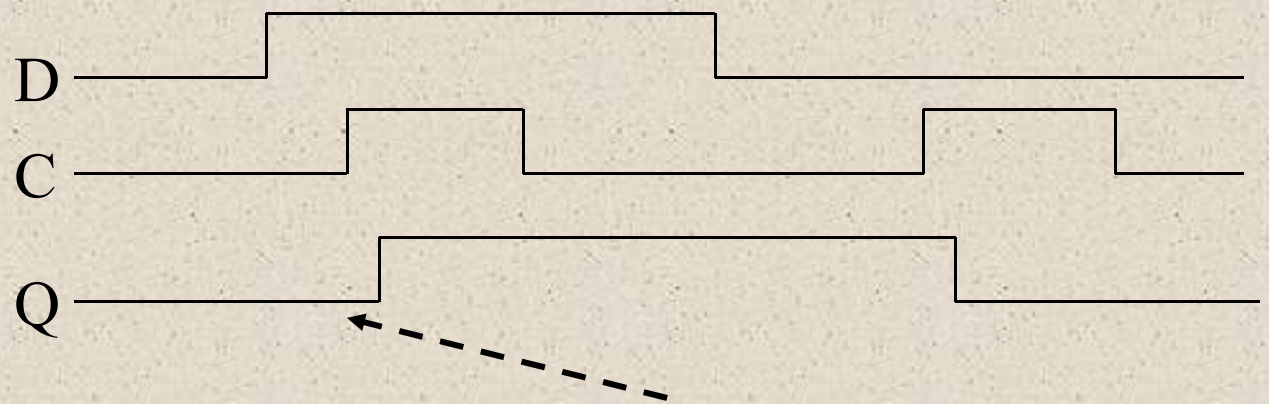
Výstupy: Q and $\bar{Q}$

Taktovaný SR Latch



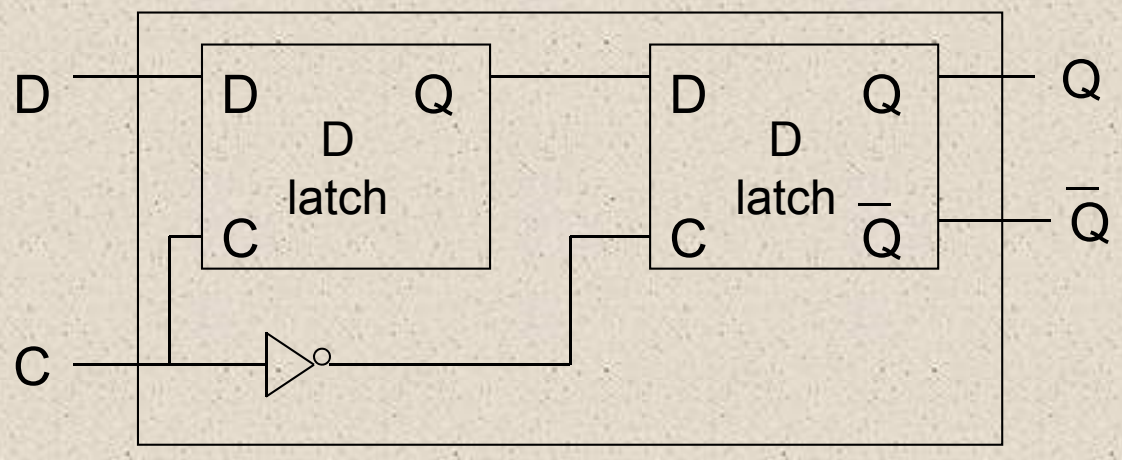
Taktovaný D Latch

