Na co se dnes podíváme

1. Co je to pojem architektura PC
2. Co je to RISC a CISC
3. Praktická ukázka obou architektur
4. Srovnání RISC a CISC

5. Chipsety a jejich funkce
6. Historické rozdělení chipsetu
7. Jak to vypadá dnes
8. Trend SoC (System on Chip)

Architektura PC

Architektura PC

- Architekturu moderně chápeme jako **pohled na podstatné vlastnosti počítačů.**

- Nejedná se tedy pouze o téma, které dnes budeme probírat (RISC a CISC).

Architektura počítače je jako plán stavby – neurčuje jen to, co kde je, ale i jak to spolupracuje.

Architektury CPU

- Zaměřme se však nyní na samotné instrukční sady, tedy RISC a CISC.

1. Co je to instrukční sada?
   - Je to soubor instrukcí, se kterými dokáže procesor (CPU) pracovat.
   - Je to tak trochu jako „jazyk, jímž procesor mluví“.

2. Co je to instrukce?
   - Základní příkaz, který procesor rozpozná a vykoná.

3. Z čeho se instrukce skládá?
   - Z operačního kódu a operandů

Příklad instrukce assembleru (x86):

```
ADD AX, BX
```

Tato instrukce říká: „sečti obsah registrů AX a BX a výsledek ulož do AX.“

Instrukční cyklus

(… -> Fetch -> Decode -> Execute -> …)

- Každá instrukce prochází tzv. instrukčním cyklem.

- Ten se skládá ze 3 hlavních částí:
  1. Fetch
     - Načtení instrukce z operační paměti do registru instrukce pomocí IP (Instruction Pointer)

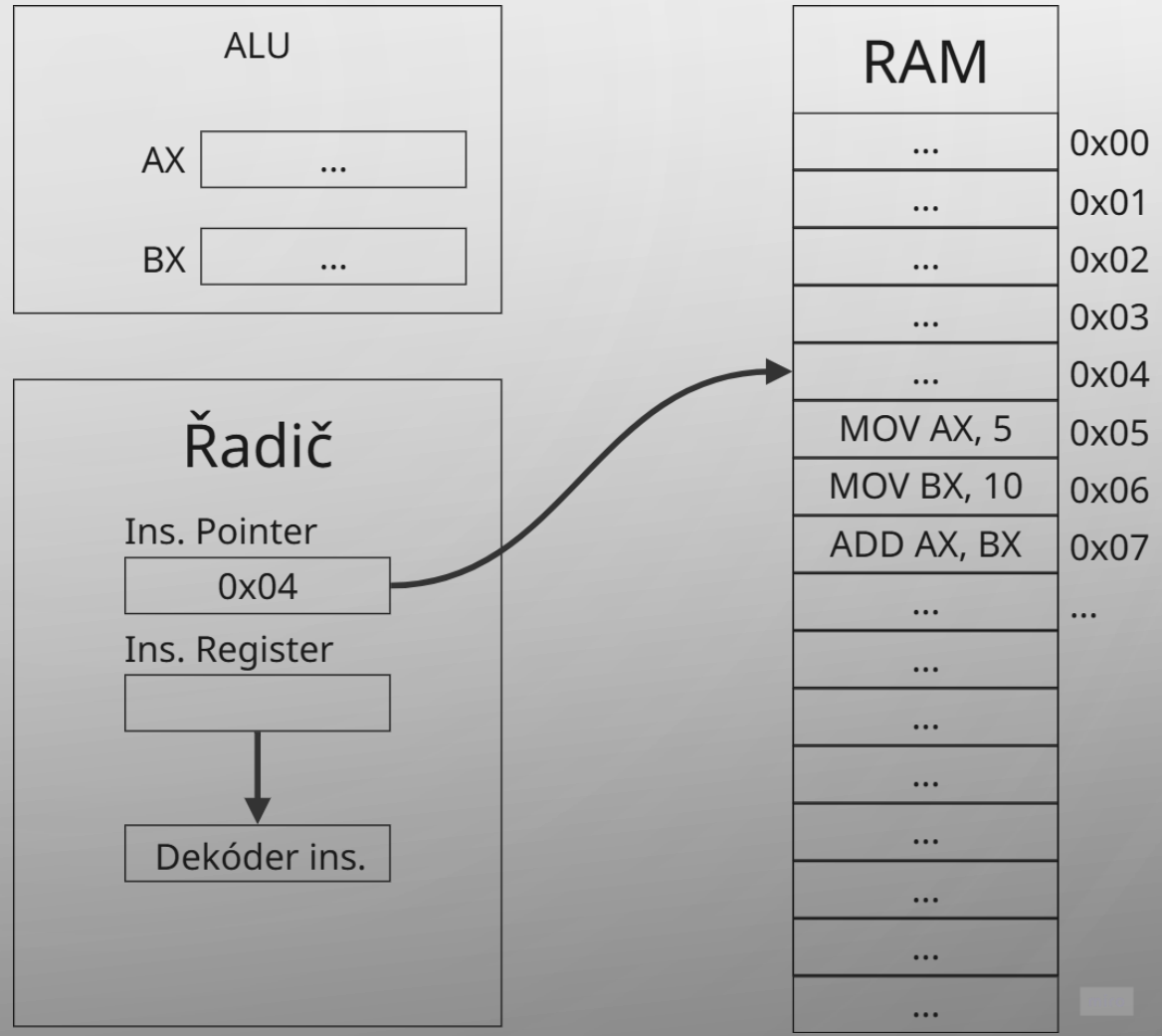
  2. Decode
     - Dekódování instrukce (jaký příkaz se má vykonat a s jakými daty)

  3. Execute
     - Provedení samotné instrukce

Instrukční cyklus

(… -> Fetch -> Decode -> Execute -> …)

