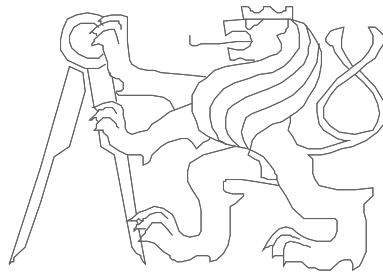


Architektura počítačů

04

Zřetězené vykonávání instrukcí; Hazardy;
Vyvažování stupňů zřetězení a časování; Superzřetězení



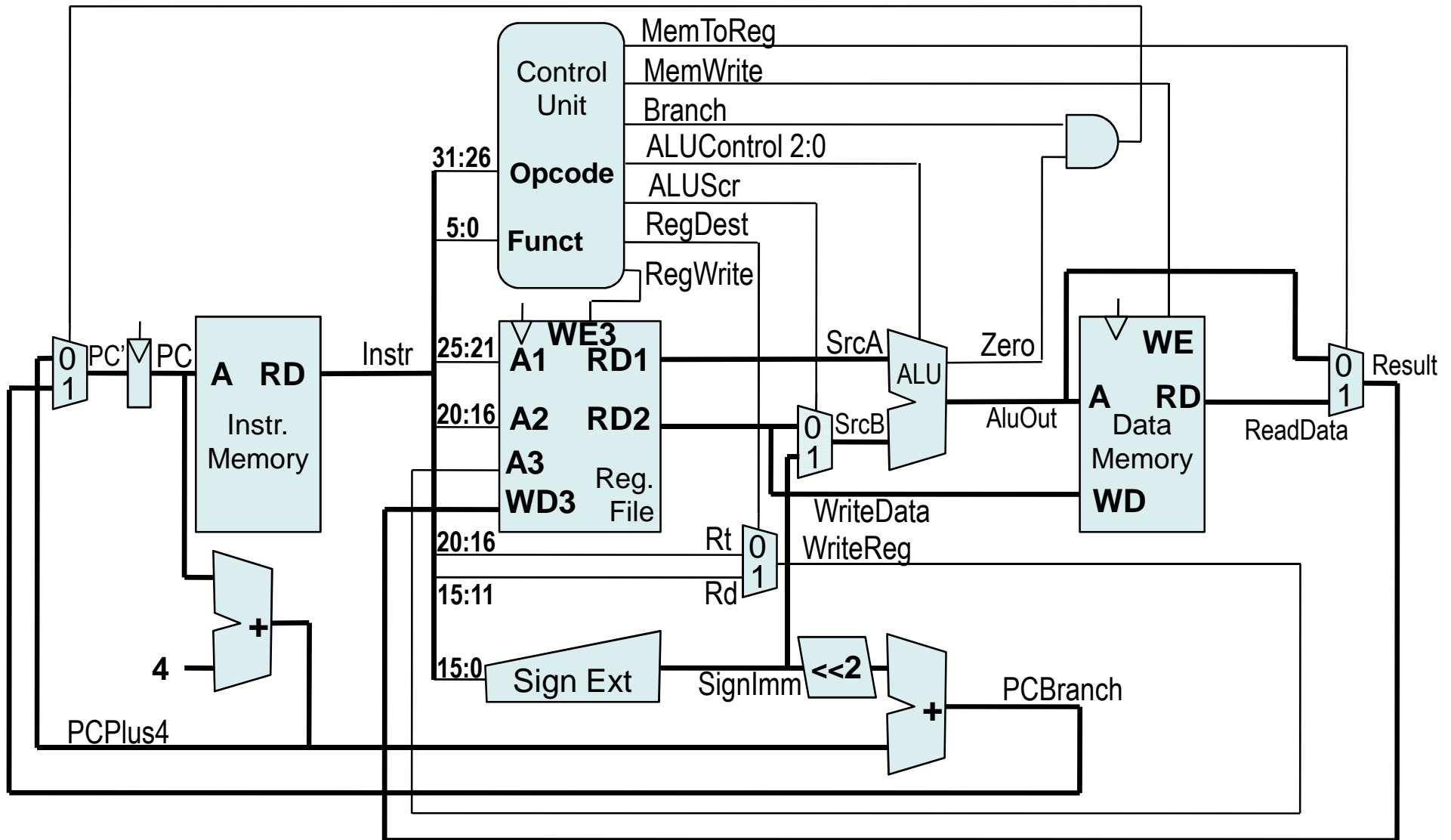
České vysoké učení technické, Fakulta elektrotechnická

Úkol dnešní přednášky

- Modifikujeme jednocyklový procesor z 2. přednášky na zřetězený procesor.
- Procesor bude podporovat instrukce:
`add, sub, and, or, slt, addi, lw, sw a beq`

Typ	31...					0
R	opcode (6), 31:26	rs (5), 25:21	rt (5), 20:16	rd (5), 15:11	shamt (5)	funct (6), 5:0
I	opcode (6), 31:26	rs (5), 25:21	rt (5), 20:16	immediate (16), 15:0		
J	opcode (6), 31:26	address (26), 25:0				

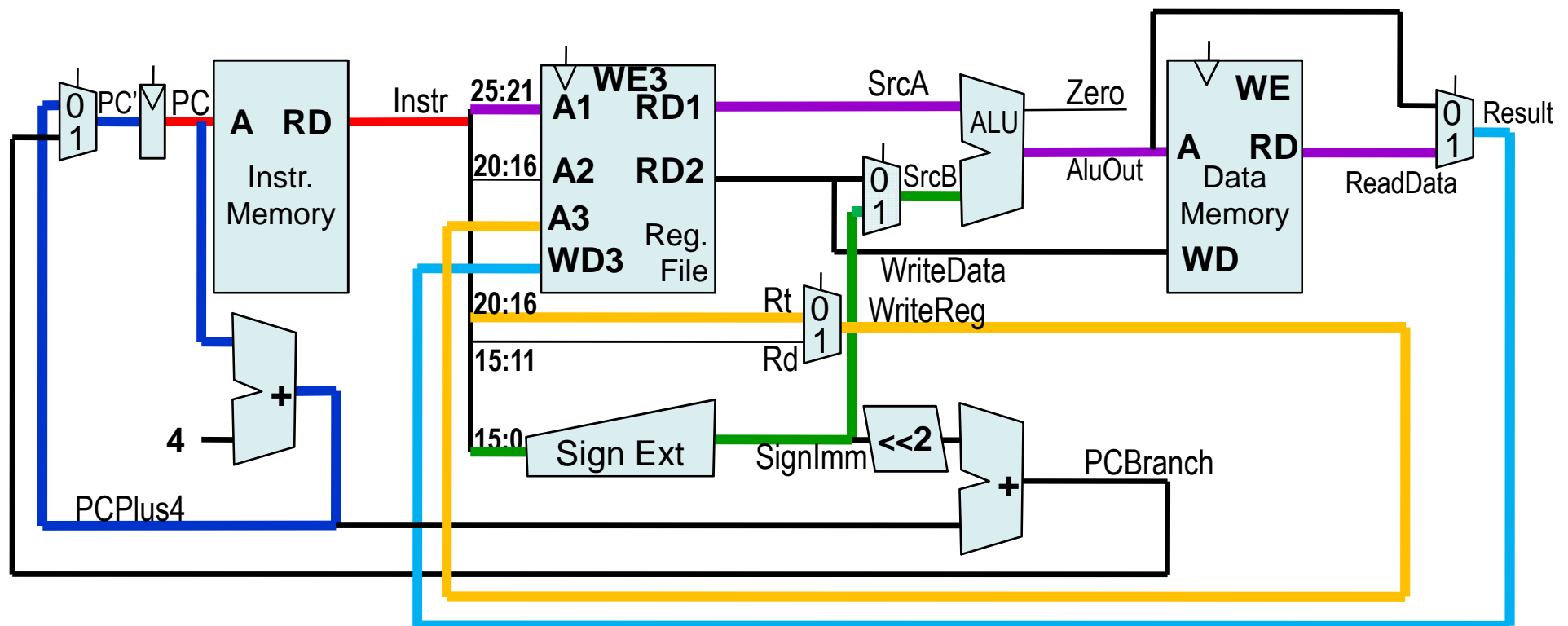
Jedno-cyklový procesor spolu s pamětmi



Jedno-cyklový procesor – výkon: $IPS = IC / T = IPC_{str} \cdot f_{CLK}$

- Jaká může být maximální frekvence procesoru?
- Zpoždění na kritické cestě – instrukce lw:

$$T_C = t_{PC} + t_{Mem} + t_{RFread} + t_{ALU} + t_{Mem} + t_{Mux} + t_{RFsetup}$$



Jedno-cyklový procesor – výkon: $IPS = IC / T = IPC_{str} \cdot f_{CLK}$

- $T_c = t_{PC} + t_{Mem} + t_{RFread} + t_{ALU} + t_{Mem} + t_{Mux} + t_{RFsetup}$

- Předpokládejme:

$$t_{PC} = 30 \text{ ns}$$

$$t_{Mem} = 300 \text{ ns}$$

$$t_{RFread} = 150 \text{ ns}$$

$$t_{ALU} = 200 \text{ ns}$$

$$t_{Mux} = 20 \text{ ns}$$

$$t_{RFsetup} = 20 \text{ ns}$$

Pak $T_c = 1020 \text{ ns} \rightarrow f_{CLK \max} = 980 \text{ kHz}$,

$IPS = 1.980e3 = 980\,000$ instrukcí za sekundu

Zřetězené vykonávání instrukcí

Předpokládejme, že vykonání instrukce můžeme rozdělit do 5 stupňů:



IF – Instruction Fetch, ID – Instruction decode (and Operand Fetch),

EX – Execute, MEM – Memory Access, WB – Write Back

a dále $\tau = \max \{ \tau_i \}_{i=1}^k$, kde τ_i je čas šíření (*propagation delay*) v i -tém stupni.

IF – poslání PC do paměti a vybrání aktuální instrukce. Aktualizace PC = PC+4

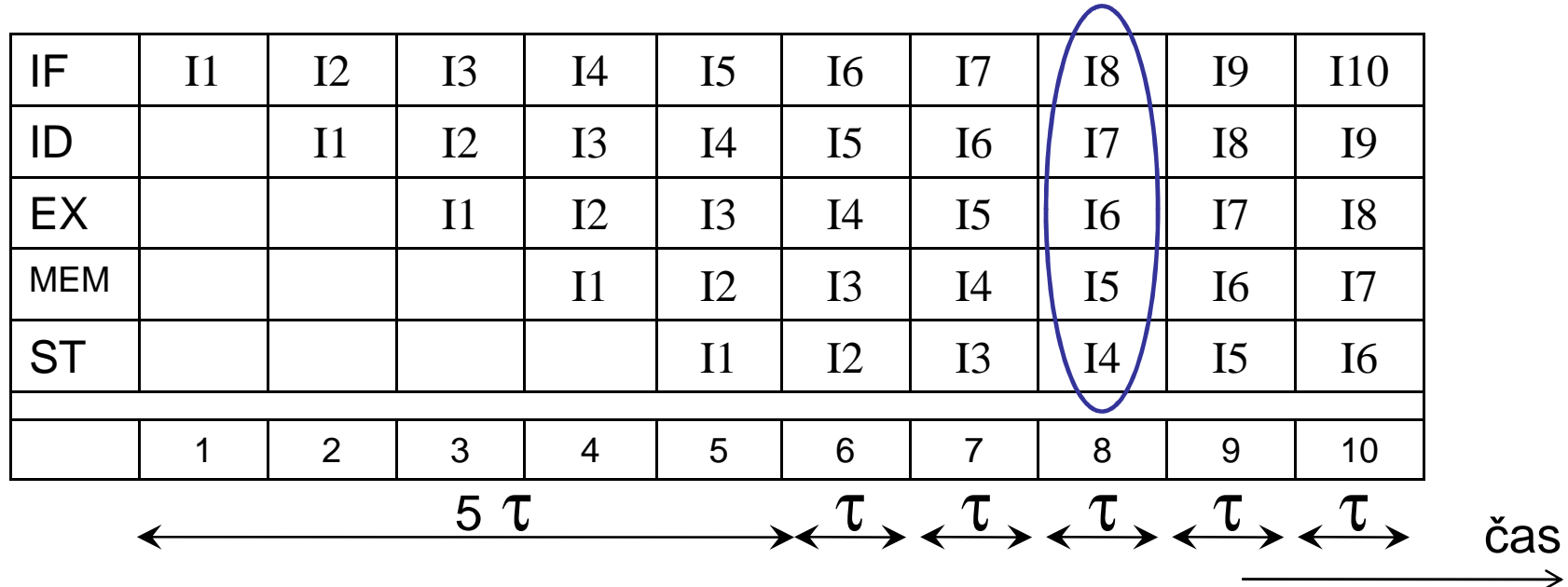
ID – dekódování instrukce a načtení registrů specifikovaných v instrukci, provedení testu na rovnost registrů (kvůli možnému větvení), znaménkové rozšíření offsetu, výpočet cílové adresy pro případ větvení (zn. rozš. offset + PC)