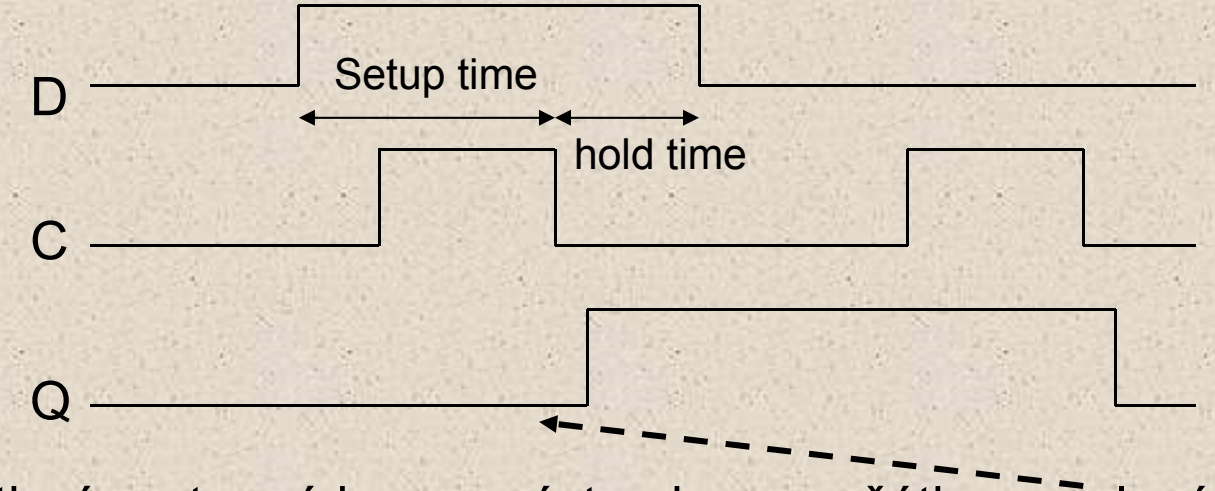




Výstup je na počátku vynulován

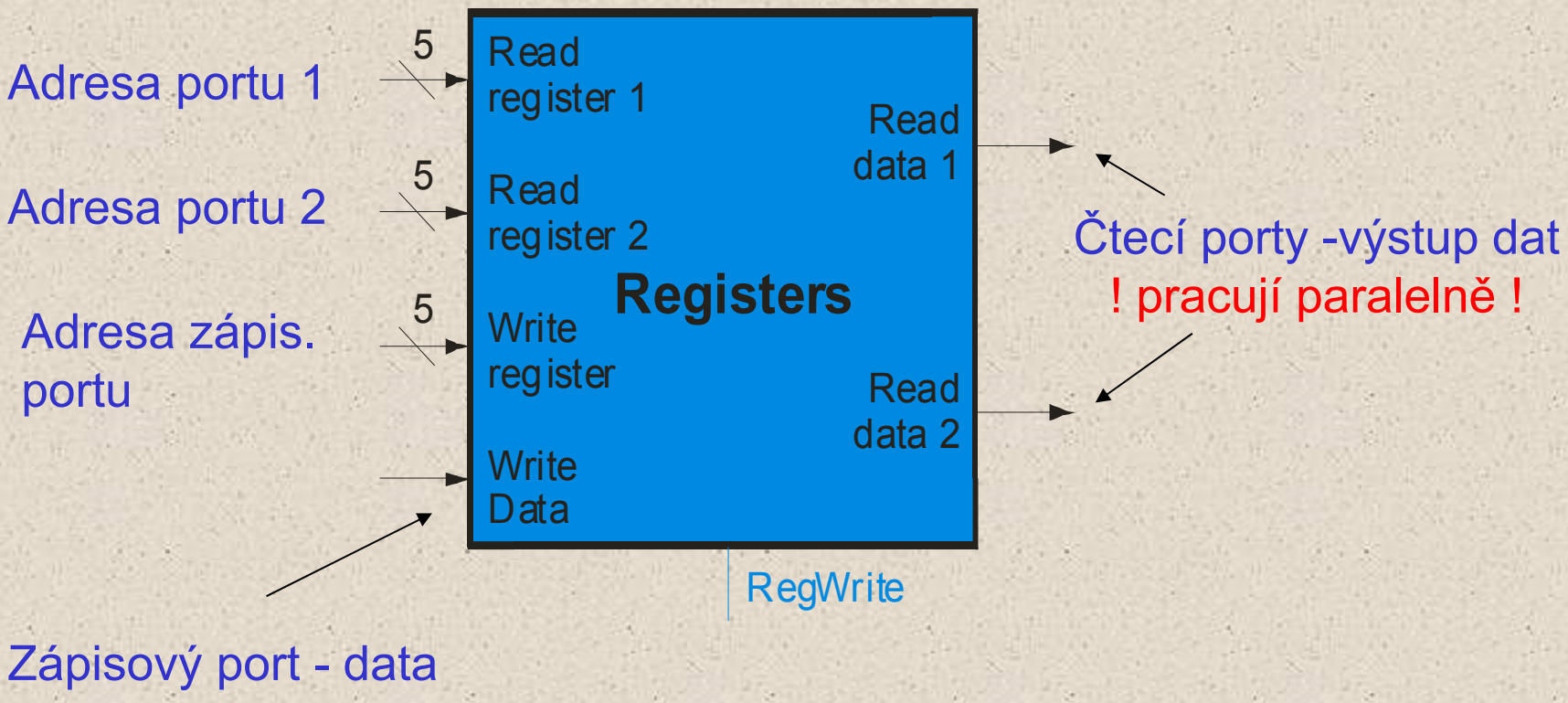
Klopný obvod typu D