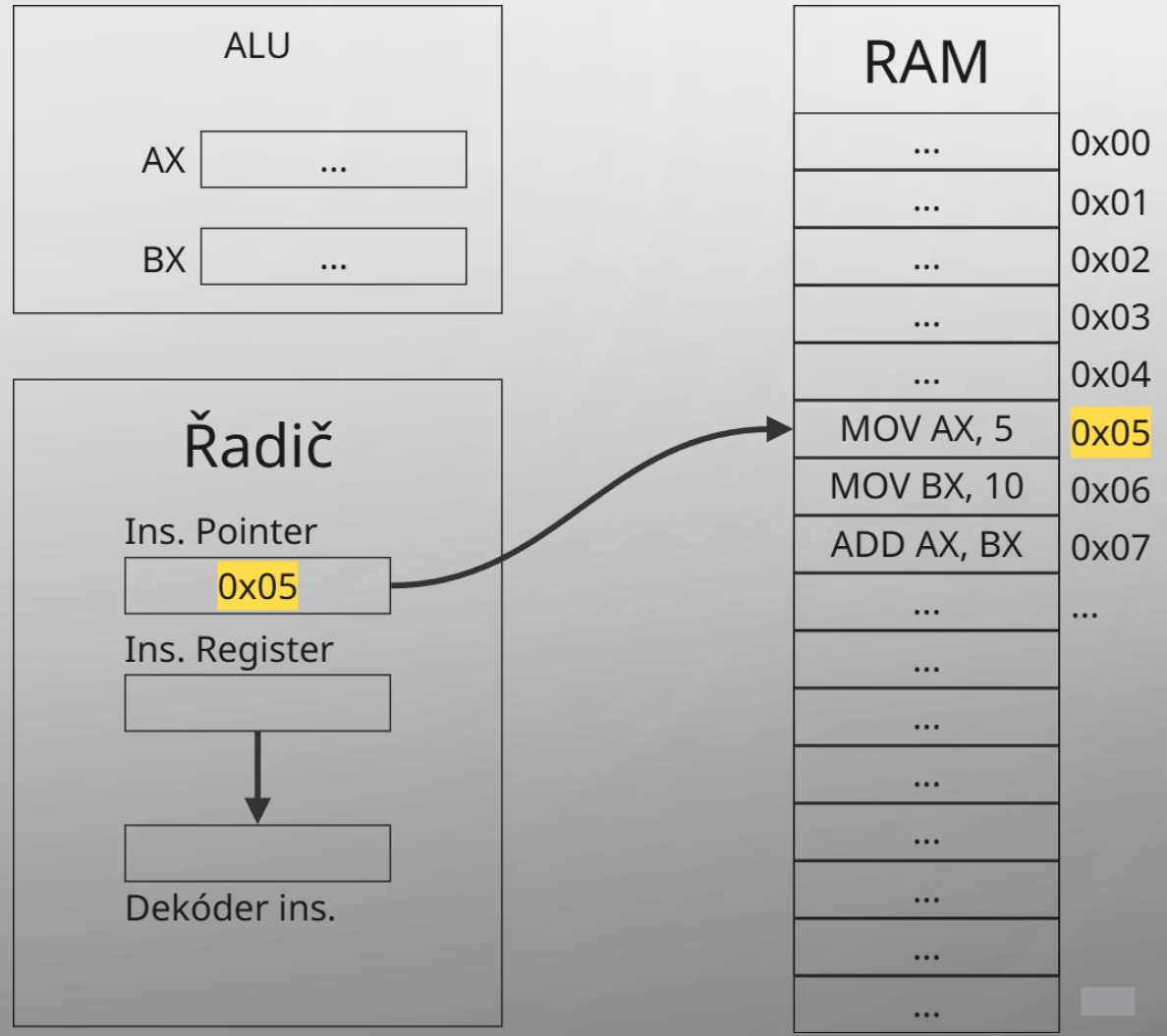
- Zde je jednoduchá ilustrace průběhu instrukčních cyklů.
- Je nutno podotknout že registry jsou umístěny v ALU pouze pro jednoduchost a jsou zde vyobrazeny pouze dva a že již v OP jsou instrukce v podobě strojového kódu.
- Toto schéma obsahuje pouze nejnutnější části pro ukázku průběhu instrukčních cyklů.



Instrukční cyklus

(… -> Fetch -> Decode -> Execute -> …)