EX – operace ALU

MEM – v případě instrukce *load* /*store* – čtení/zápis do paměti

WB – v případě instrukcí typu register-register nebo instrukce *load* – zápis výsledku do RF (výsledek může přicházet z ALU nebo paměti)

Paralelizmus na úrovni instrukcí - zřetězení



- Čas vykonání n instrukcí k -stupňové pipeline:

$$T_k = k \cdot \tau + (n - 1) \tau$$

- Zrychlení: $S_k = \frac{T_1}{T_k} = \frac{nk\tau}{k\tau + (n-1)\tau} \quad \lim_{n \rightarrow \infty} S_k = k$

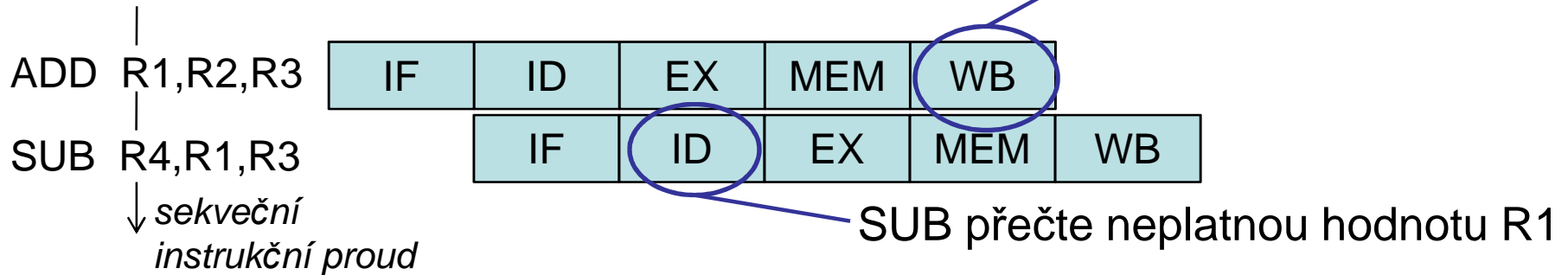
Předpoklad: ideálně vyvážená pipeline, obvod můžeme libovolně dělit

Paralelizmus na úrovni instrukcí - zřetězení

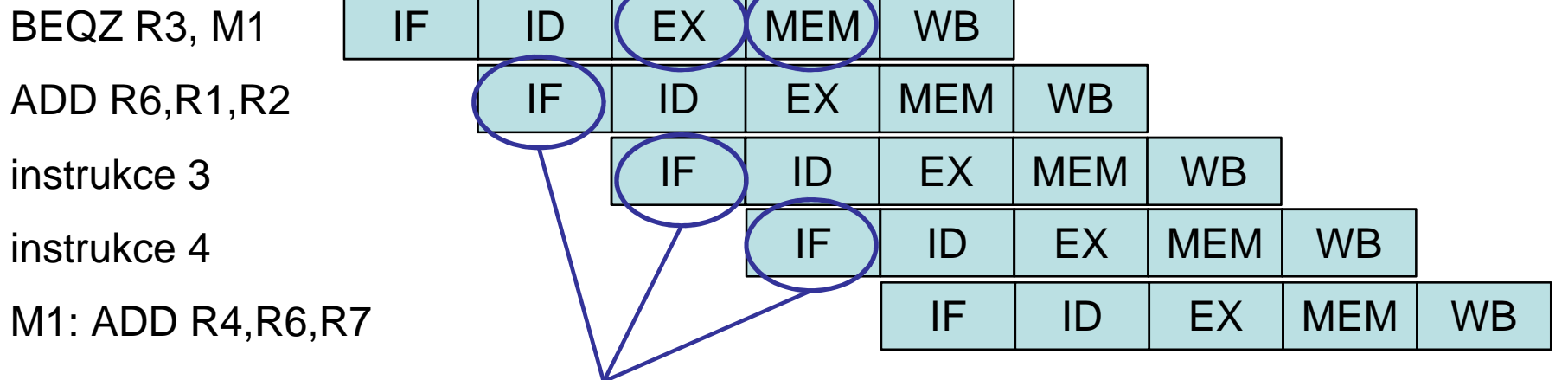
- Neredukuje čas vykonání individuální instrukce, právě naopak..
- Hazardy:
 - Strukturální (řešeny duplikací),
 - datové (důsledek datových závislostí: RAW, WAR, WAW) a
 - řídící (instrukce měnící PC)...
- Hazardy způsobují (mohou způsobovat) pozastavení vykonávání (stall) nebo vyprázdnění pipeline
- Pozn. : Hlubší pipeline (více stupňů) znamená méně hradel v každém stupni a tím pádem možnost zvýšit pracovní frekvenci procesoru.., avšak více stupňů znamená i vyšší režii (nutnost lépe řadit instrukce do pipeline)

Paralelizmus na úrovni instrukcí - Narušení sémantiky

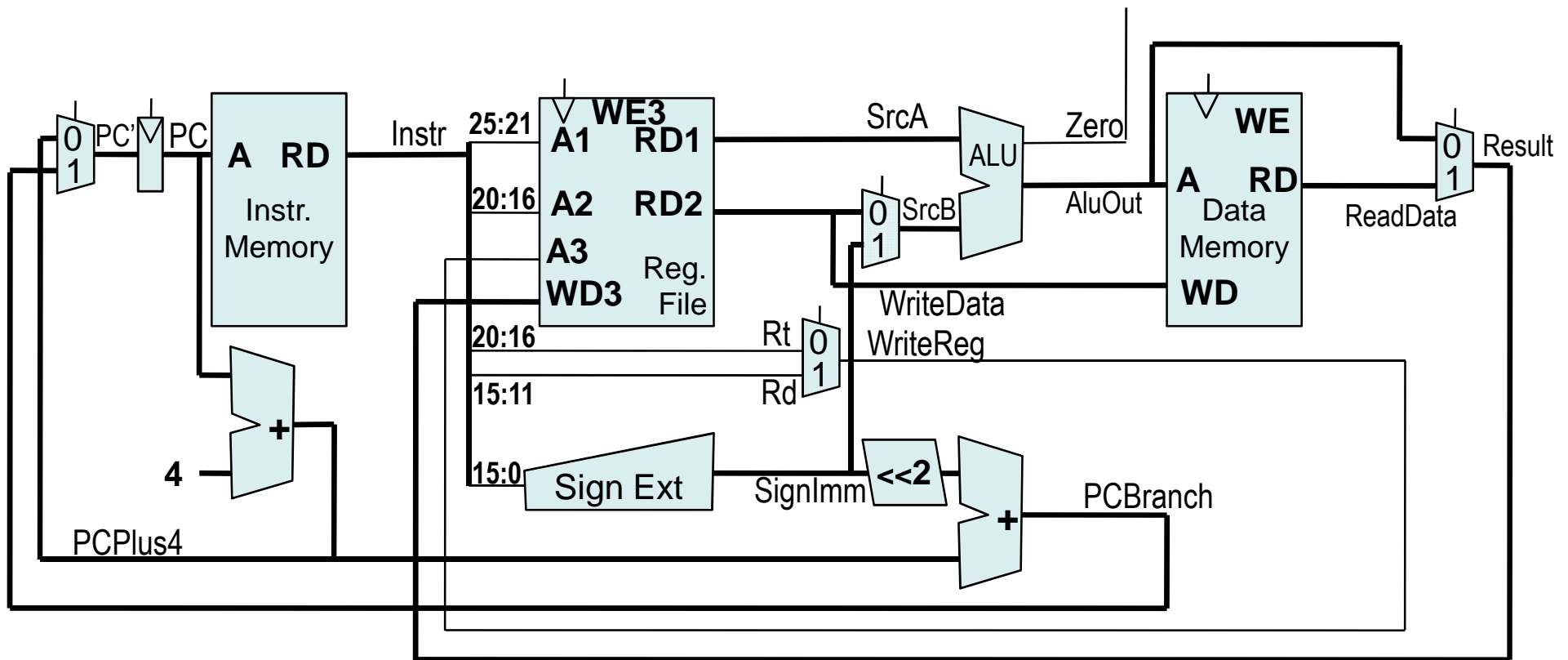
Datový hazard:



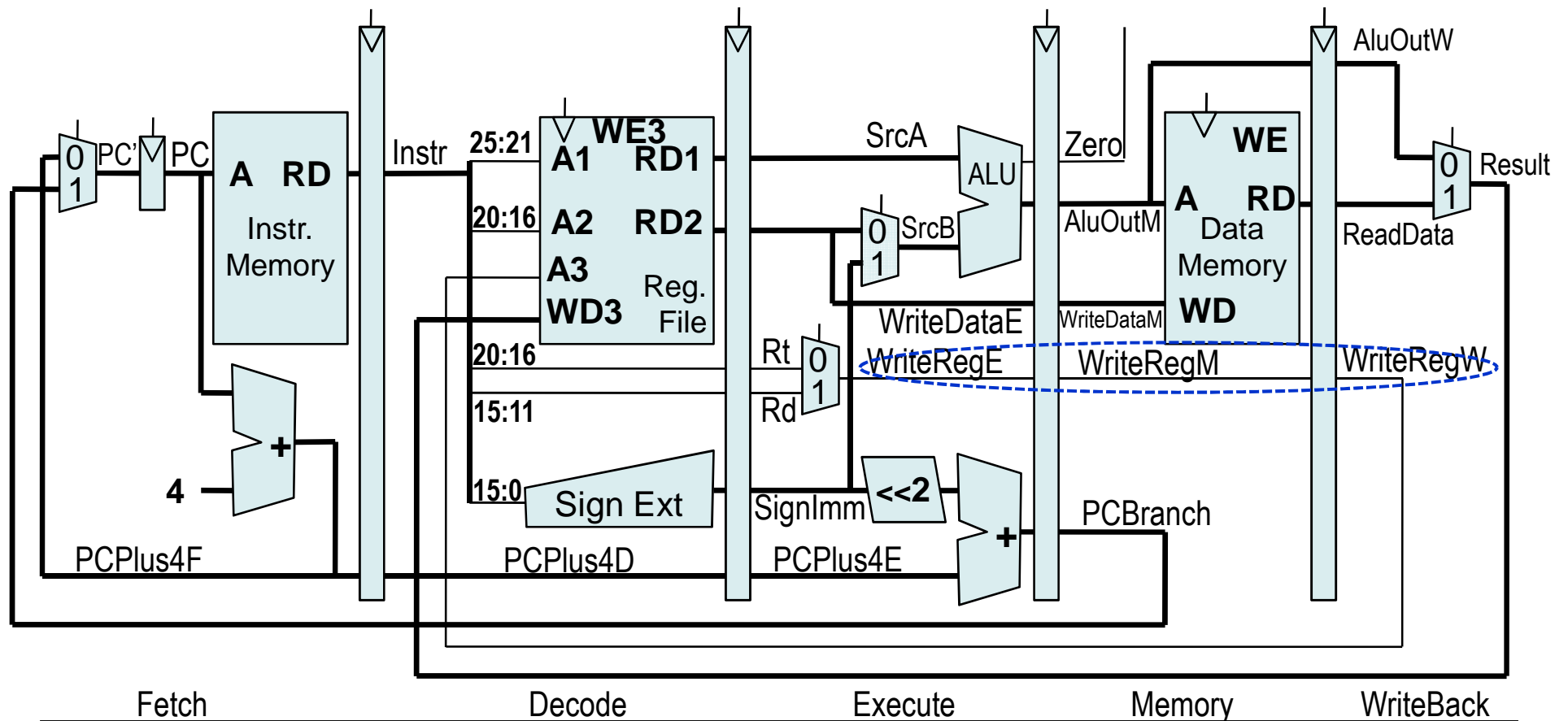
Řídicí hazard:



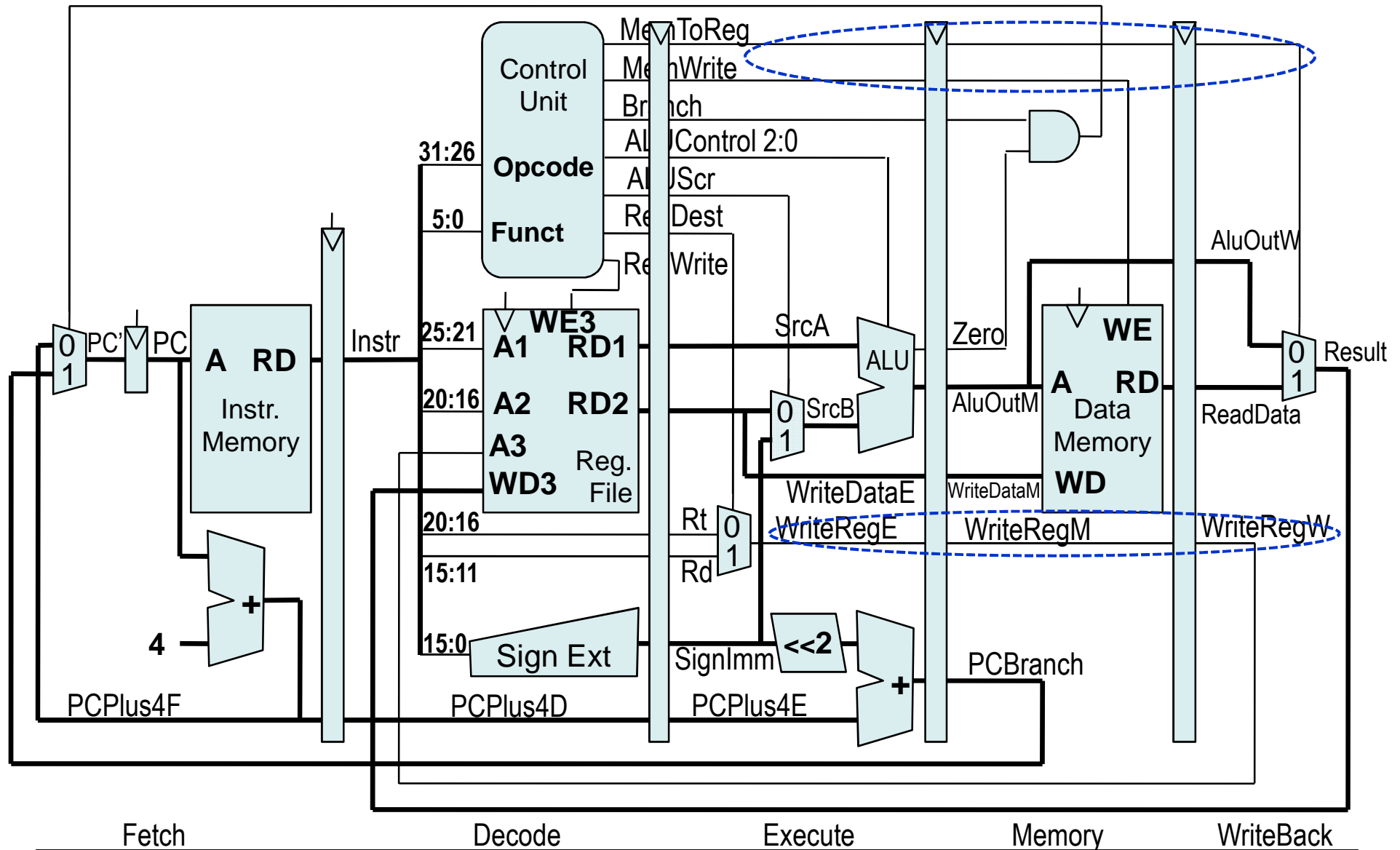
Nezřetězené vykonávání



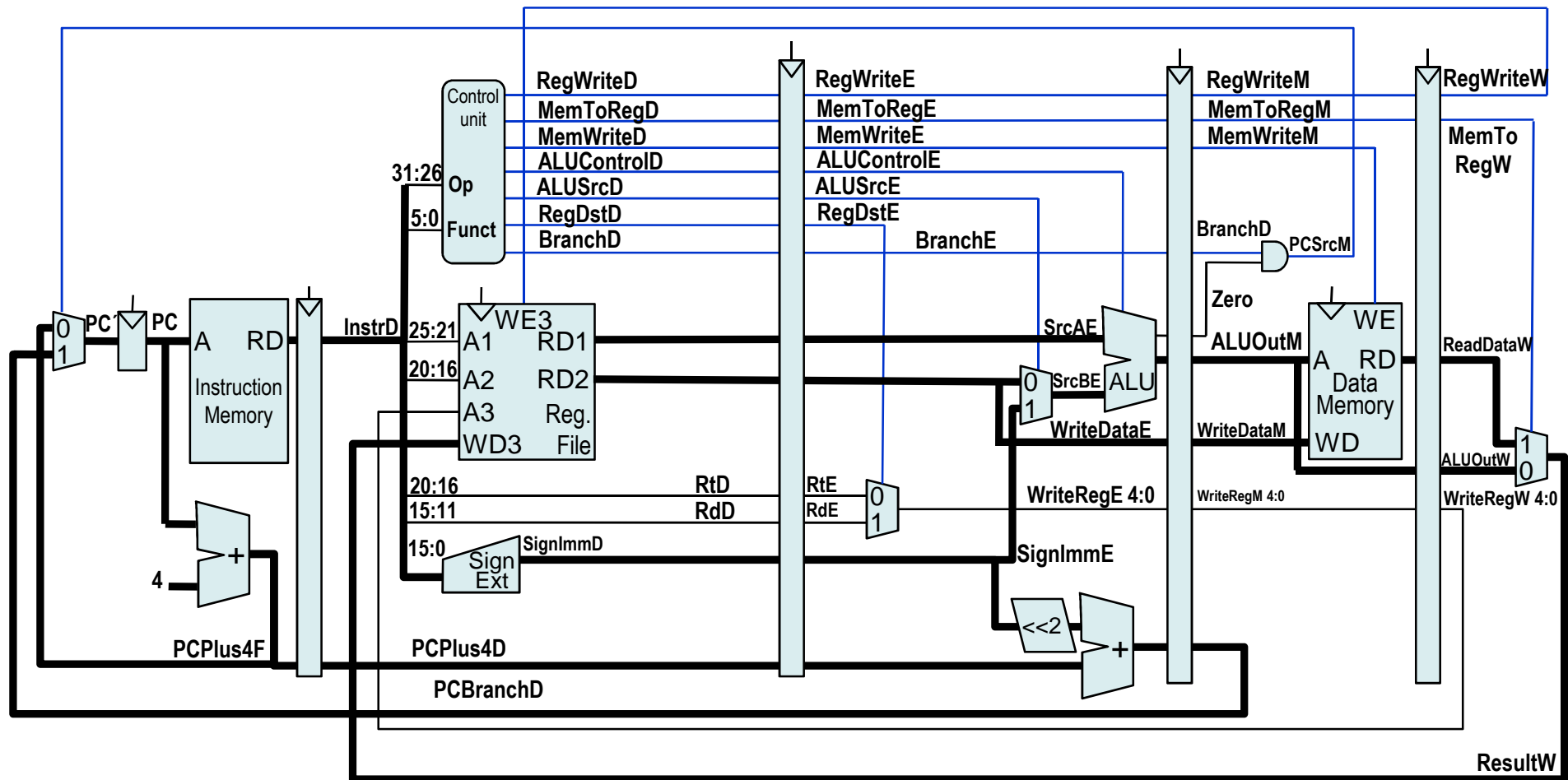
Zřetěžené vykonávání



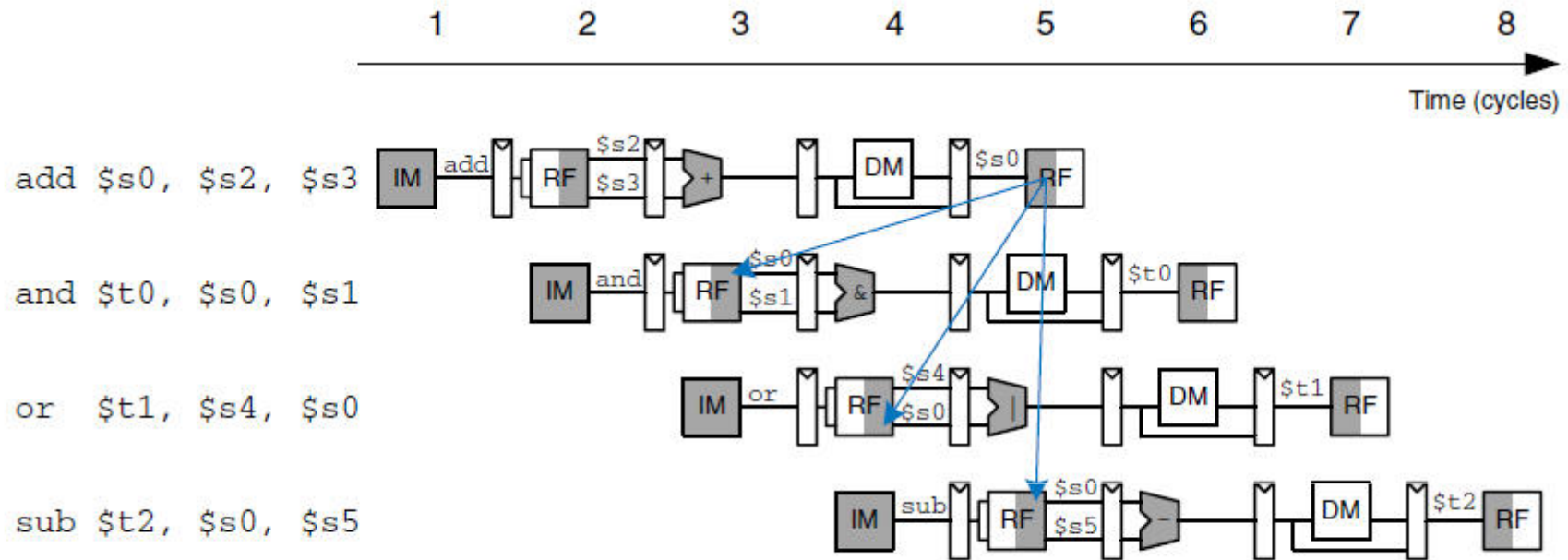
Zřetěžené vykonávání



Totéž, pouze zmenšeno a překresleno...