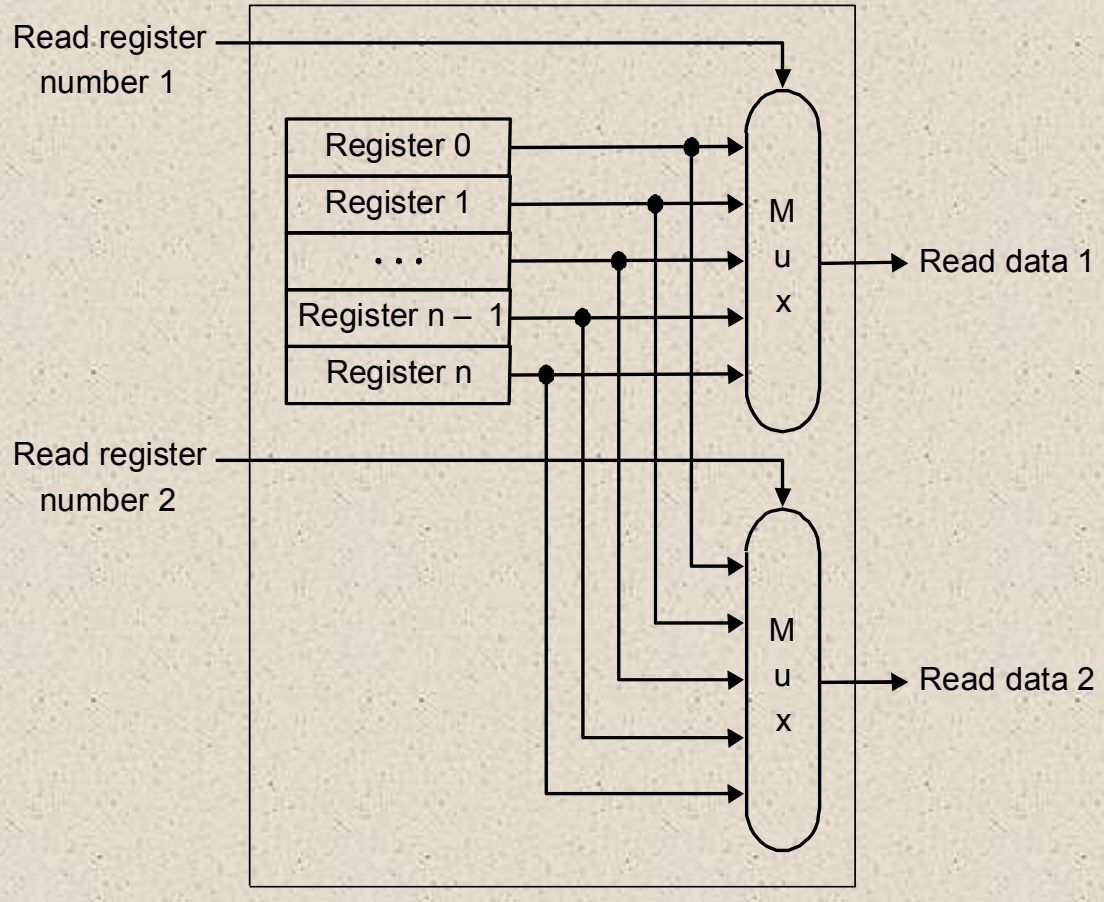


Aktivní sestupná hrana, výstup je na počátku vynulován

Registrový soubor

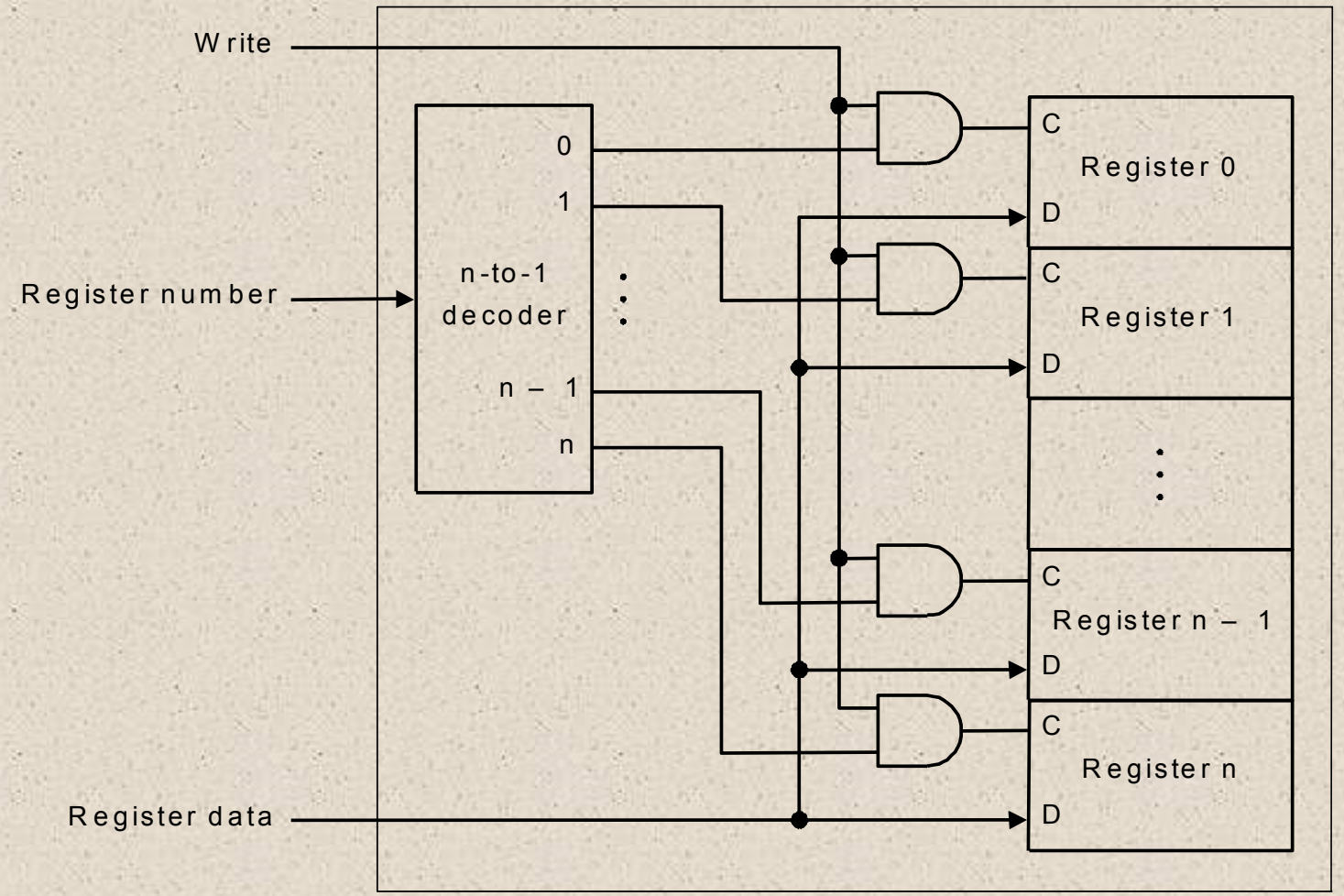