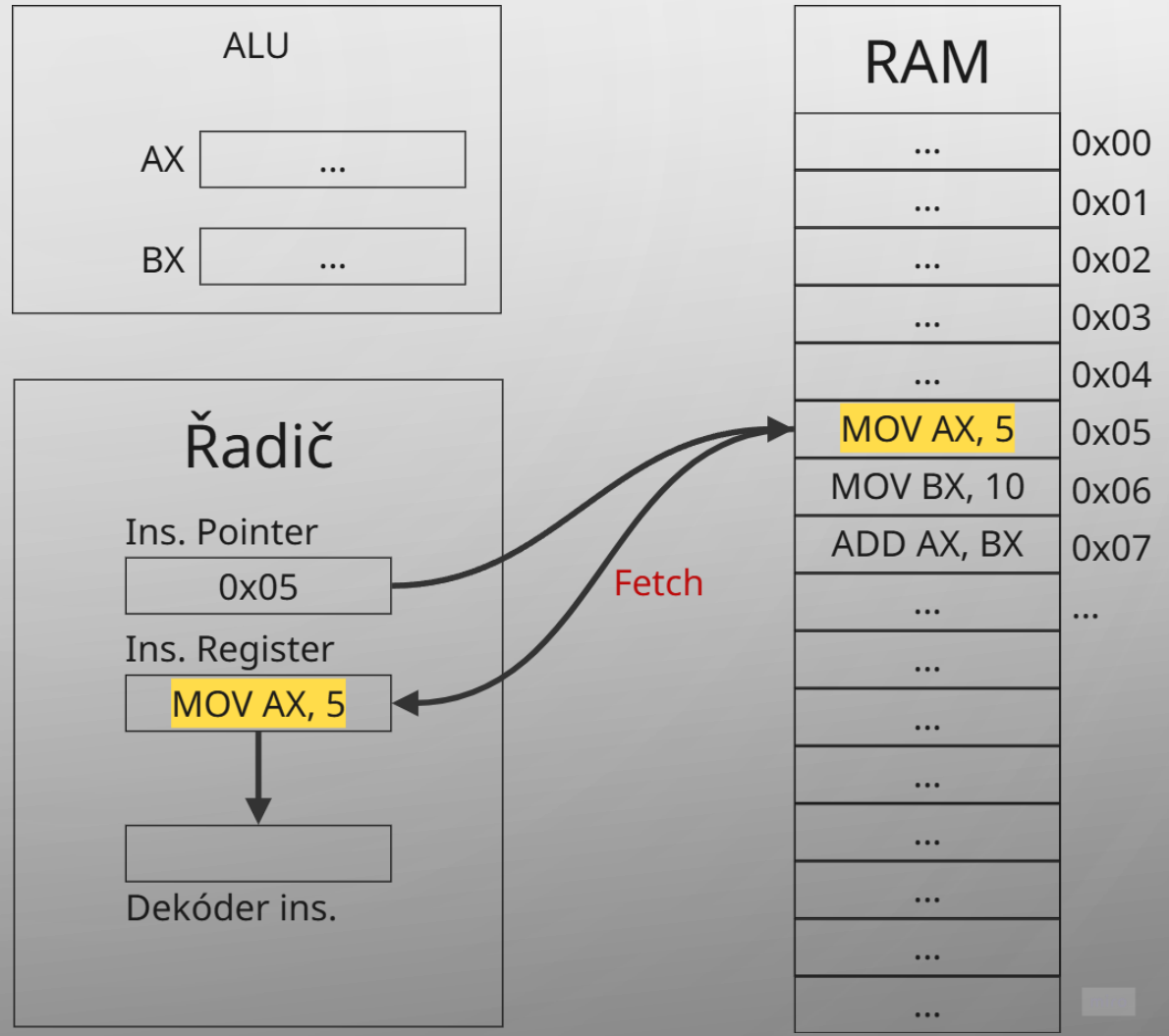
- Ins. Pointer (Dále jen IP) se s každým cyklem zvětší o 1 nebo na specifickou adresu, pokud proběhne skok.
- Nyní IP ukazuje na adresu 0x05 v operační paměti (Dále jen OP).



Instrukční cyklus

(… -> Fetch -> Decode -> Execute -> …)

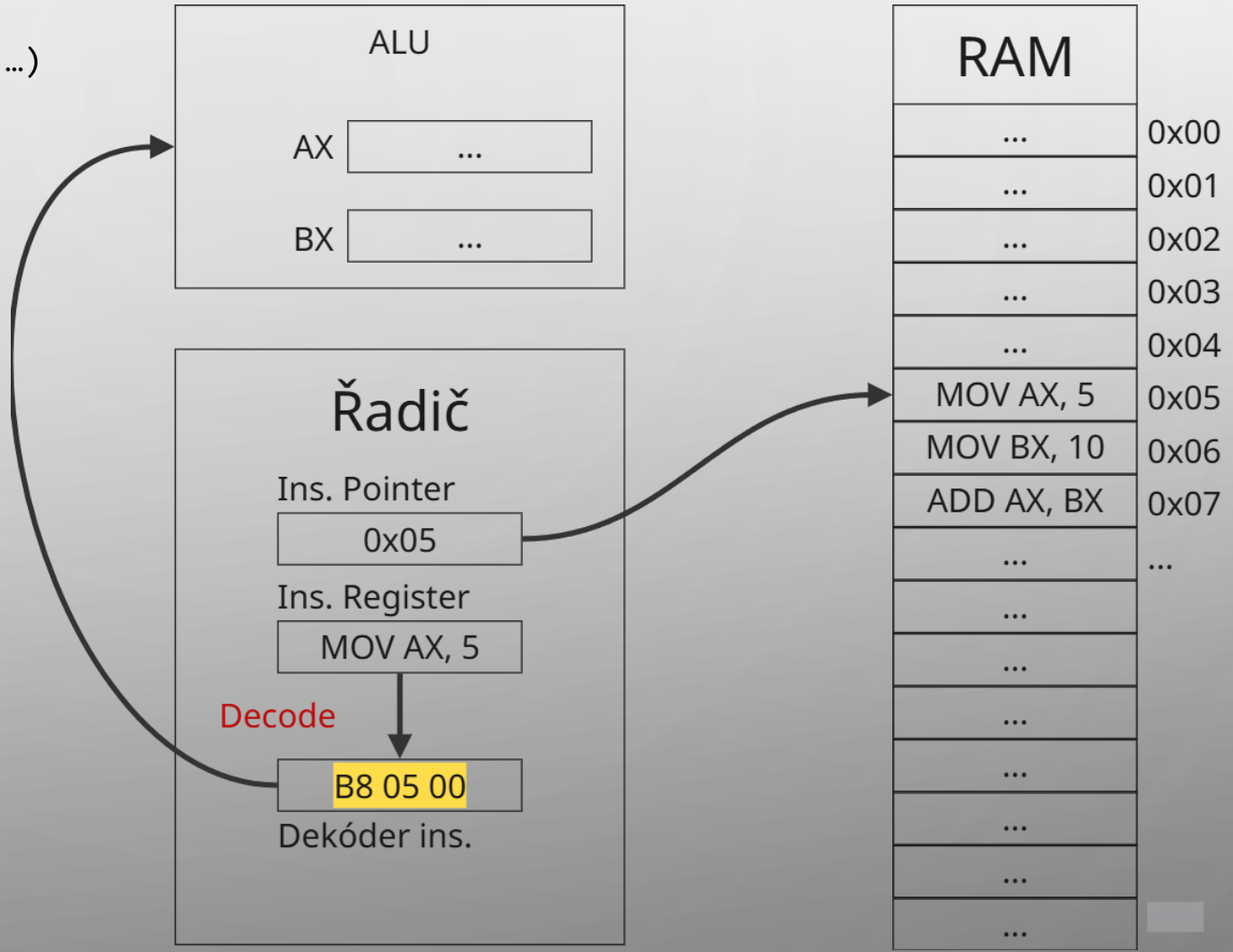
- Instrukce na této adrese se nahraje z OP do registru instrukce.
- Tento proces jednoduše nazýváme FETCH.