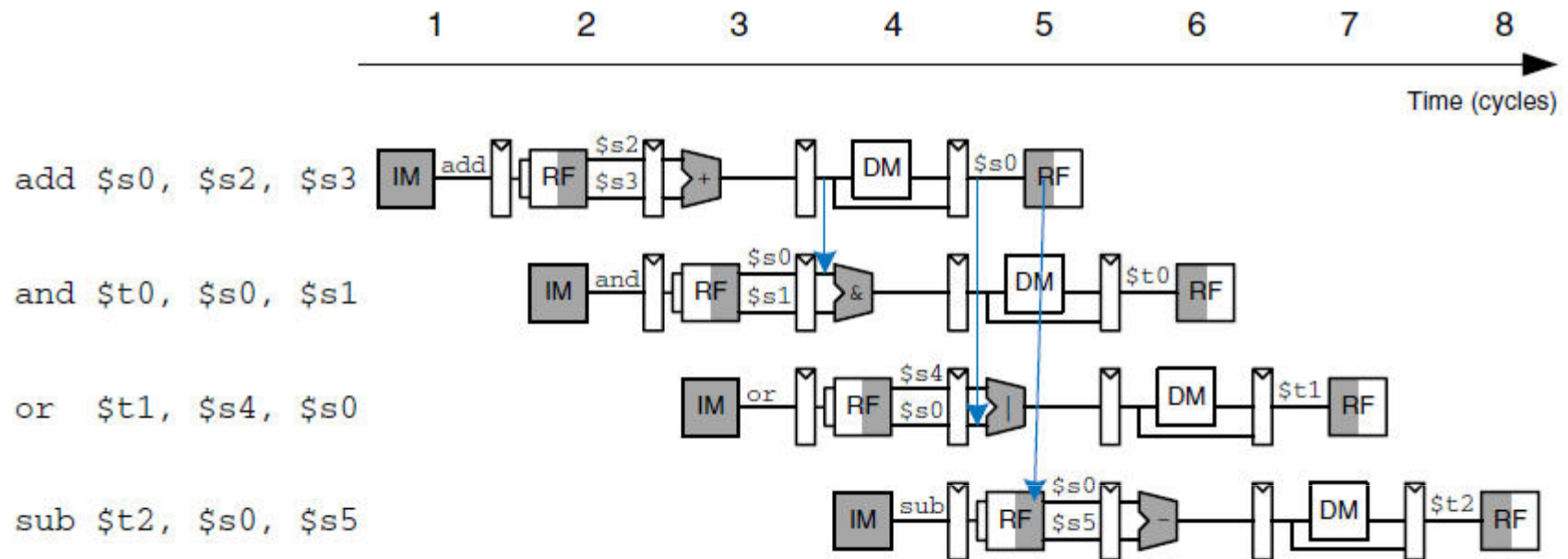


Vznik datových hazardů



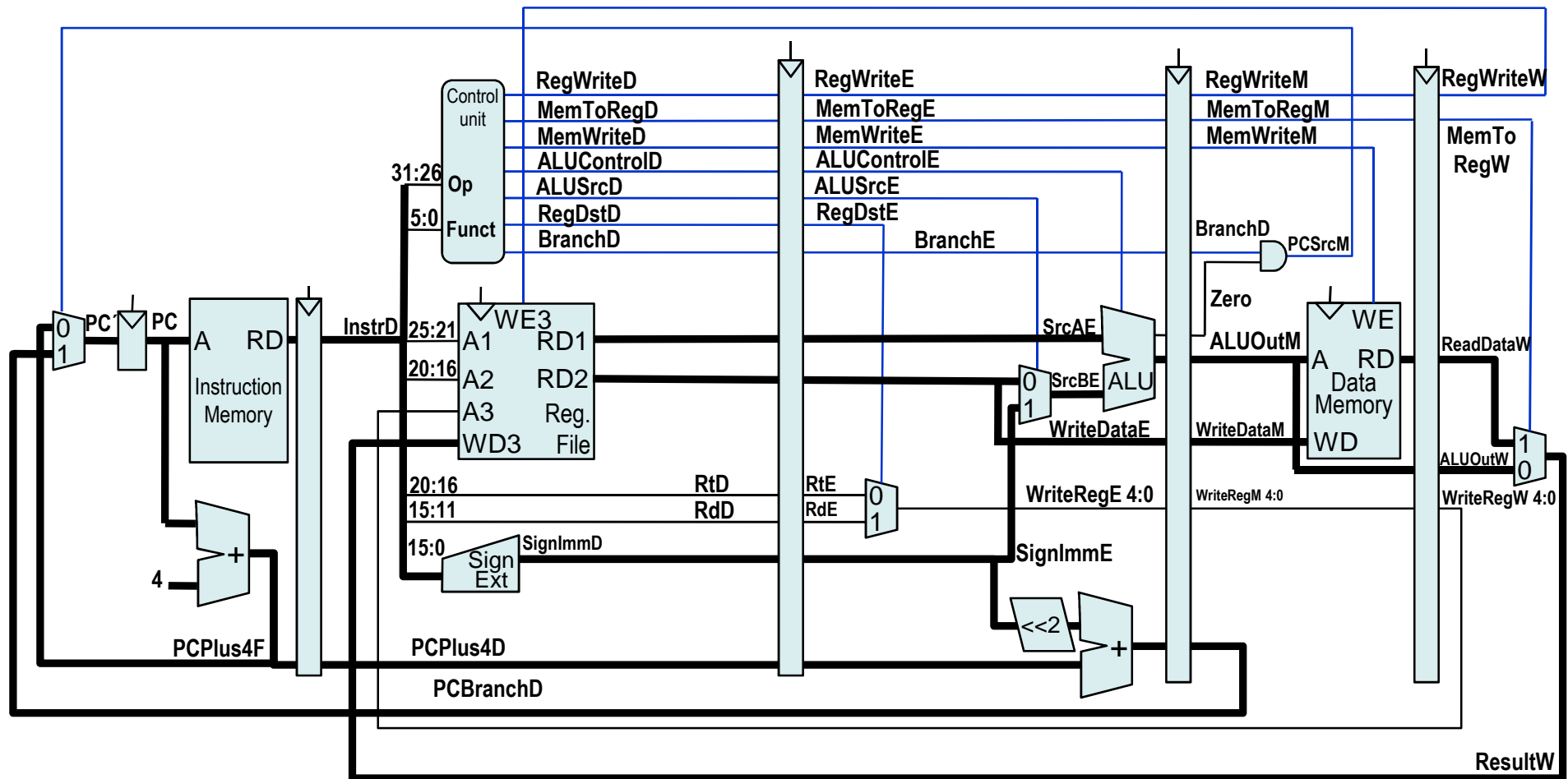
- Pracovní registry (Register File) – přístup v dvou fázích (Decode, WriteBack) – zápis v první polovině cyklu, čtení ve druhé..
- RAW hazard...
- Jak je možné řešit tento hazard a nedegradovat výkon pipeline?

Řešení datových hazardů přeposíláním (forwarding)

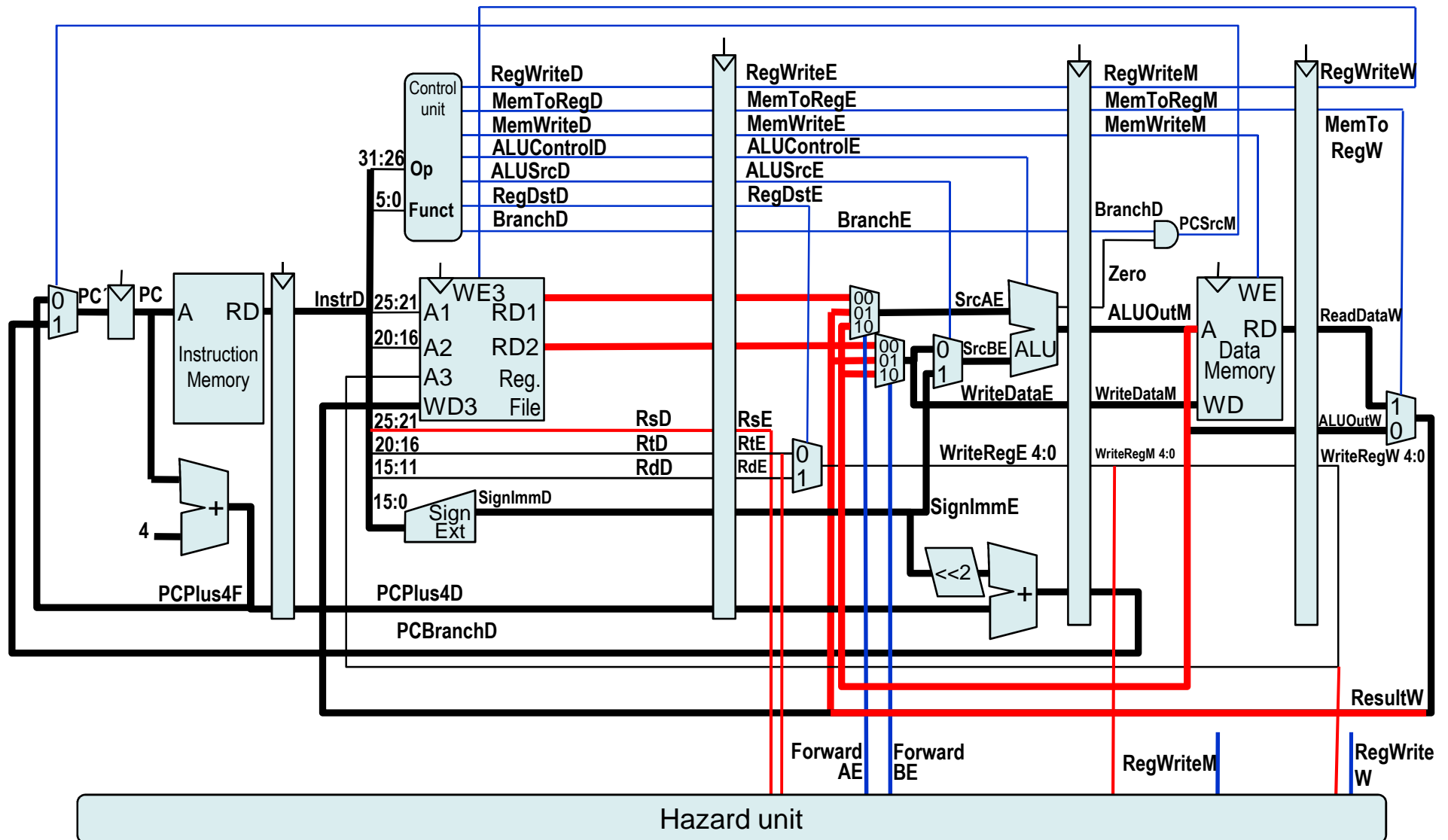


- Pokud výsledek vzniká dříve než jej následující instrukce skutečně potřebují je možné tento hazard řešit přeposíláním (forwarding)
- nastává když se použité zdrojové registry instrukce ve stupni E shodují s cílovým registrem ve stupni M nebo WB
- proto musejí být čísla těchto registrů z těchto stupňů posílána do Hazard Unit
- taktéž zda cílový registr bude skutečně použit k zápisu (jinými slovy, zda se skutečně jedná o cílový registr – lw vs. sw) – RegWrite z M a WB

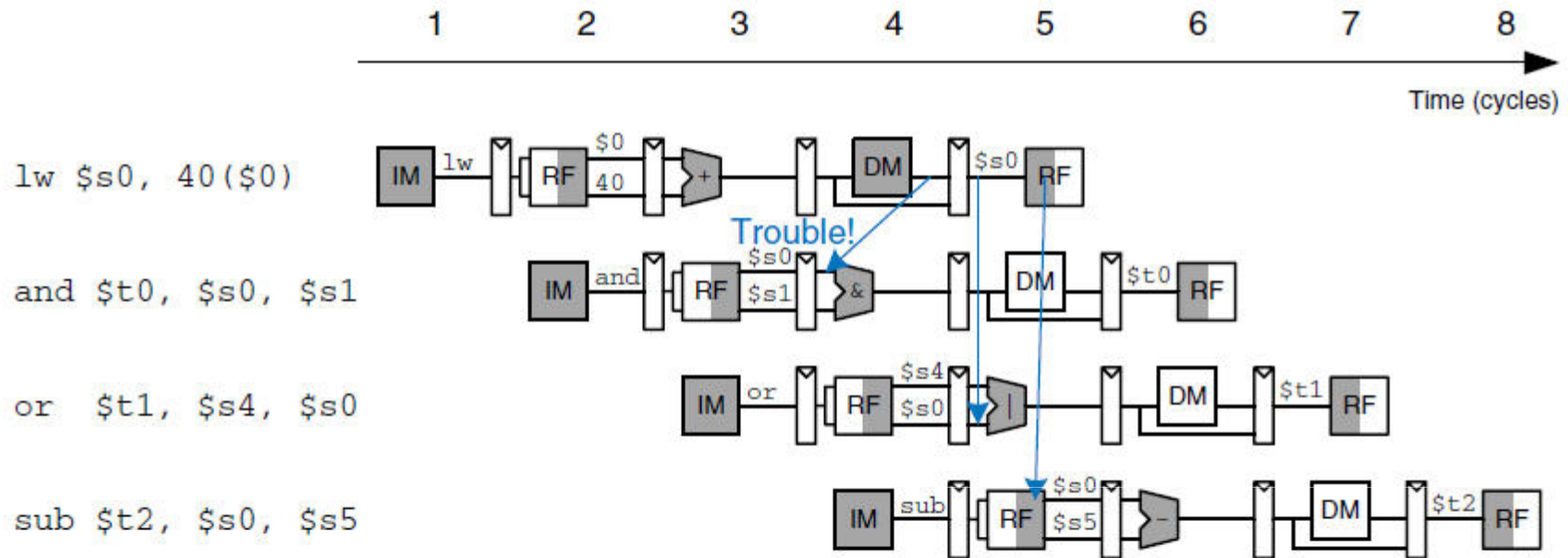
Stávající procesor



Řešení datových hazardů přeposíláním (forwarding)