Čtecí porty registrového souboru



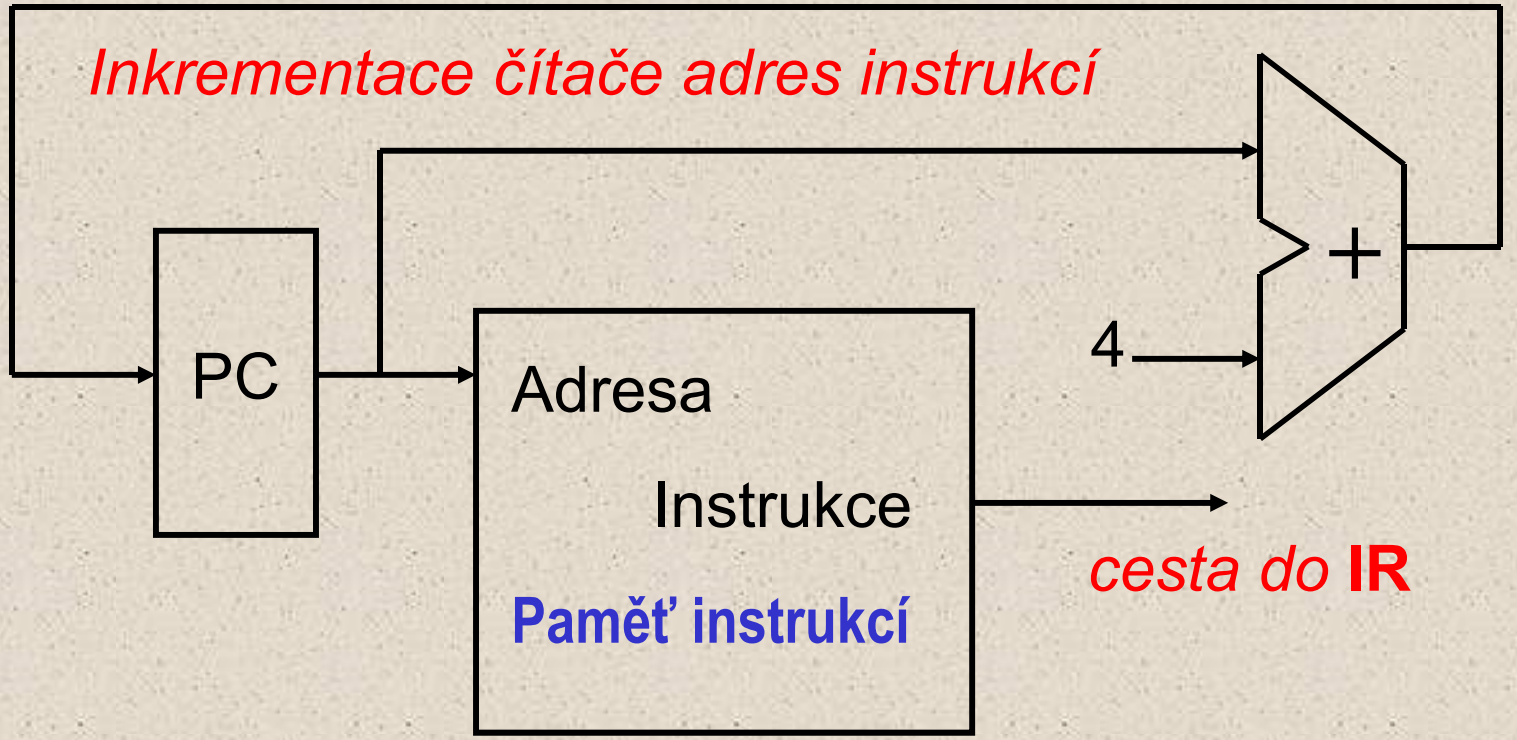
Implementace čtecích portů

Zápisové porty registrového souboru