Instrukční cyklus

(… -> Fetch -> Decode -> Execute -> …)

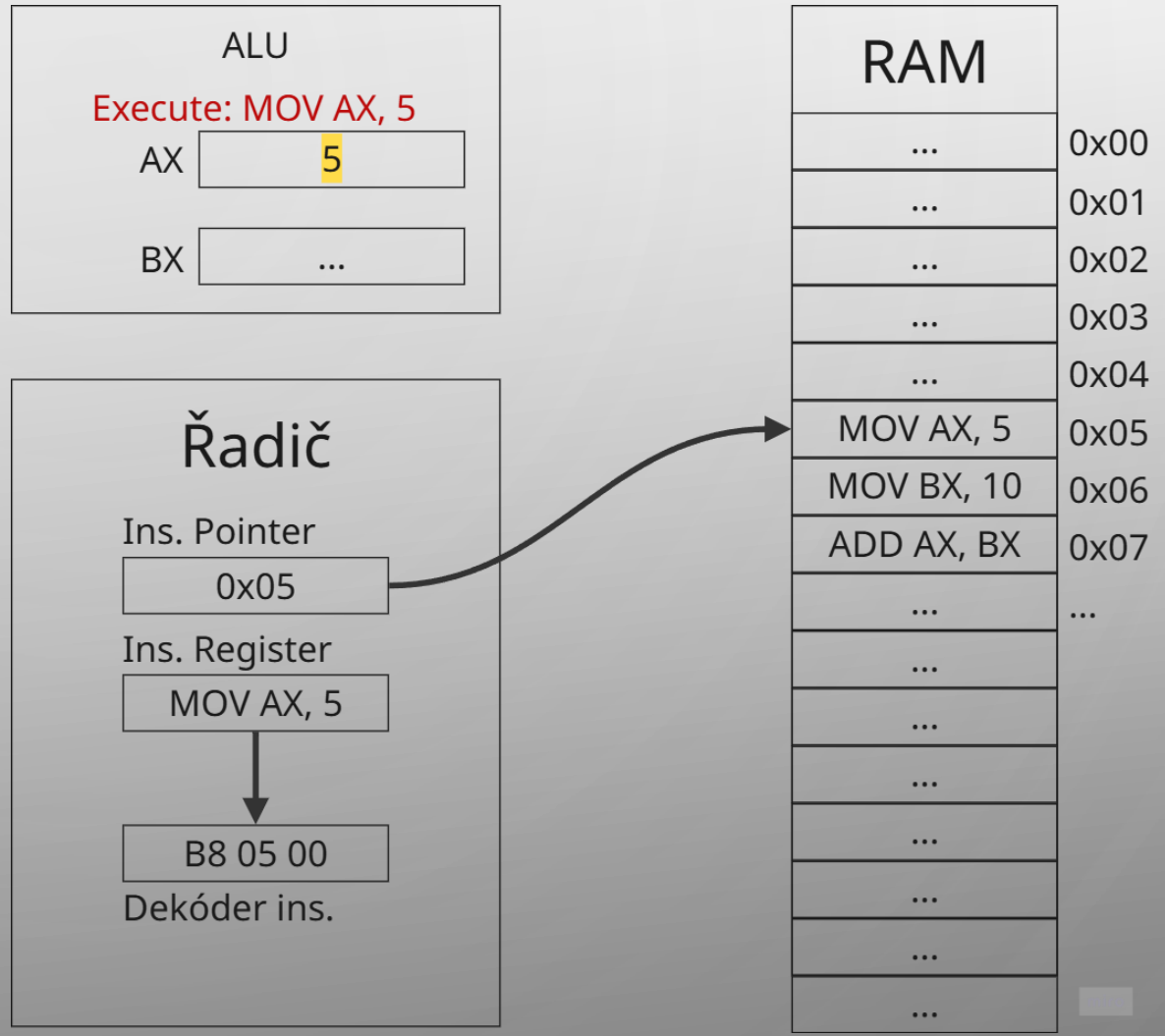
- Tato instrukce se dále dekóduje a „předá“ se do ALU.
- V reálném případě také dochází ke generování různých řídících signálů, ale zde nám jde pouze o pochopení průchodu instrukce.