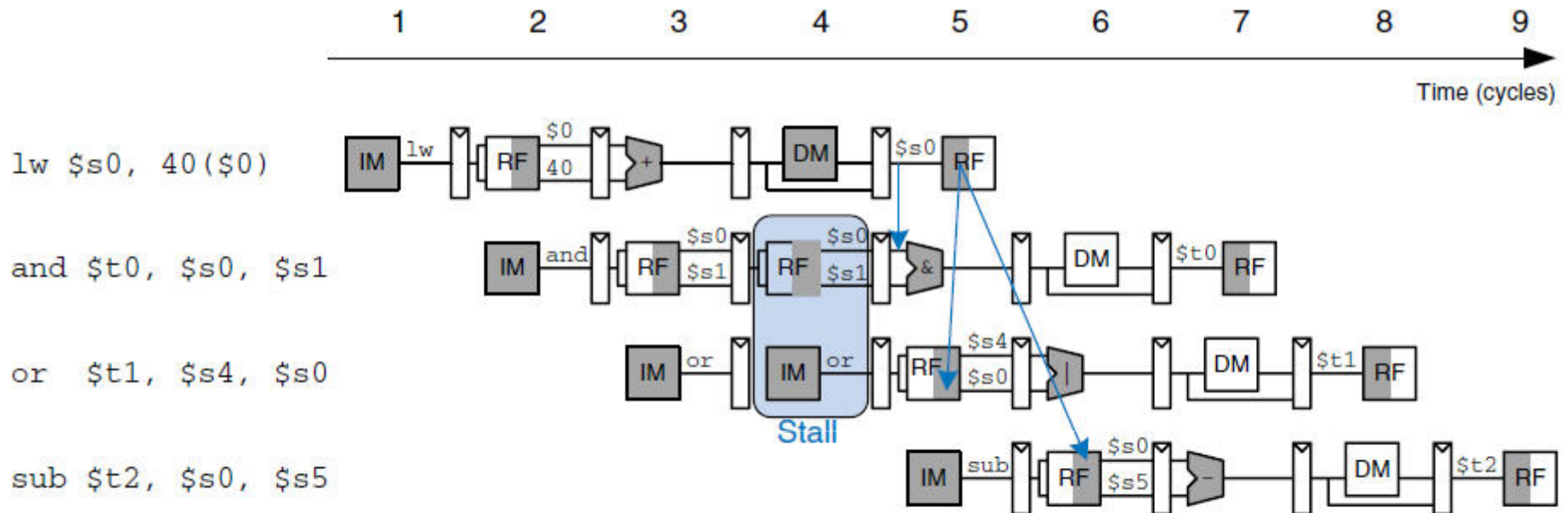


Řešení datových hazardů pozastavením (stall)



- Pokud následující instrukce potřebují výsledek dřív než skutečně vzniká je možné tento hazard řešit pozastavením (stall)
- Pozastavení pipeline je prostředkem řešení hazardů; nezvyšuje však propustnost systému
- stupně pipeline předcházející stupni kde hazard vzniká jsou pozastaveny do doby, než jsou k dispozici výsledky požadované následujícími instrukcemi – ty jsou pak přeposílány (forwarding)

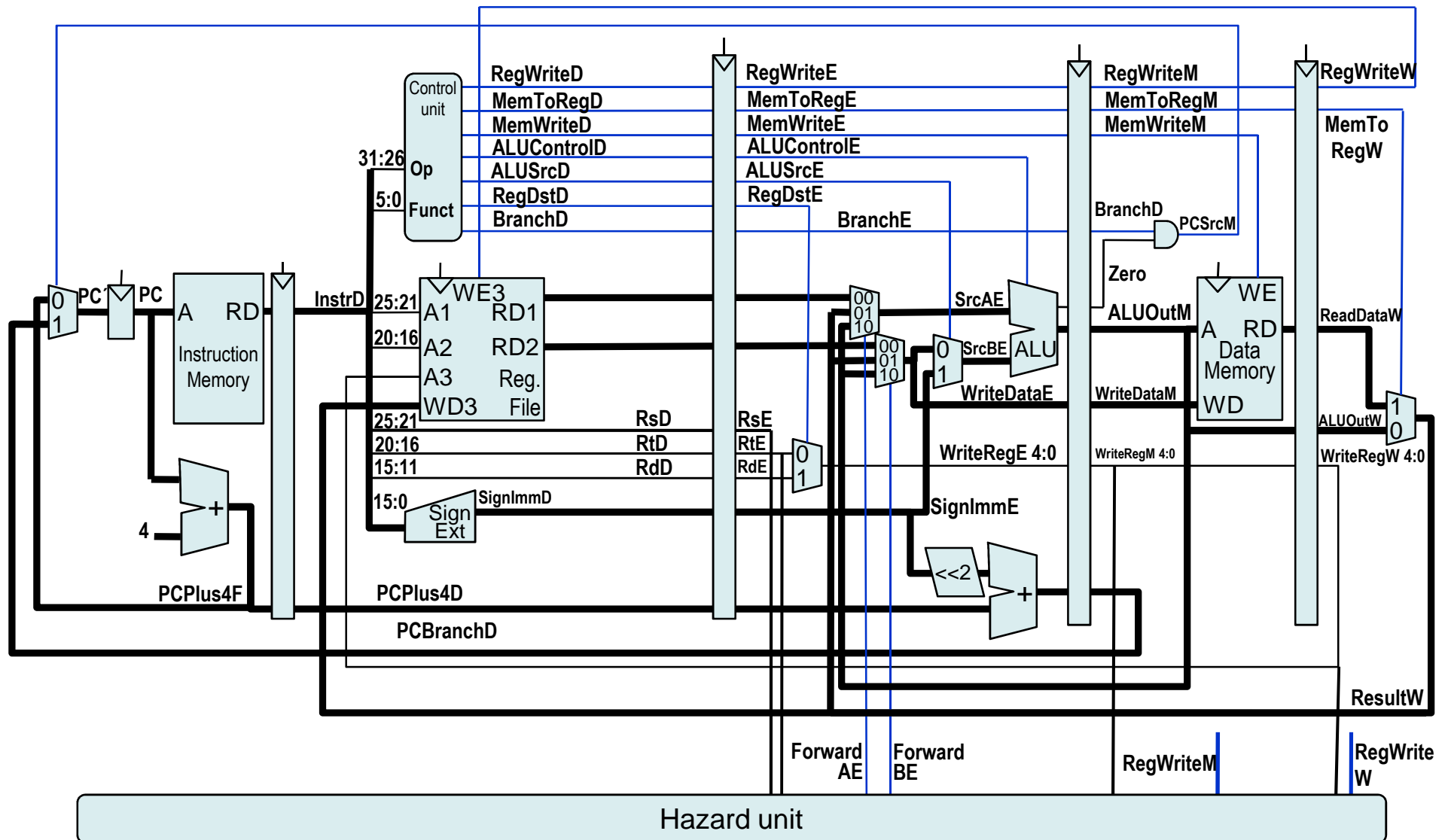
Řešení datových hazardů pozastavením (stall)



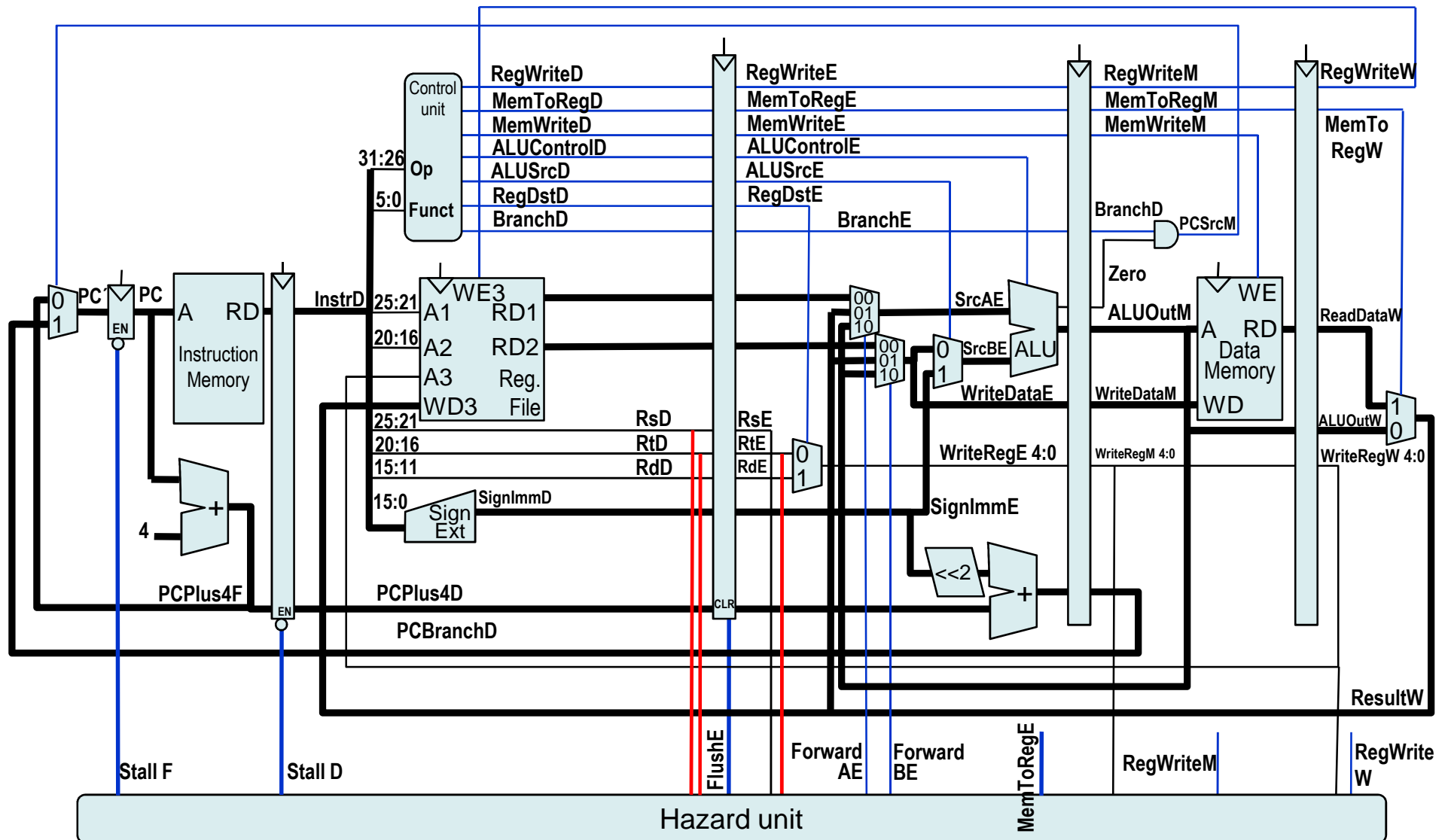
- pozastavení se dosáhne podržením hodnoty mezistupňových registrů
- výsledky z kolizního stupně se musejí „ztratit“ – řídicí signály umožňující měnit stav (kontext) procesoru (zápis pracovních registrů nebo do paměti, řízení povolení větvení) se nulují
- obojí se dosáhne přidáním řídicích vodičů k mezistupňovým registrům umožňujících měnit/uchovat nebo nulovat jejich obsah

lw: typ I, rs – básová adresa, imm – offset, rt – kde uložit

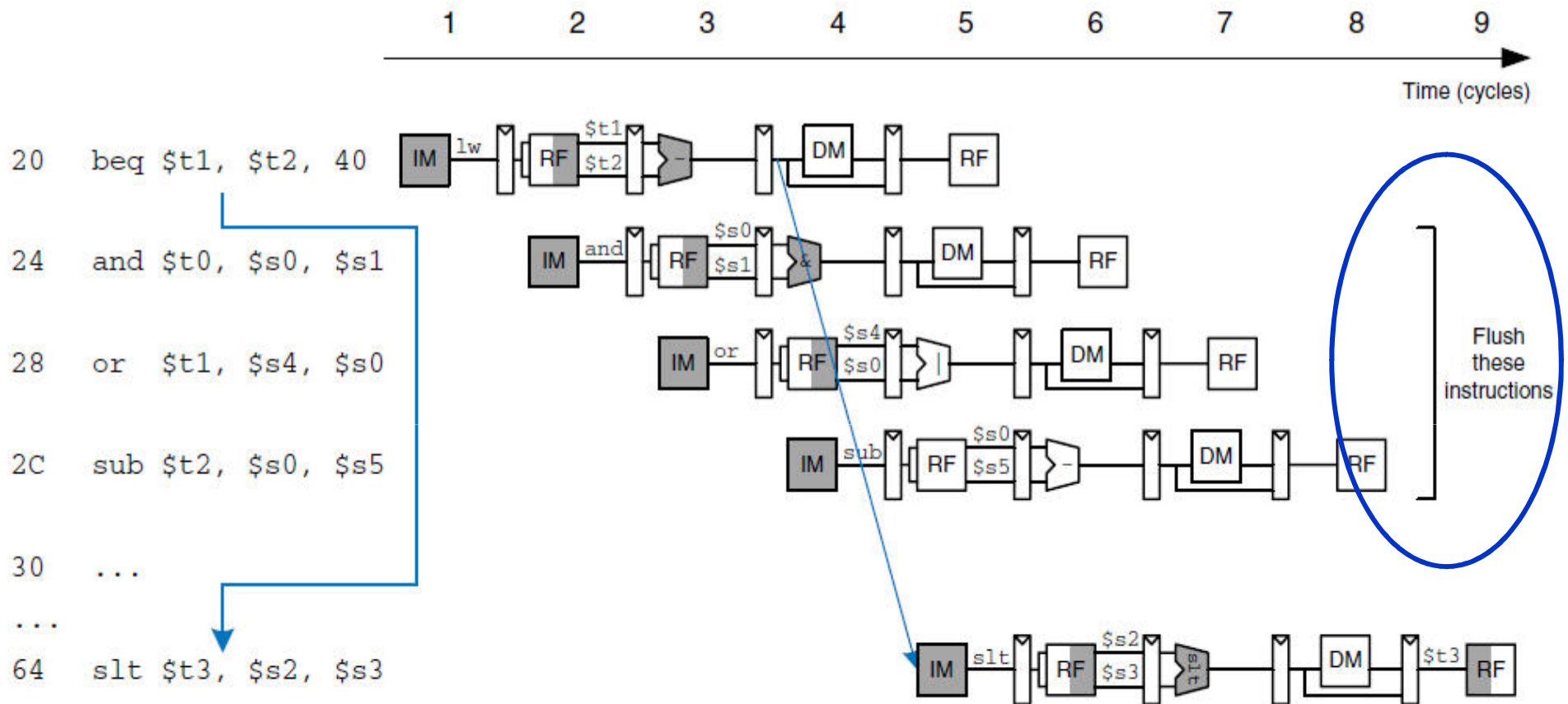
Stávající procesor



Řešení datových hazardů pozastavením (stall)