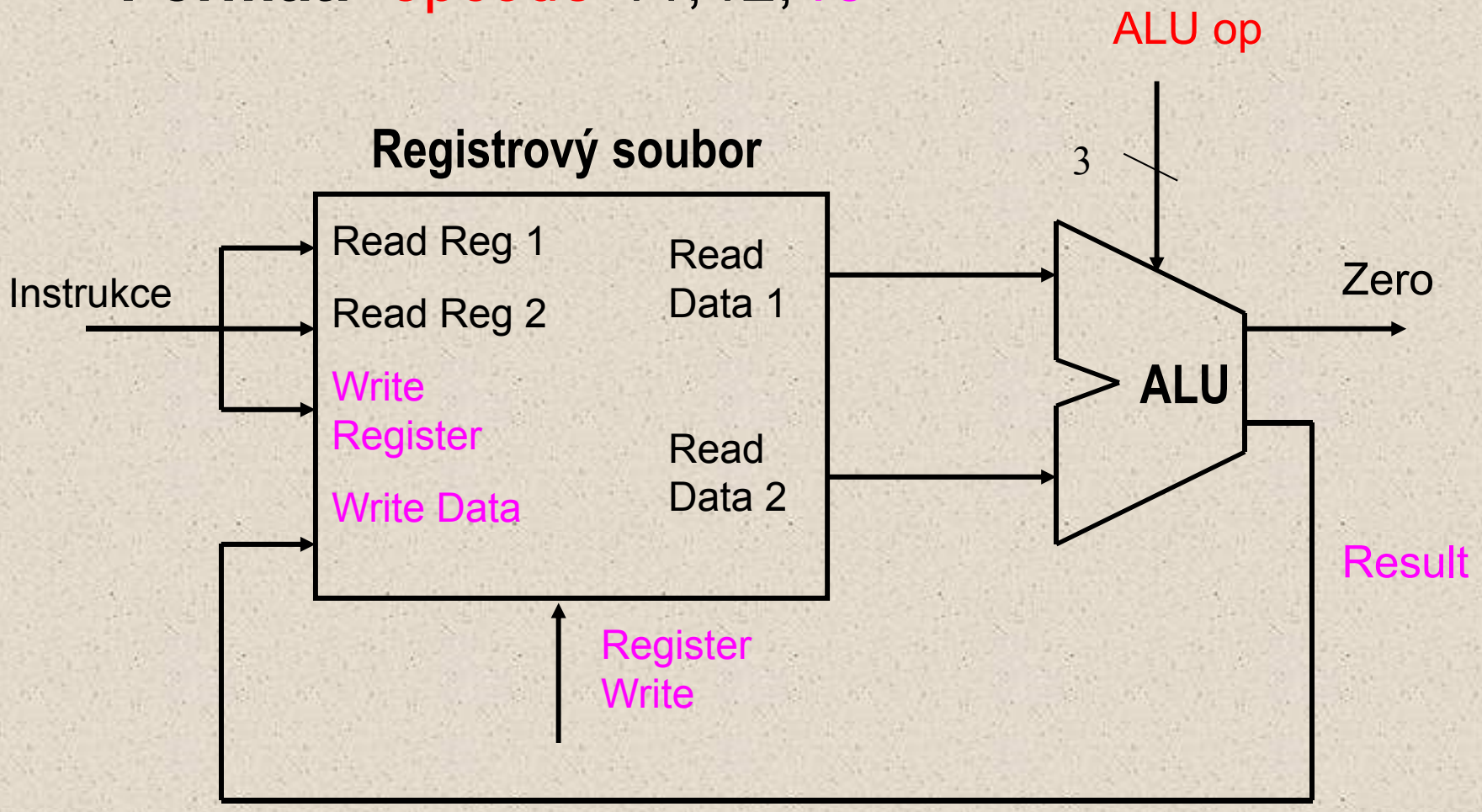
Základní datové cesty

- **Paměť** obsahuje instrukce
- **PC** adresa aktuální instrukce
- **ALU** provádí aktuální instrukci