Instrukční cyklus

(… -> Fetch -> Decode -> Execute -> …)

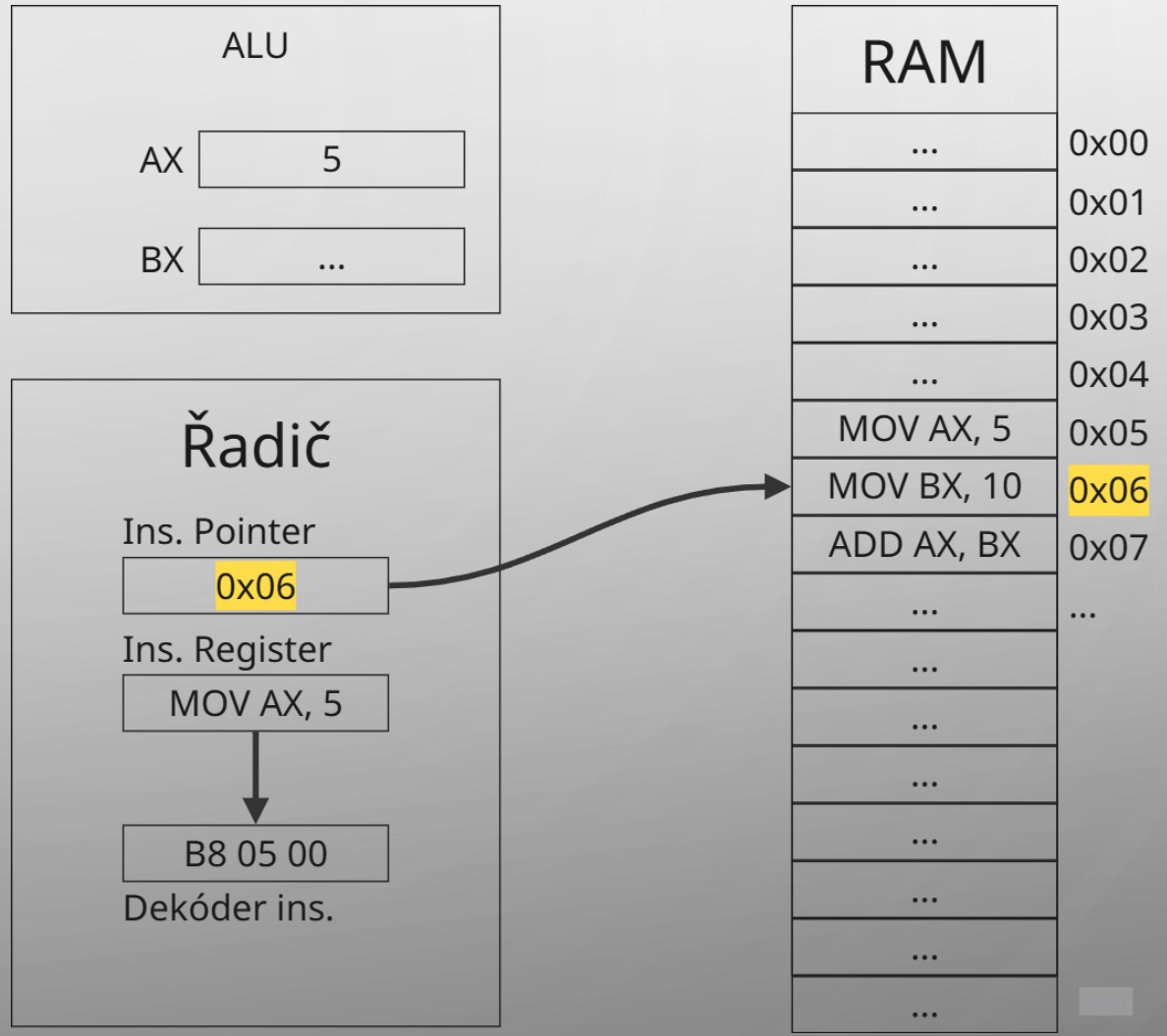
- V ALU se instrukce provede a do příslušného registru (AX) se nahraje číslo 5.



Instrukční cyklus

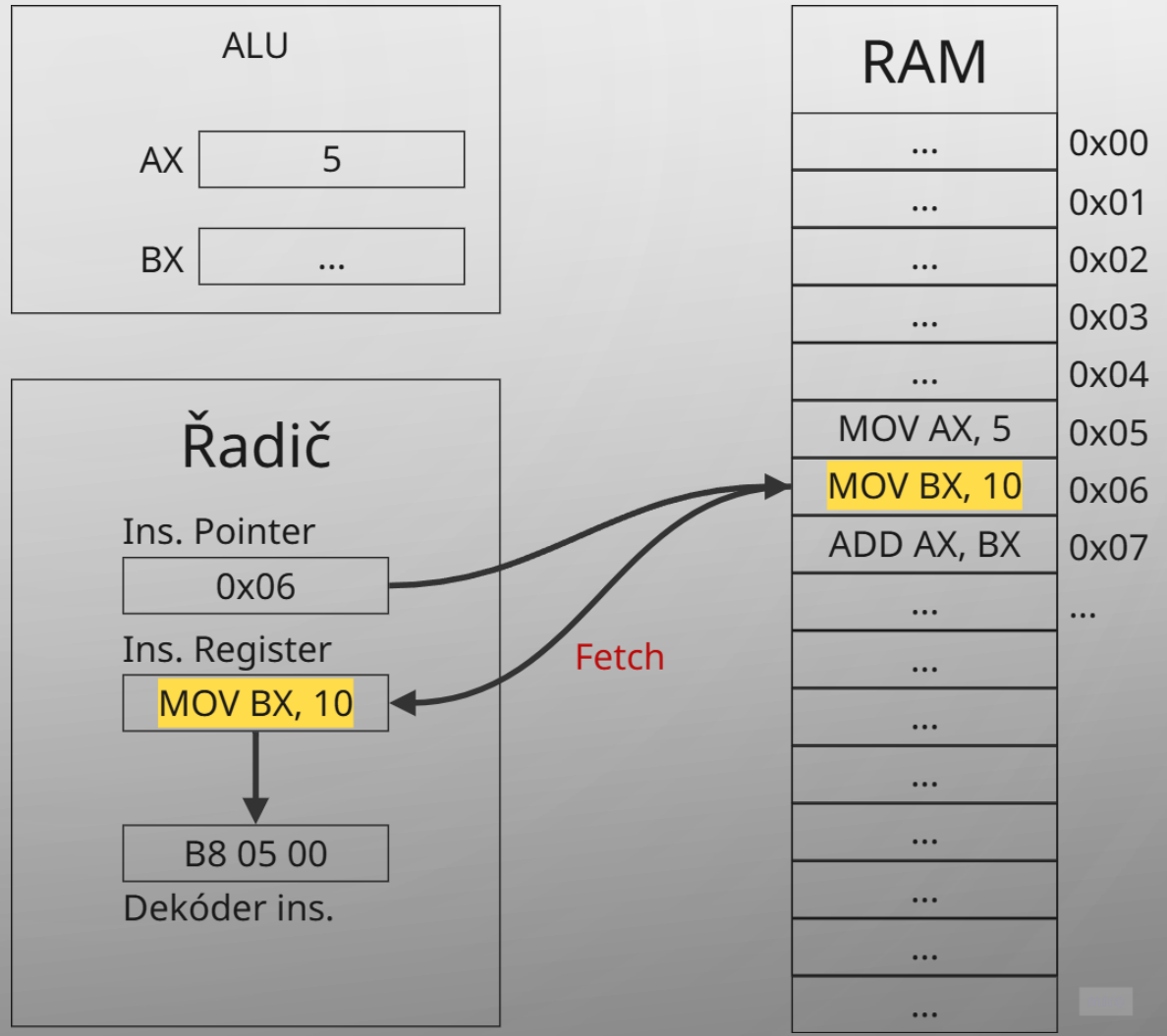
(… -> Fetch -> Decode -> Execute -> …)