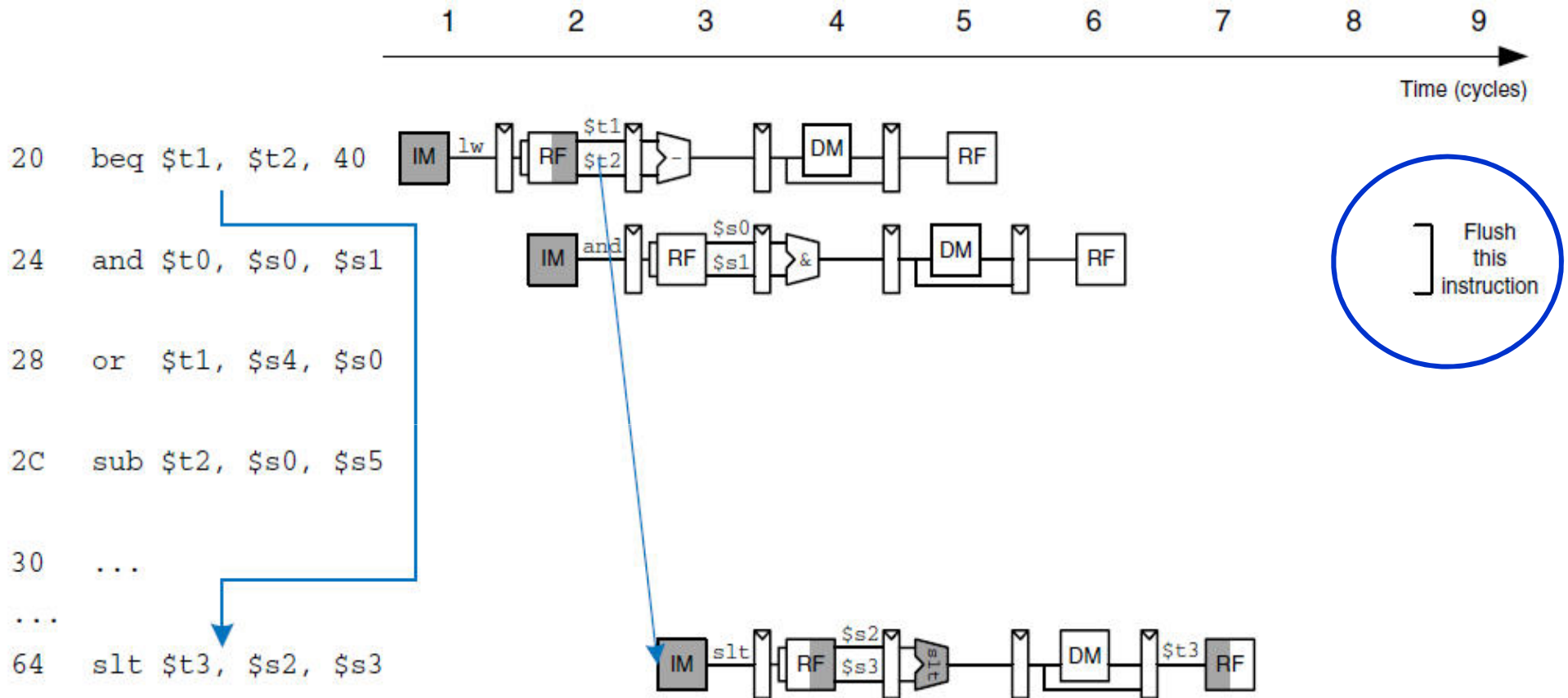


Řídicí hazardy



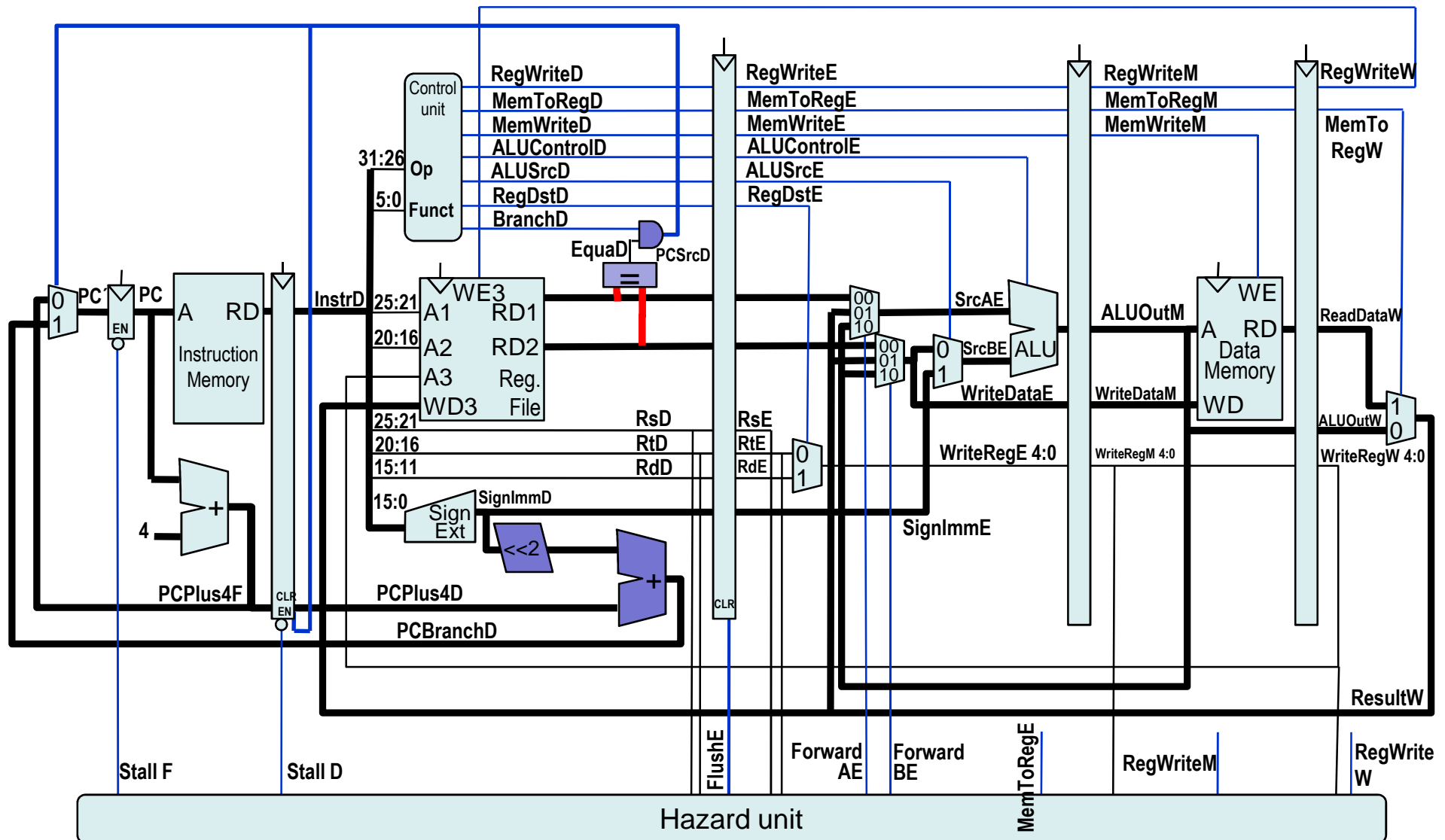
- výsledek porovnání je znám až v 4. cyklu.. Proč?

Řídicí hazardy – raději znát výsledek dříve..

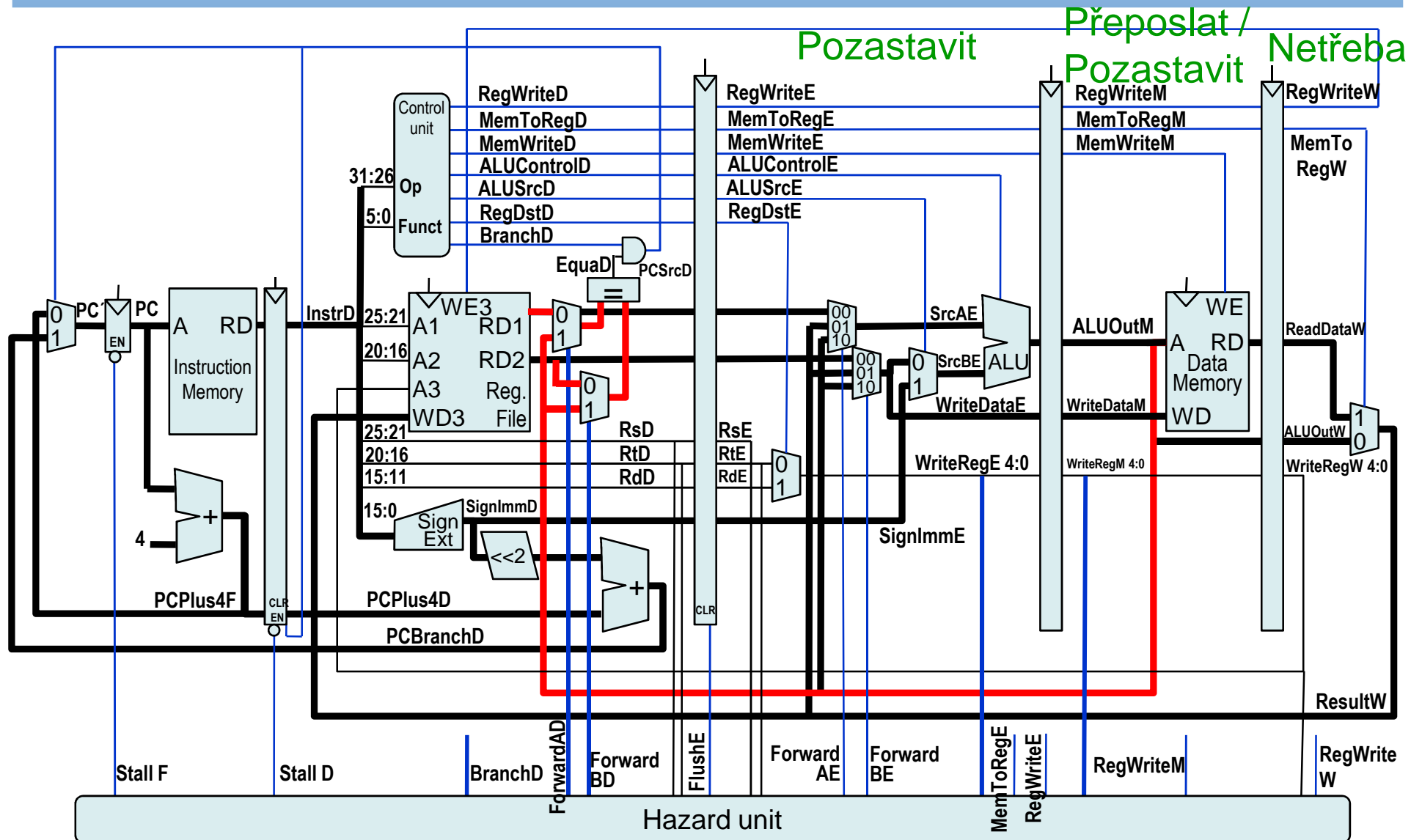


- pokud dokážeme stanovit výsledek porovnání už v 2. cyklu můžeme redukovat tzv. „misprediction penalty“
- přesun rozhodování dopředu může zavést nové RAW hazardy..!!!