Datové cesty pro R-formát

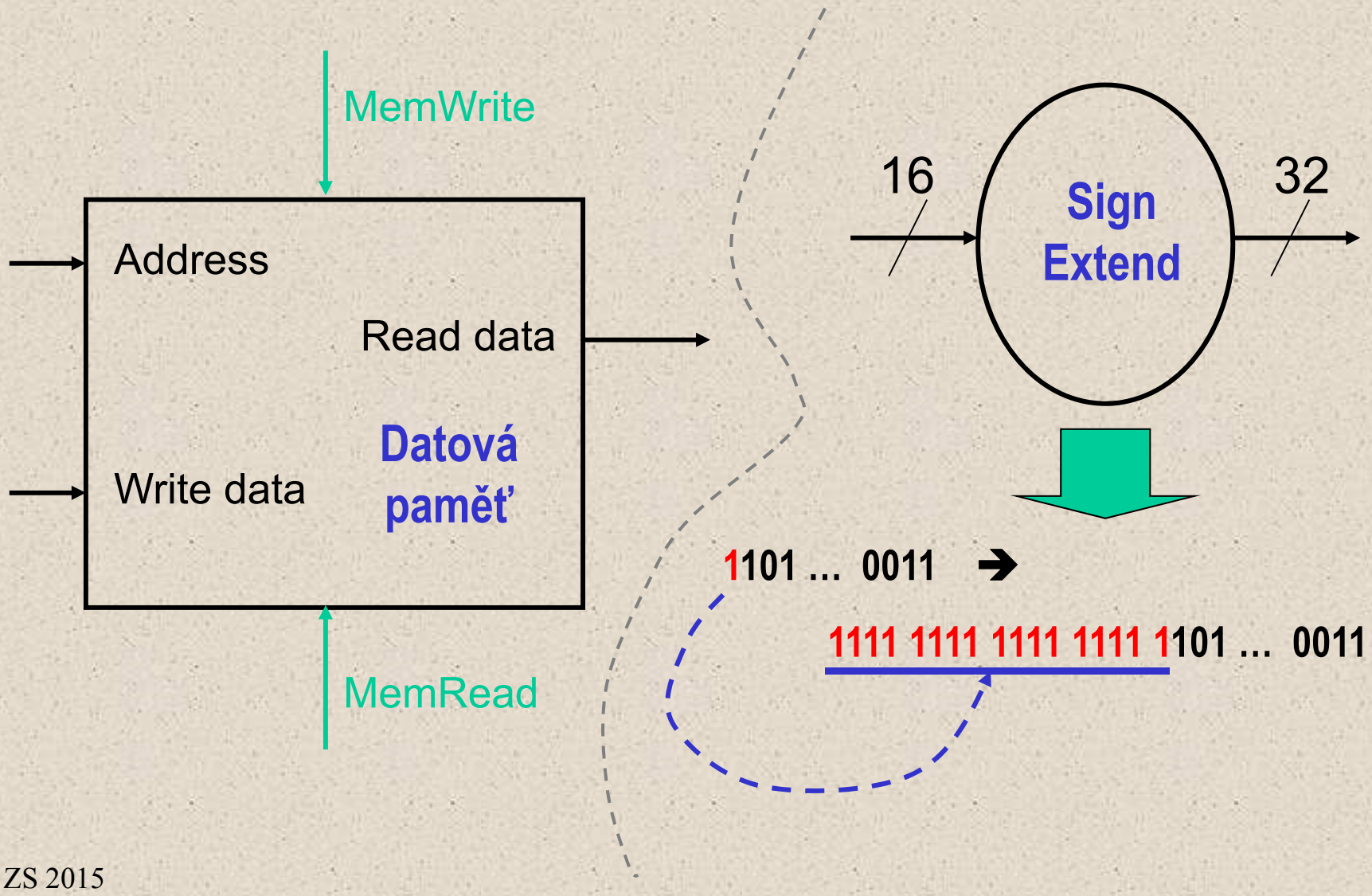
- **Formát:** opcode r1, r2, r3



Elementární kroky při Load/Store

- **lw $t1, offset($t2)**
  - reference do paměti, báze v $t2 s offsetem
  - **lw**: čtení paměti, zápis do registru $t1
  - **sw**: čtení z registru $t, zápis do paměti
- Výpočet adresy – podle ISA:
  - 16-bitový offset se znaménkem se převede na 32-bitovou hodnotu se znaménkem
- *Hardware*: Datová paměť pro read/write