- Dále pokračujeme přičtením 1 k adrese v IP a opakujeme celý proces.
- Tento samotný proces se nazývá Instrukční cyklus.



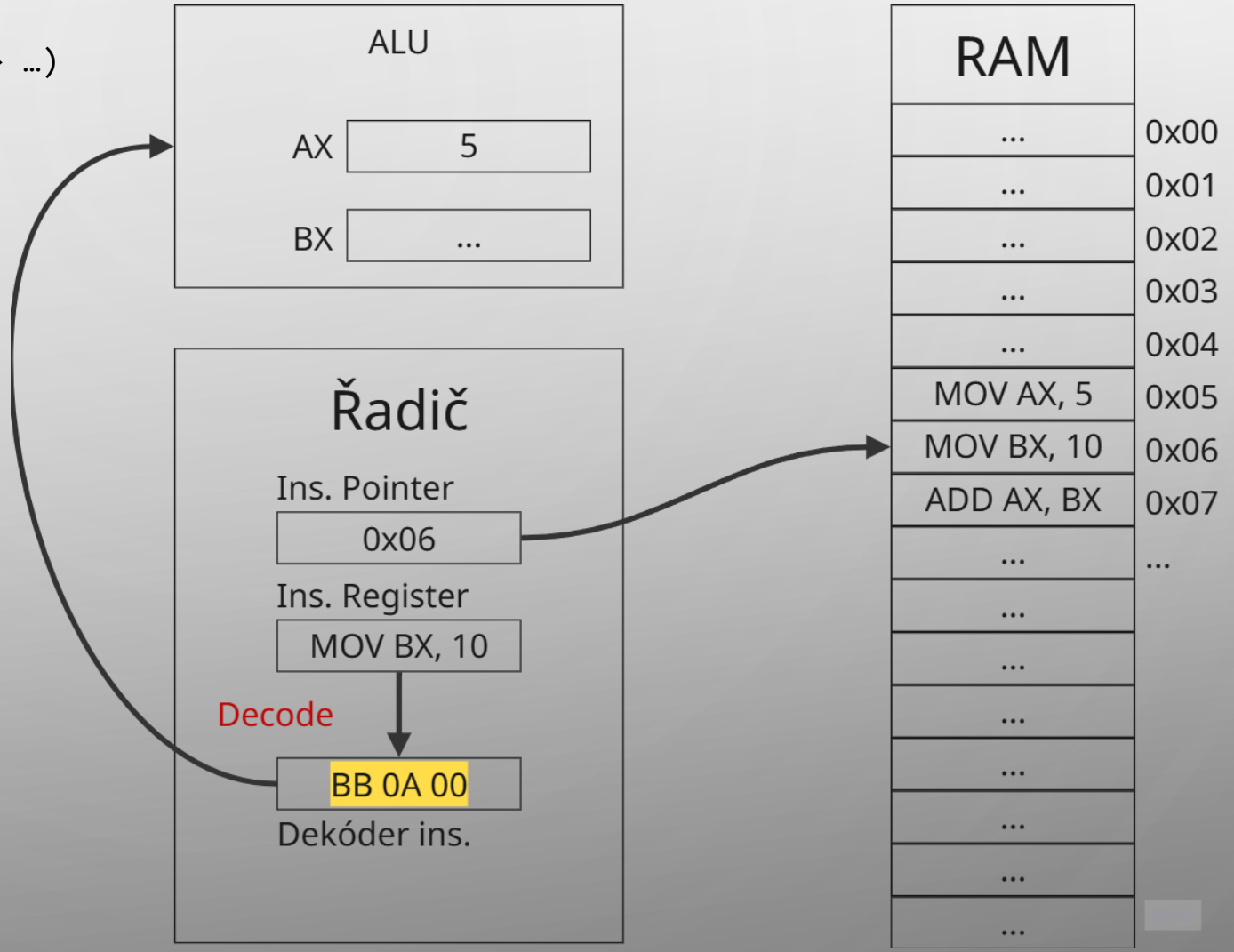
Instrukční cyklus

(… -> Fetch -> Decode -> Execute -> …)