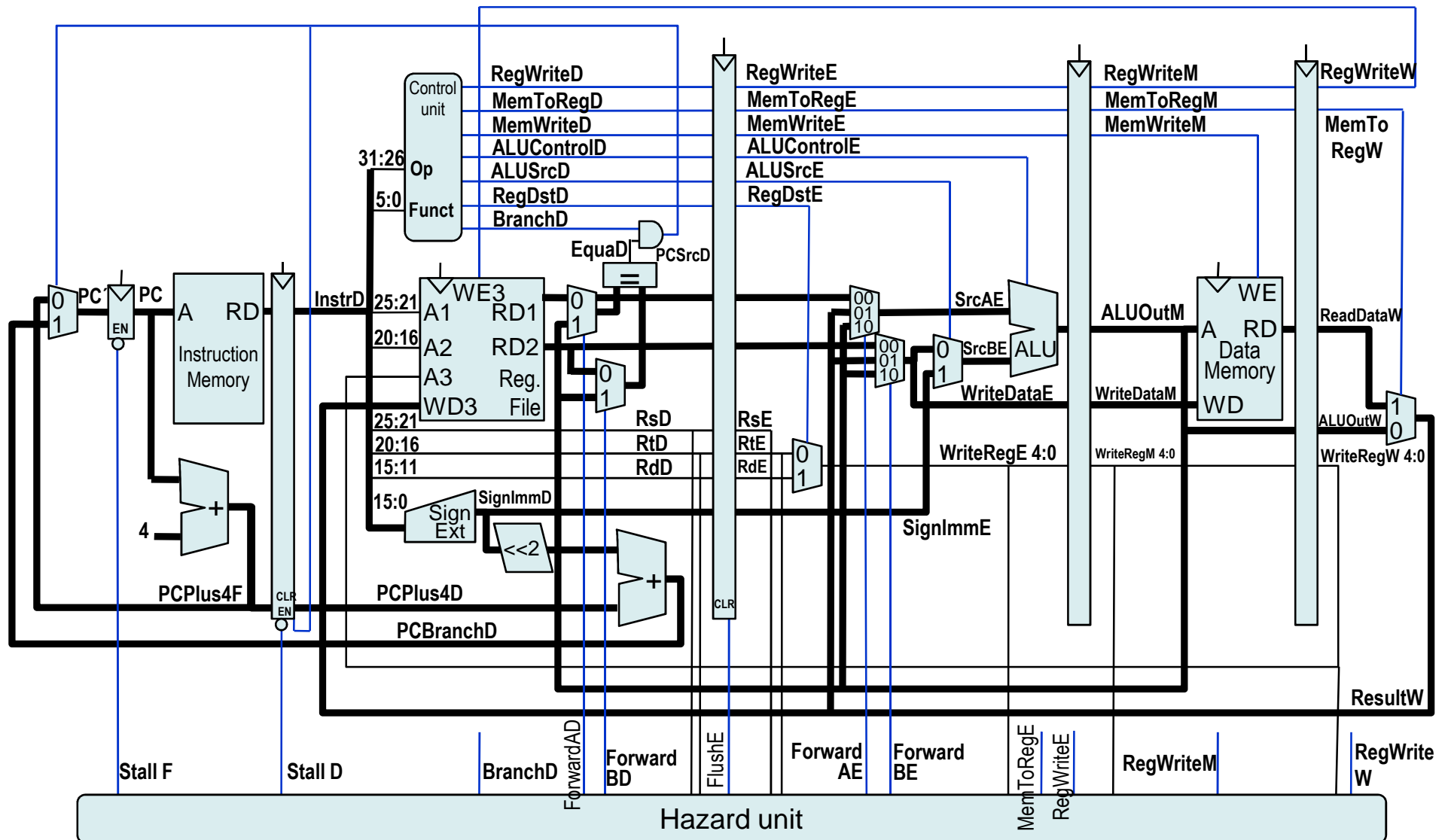
Řešení řídicích hazardů vyprázdněním (flush)



Řešení vzniklých RAW hazardů přeposíláním nebo pozastavením



Hotovo – navržený zřetězený procesor



Zřetězený procesor – výkon: $IPS = IC / T = IPC_{str} \cdot f_{CLK}$

- Jaká může být maximální frekvence procesoru?
- Který stupeň je nejpomalejší?
- Dobu cyklu určuje nejpomalejší stupeň
- V našem případě:
 $T_c = 300 \text{ ns} \rightarrow 3\,333 \text{ kHz}$

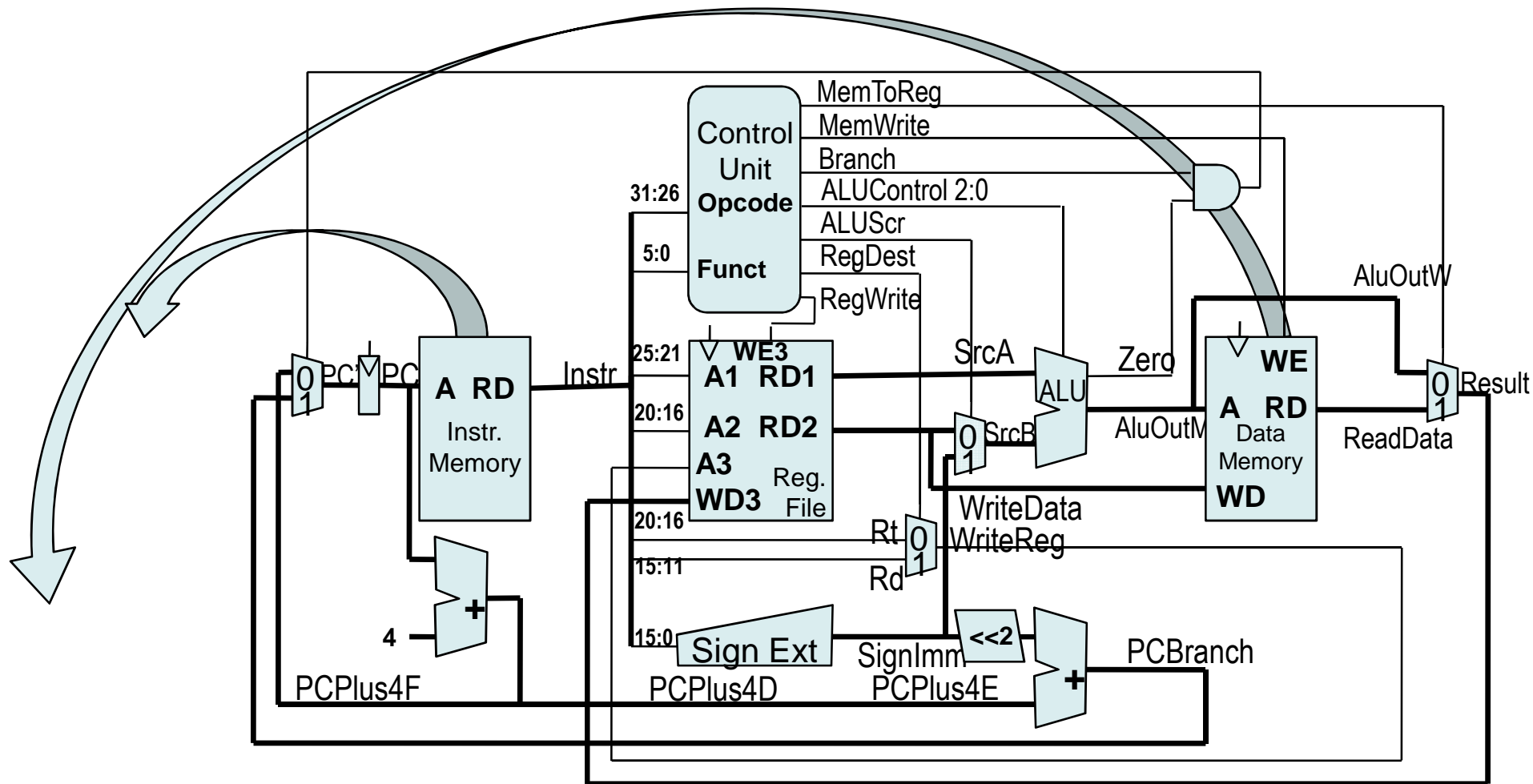
Zanedbejme plnění pipeline, všechna pozastavení pipeline a všechna vyprázdnění. Pak bude $IPC = 1$.

$IPS = 1 \cdot 3\,333\,000 = 3\,333\,000$ instrukcí za sekundu

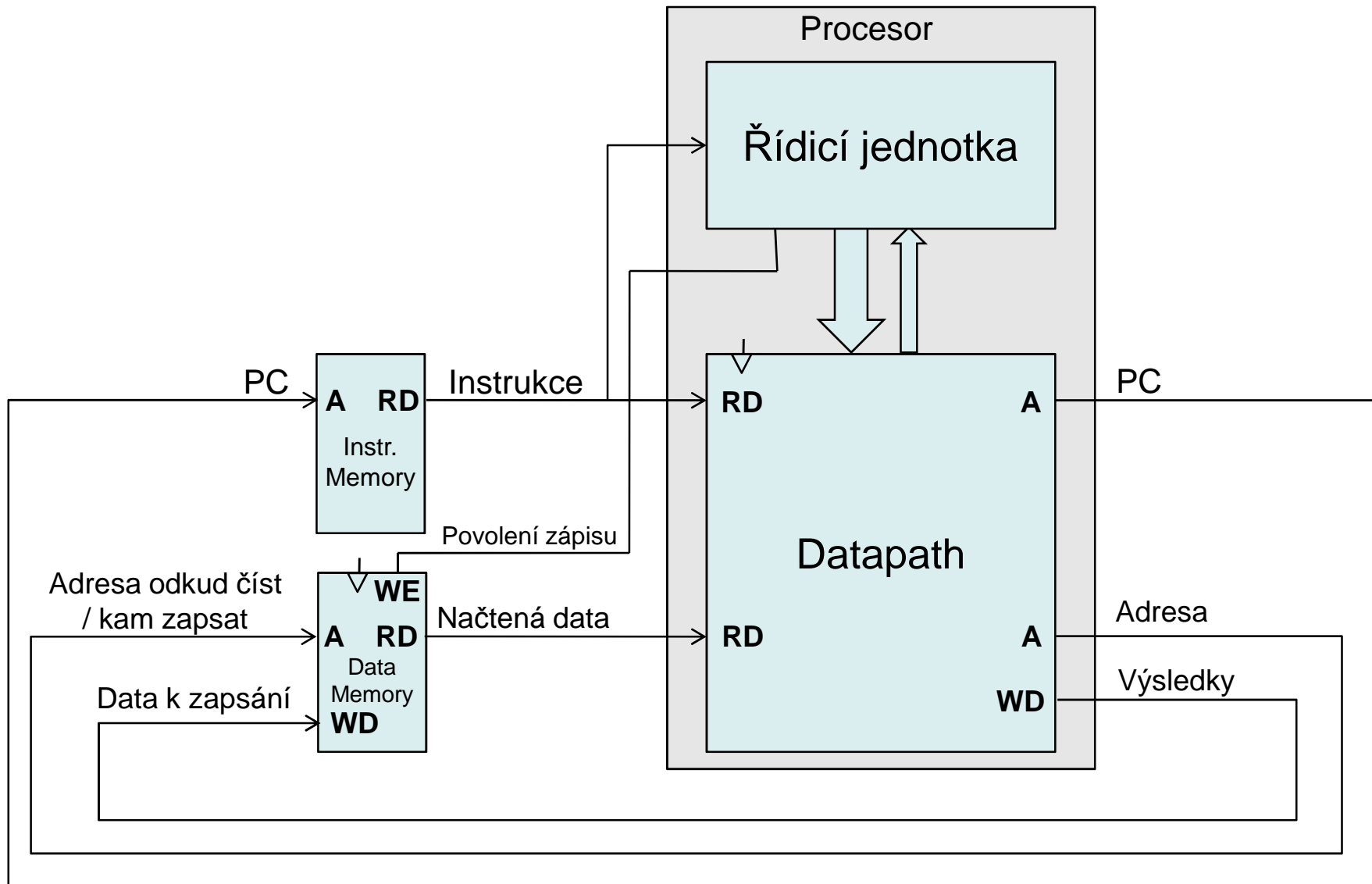
- Zavedením 5-stupňového zřetězení jsme zlepšili propustnost $3\,333\,000 / 980\,000 = 3,4$ krát! (i za předpokladu $IPC=1$)

Co jsme navrhli?