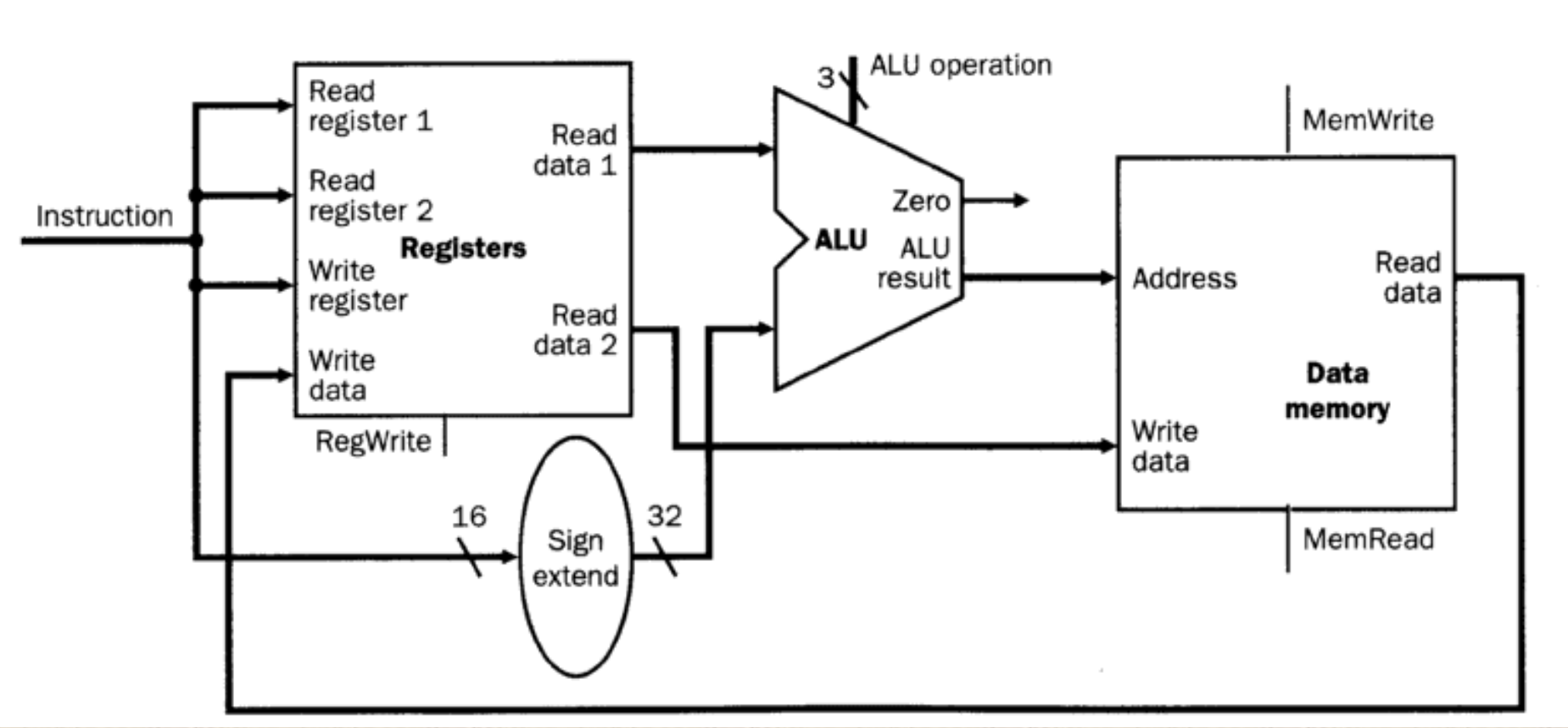
Prvky datové cesty pro Load/Store



Provedení operace Load/Store

1. **Přístup do registru** — Registrový soubor
   -- ***Čtení*** instrukce/dat/adresy
2. **Výpočet adresy do paměti** — ALU
   -- ***Dekódování*** adresy
3. **Čtení/zápis z/do paměti** — Datová paměť
4. **Zápis do registrového souboru** — Registrový soubor
   -- ***Provedení*** instrukce Load/Store