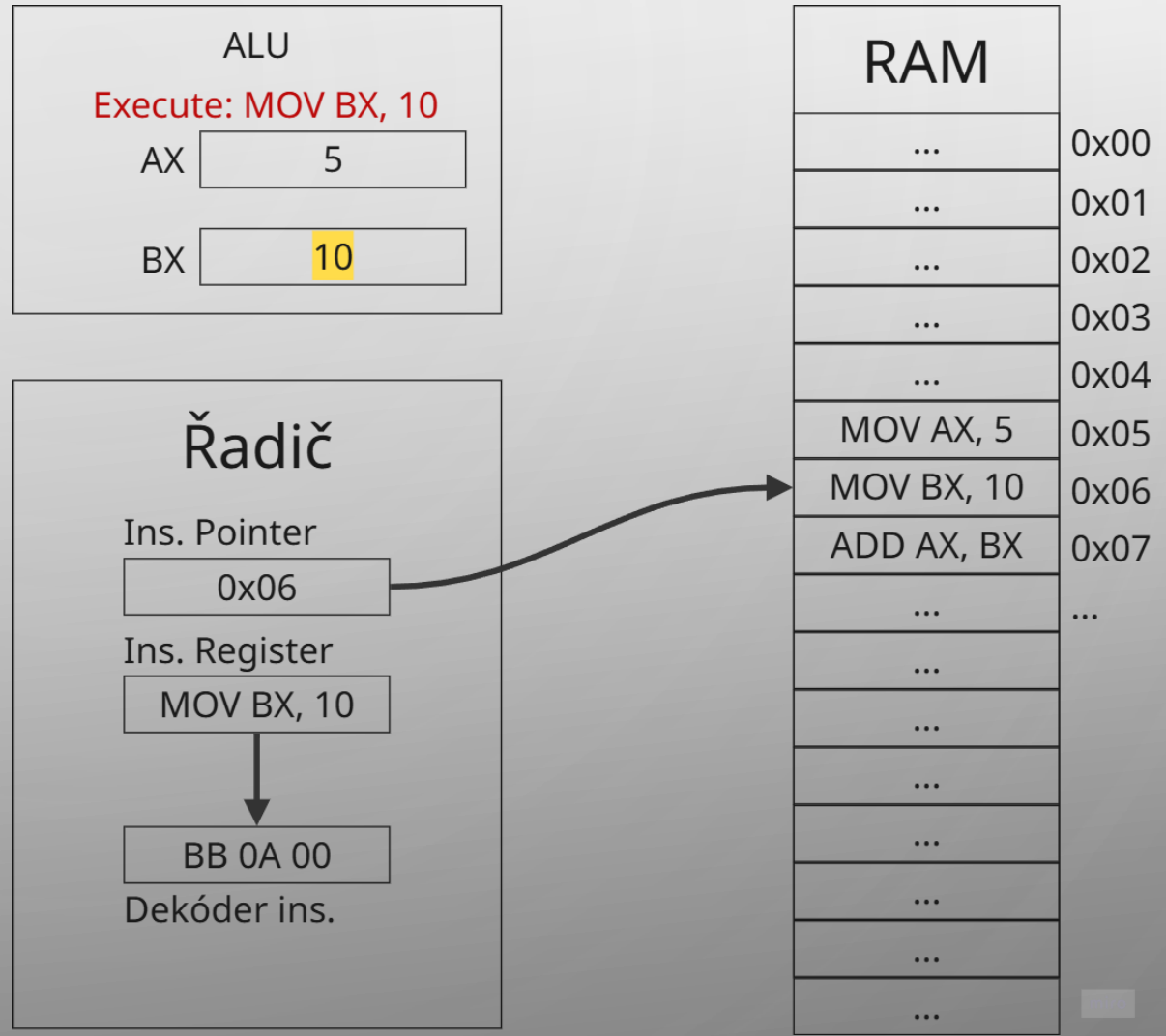
Instrukční cyklus

(… -> Fetch -> Decode -> Execute -> …)