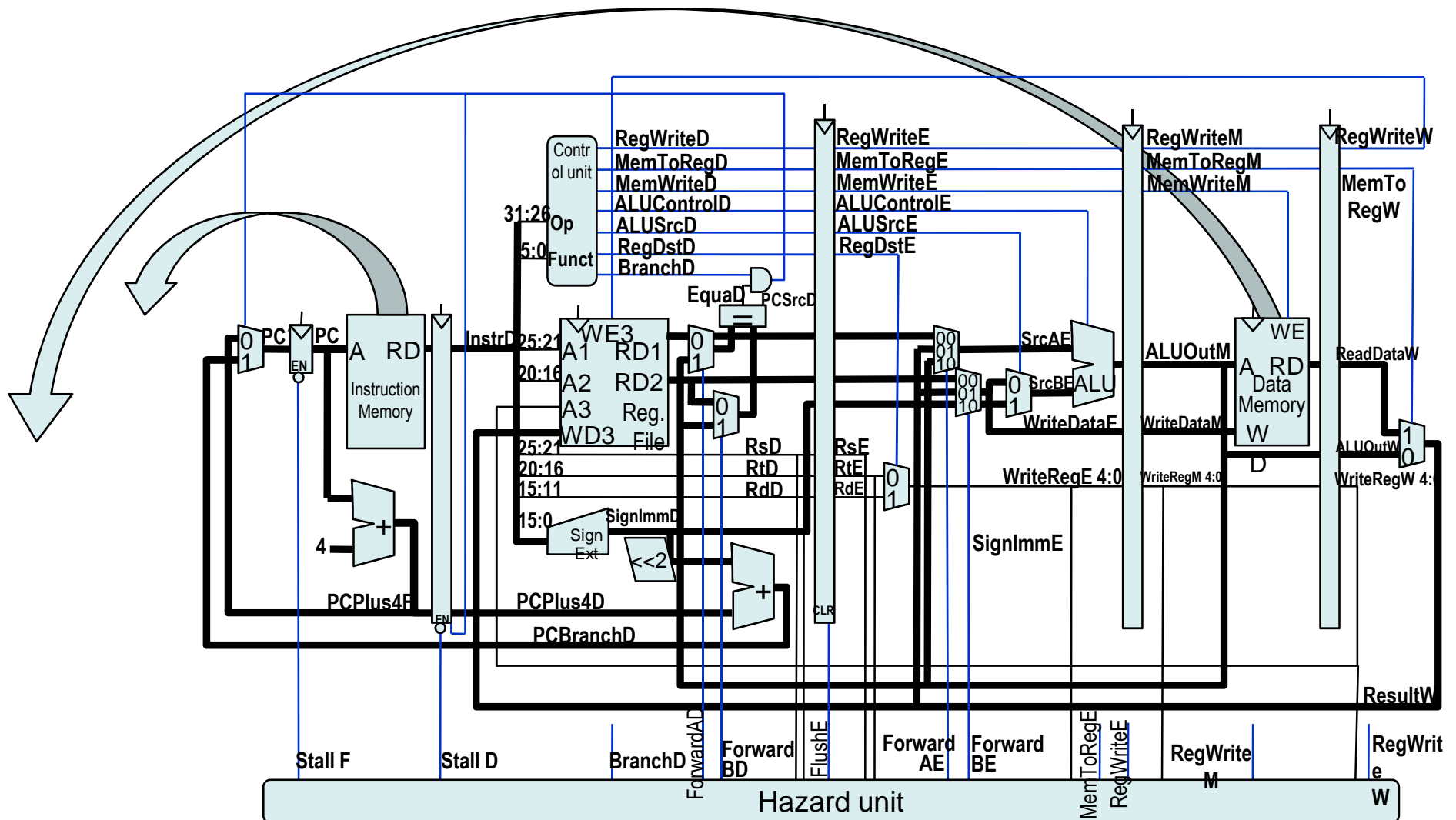
Návrat k nezřetězenému procesoru



Co jsme navrhli?

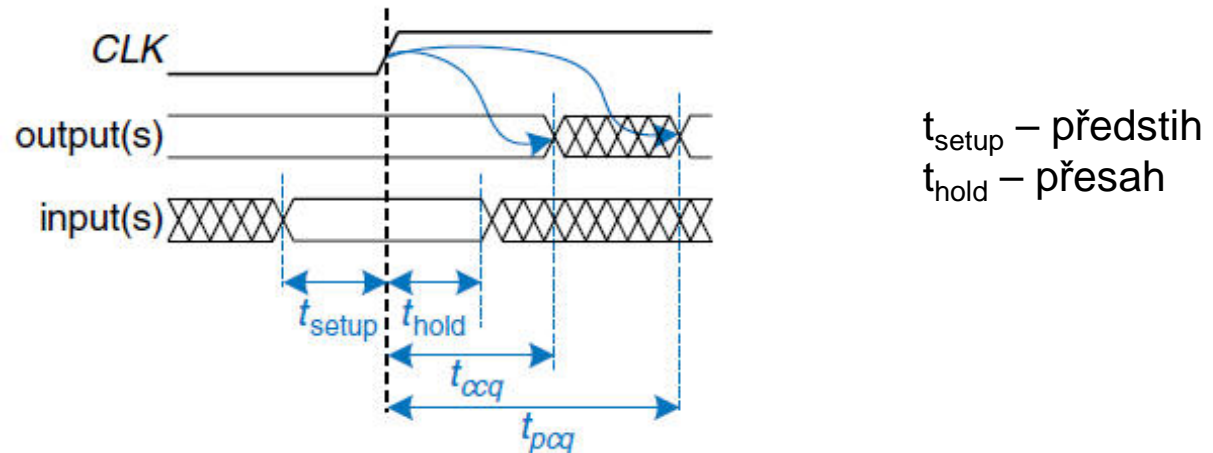


Co jsme navrhli? – zřetězená verze

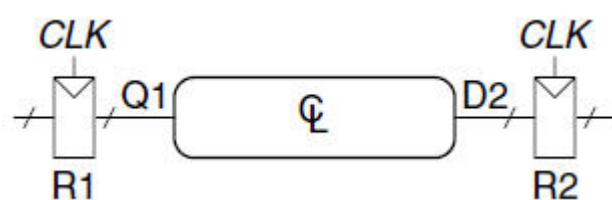


Zřetězený procesor – časování

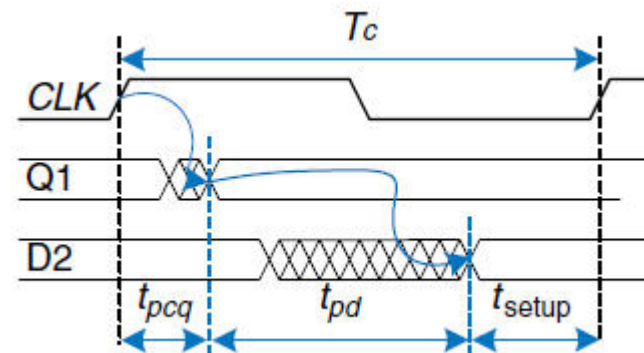
- Specifikace časování pro synchronní sekvenční obvod:



- Omezující podmínka na předstih signálu před hodinami:

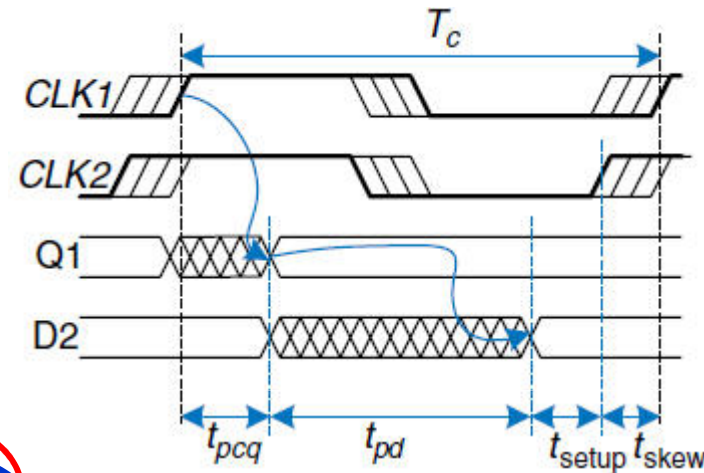
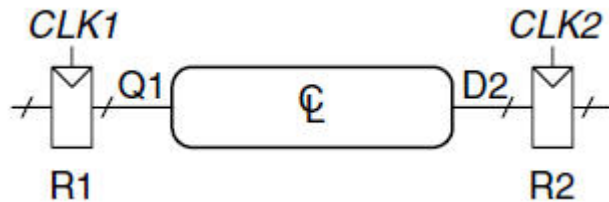


$$T_c \geq t_{pcq} + t_{pd} + t_{setup}$$



Zřetězený procesor – časování

- Omezující podmínka na předstih signálu před hodinami (zohlednění nedokonalosti rozvodu hodin):

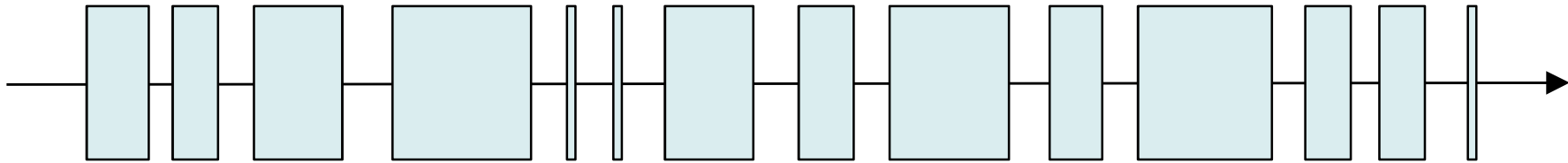


$$T_c \geq t_{pcq} + t_{pd} + t_{setup} + t_{skew}$$

Představuje omezení pokud se začíná blížit
nebo dokonce dominovat nad t_{pd}
(příliš hluboká pipeline / příliš mnoho stupňů...)

Vyvažování stupňů zřetězení

Lineární zřetězení:

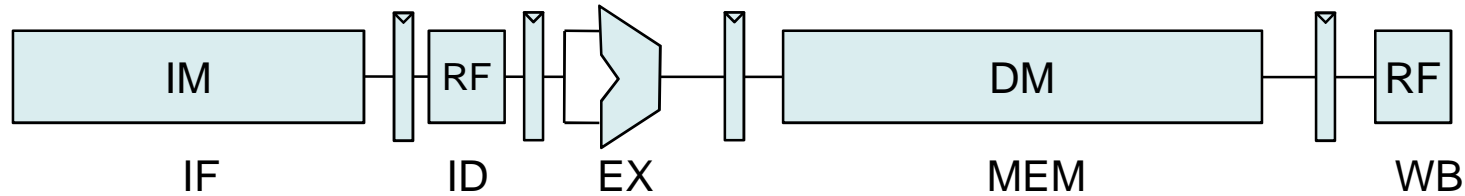


(též: stromový sumátor, stromová násobička, iterační dělička..)

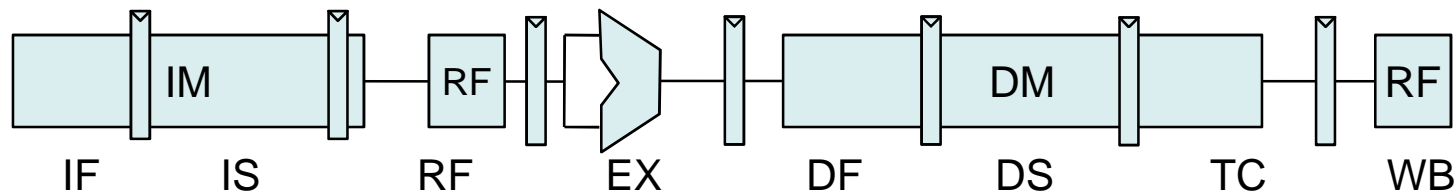
- **Vyvažování:** cílem je rozdělit jednotlivé bloky do N stupňů tak, aby ve zpoždění ve všech stupních bylo pokud možno stejné...
- Volba počtu stupňů závisí od preference: propustnost vs. latence

Superzřetězení

- nevyvážené 5-stupňové zřetězení:



- hlubší zřetězení vzniklá další dekompozicí



- přináší možnost dalšího zvýšení pracovní frekvence, avšak také řadu dalších problémů..
- další forwarding, nárůst pozastavení pipeline, hazardy