Datové cesty pro Load/Store



Fetch **Decode** **Execute**

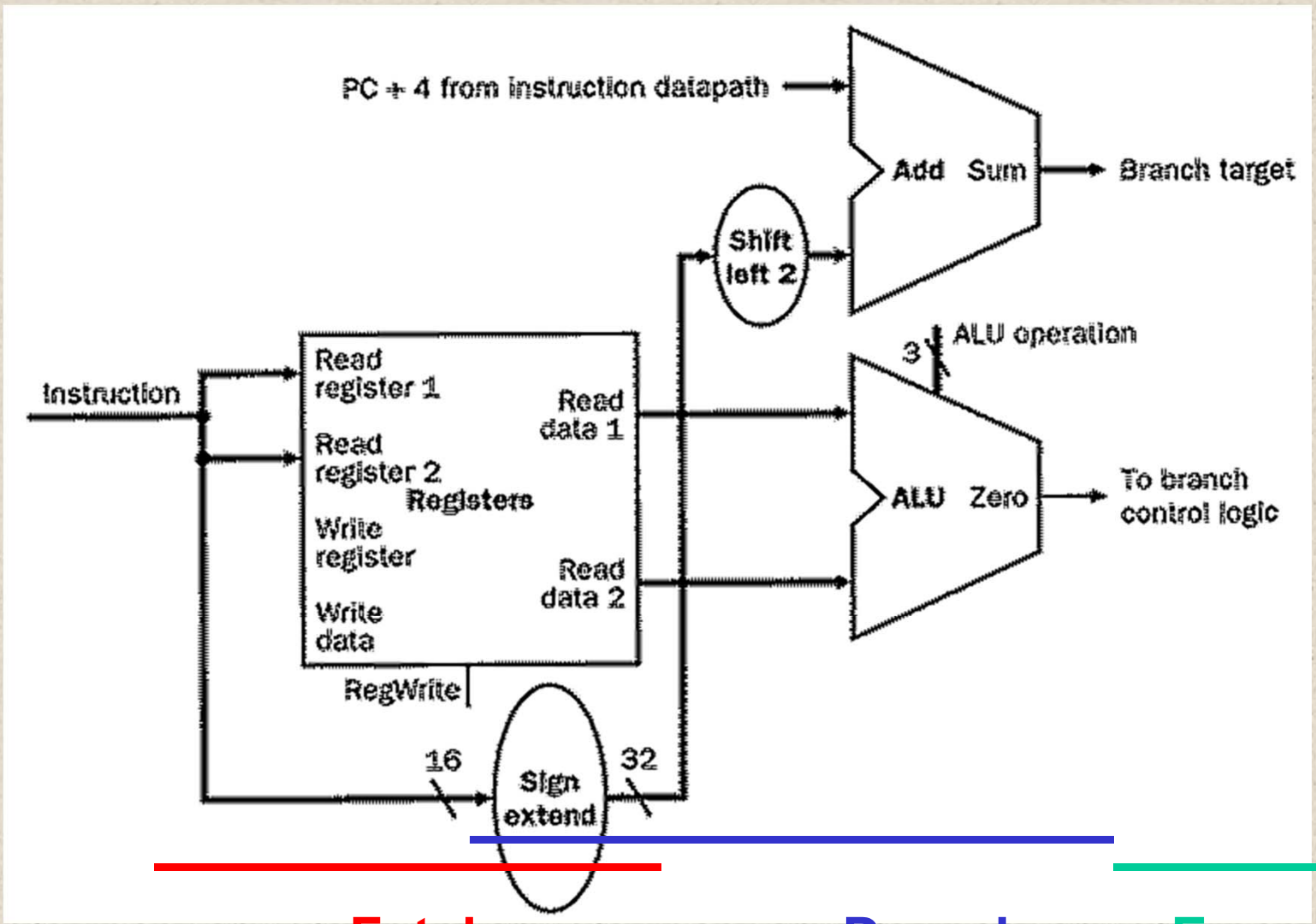
Datové cesty - instrukce větvení (Branch)

- **beq $t1, $t2, offset**
  - Dva registry ($t1, $t2) – test na rovnost
  - 16-bitový offset výpočet cílové adresy skoku
- Adresa cíle skoku – podle ISA:
  - Přičti znaménkem rozšířený offset k PC
  - Bázová adresa je instrukce za skokem (PC+4)
  - Posuň offset vlevo o 2 bity => word offset
- Skok na cílovou adresu
  - Nahraď dolních 26 bitů PC dolními 26 bity instrukce posunuté o 2 bity

Provedení větvení (Branch)

1. **Přístup do registru** — Registrový soubor
   -- ***Čtení*** instrukce/dat
2. **Vyhodnocení podmínky skoku** — ALU #1
3. **Výpočet cílové adresy** — ALU #2
   -- Výpočet skoku – podobné ***dekódování***
4. **Skok na cílovou adresu** — Řadič
   -- ***Provedení*** instrukce větvení

Datové cesty pro větvení (Branch)



Opožděné podmíněné skoky (MIPS)

- **MIPS ISA**: Podmíněné skoky jsou vždy *opožděné*
  - Instrukce I_b ležící za skokem se vždy provede
  - *condition* = *false* => normální skok
  - *condition* = *true* => I_b se provede

Je to špatně?

1. Zlepší se efektivita
2. Skok se neprovádí (nesplněná podmínka) může být *častější případ*

Závěr

- Datové cesty zajišťují přesuny dat v CPU
- Datové cesty propojují:
  - *ALU*: Provádí elementární operace
  - *Registrový soubor*: Čtení a zápis dat
- Registrový soubor je implementován pomocí D klopných obvodů, multiplexerů a dekodérů
  - Taktované (synchronní) logické obvody
  - Řídící signály určují zápis do registrů
  - Datové přenosy z registrů nejsou chráněné

Závěr

- **MIPS ISA**: Tři instrukční formáty (R, I a J)
- Datové cesty navrženy pro každý instrukční formát
- **Stavební prvky datových cest:**

  -- *R-formát:* ALU, registrový soubor

  -- *I-formát:* Sign Extender, datová paměť

  -- *J-formát:* ALU #2 pro výpočet cílové adresy

Trik: Opožděné skoky zlepšují efektivitu