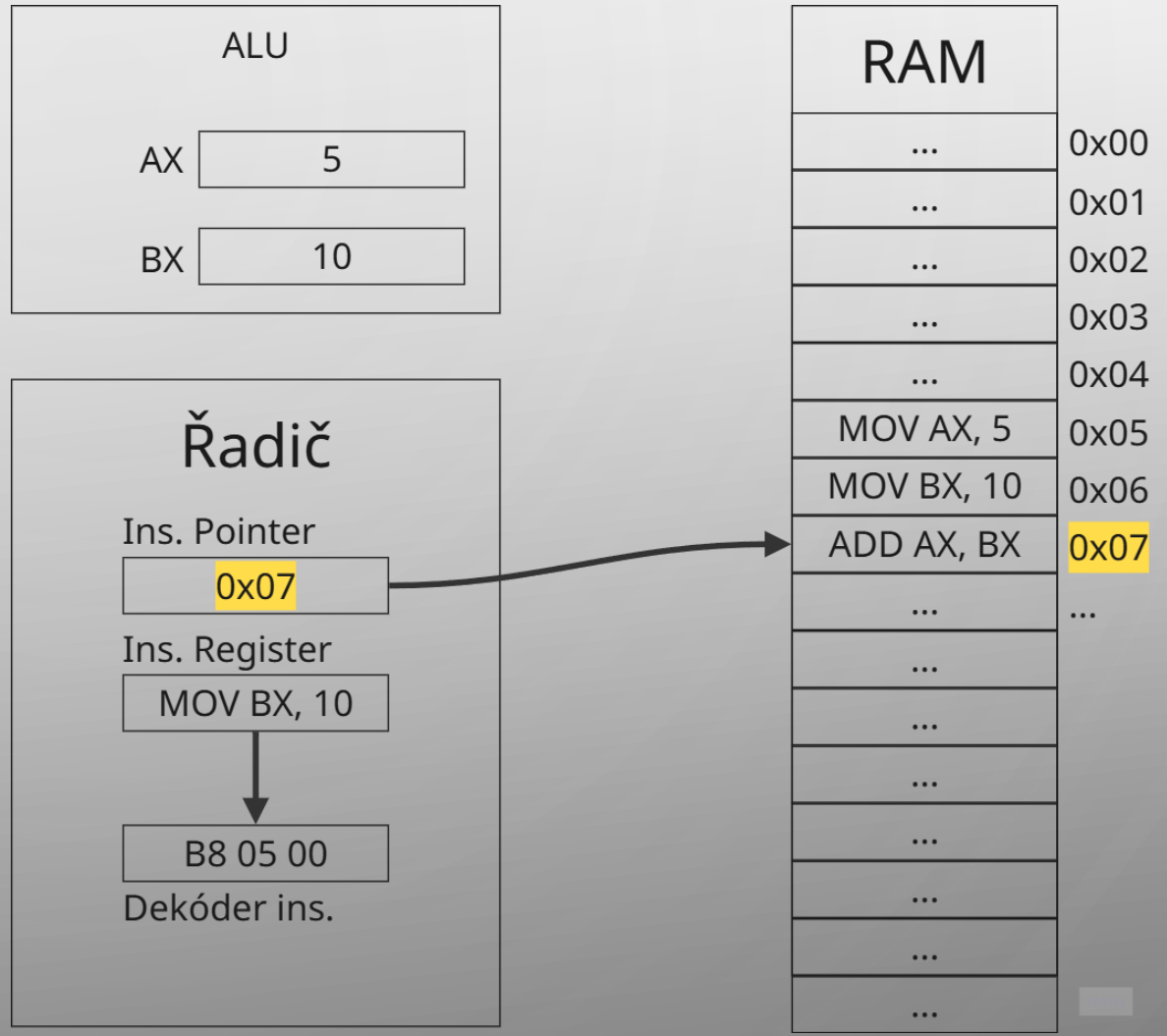
Instrukční cyklus

(… -> Fetch -> Decode -> Execute -> …)



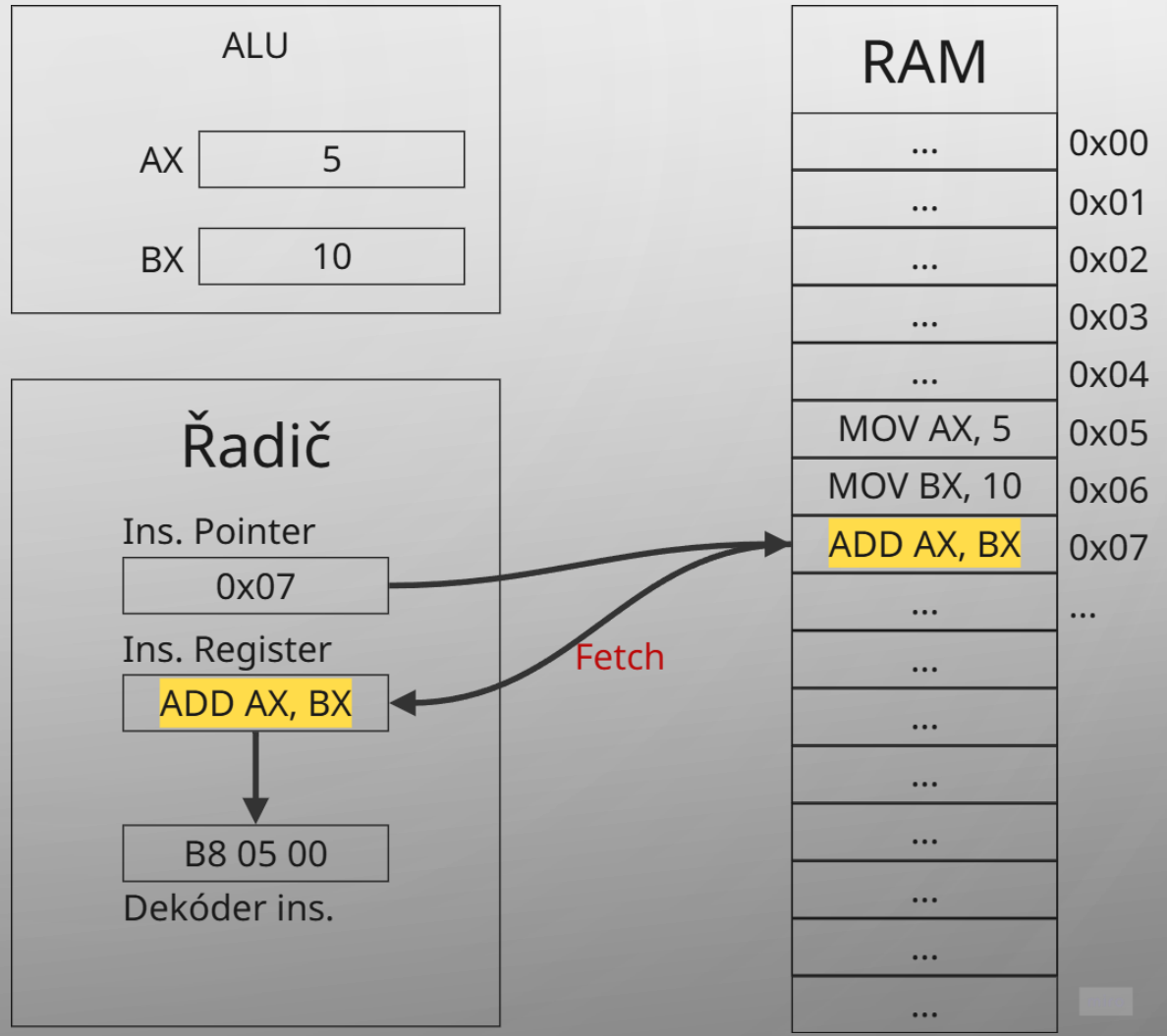
Instrukční cyklus

(… -> Fetch -> Decode -> Execute -> …)



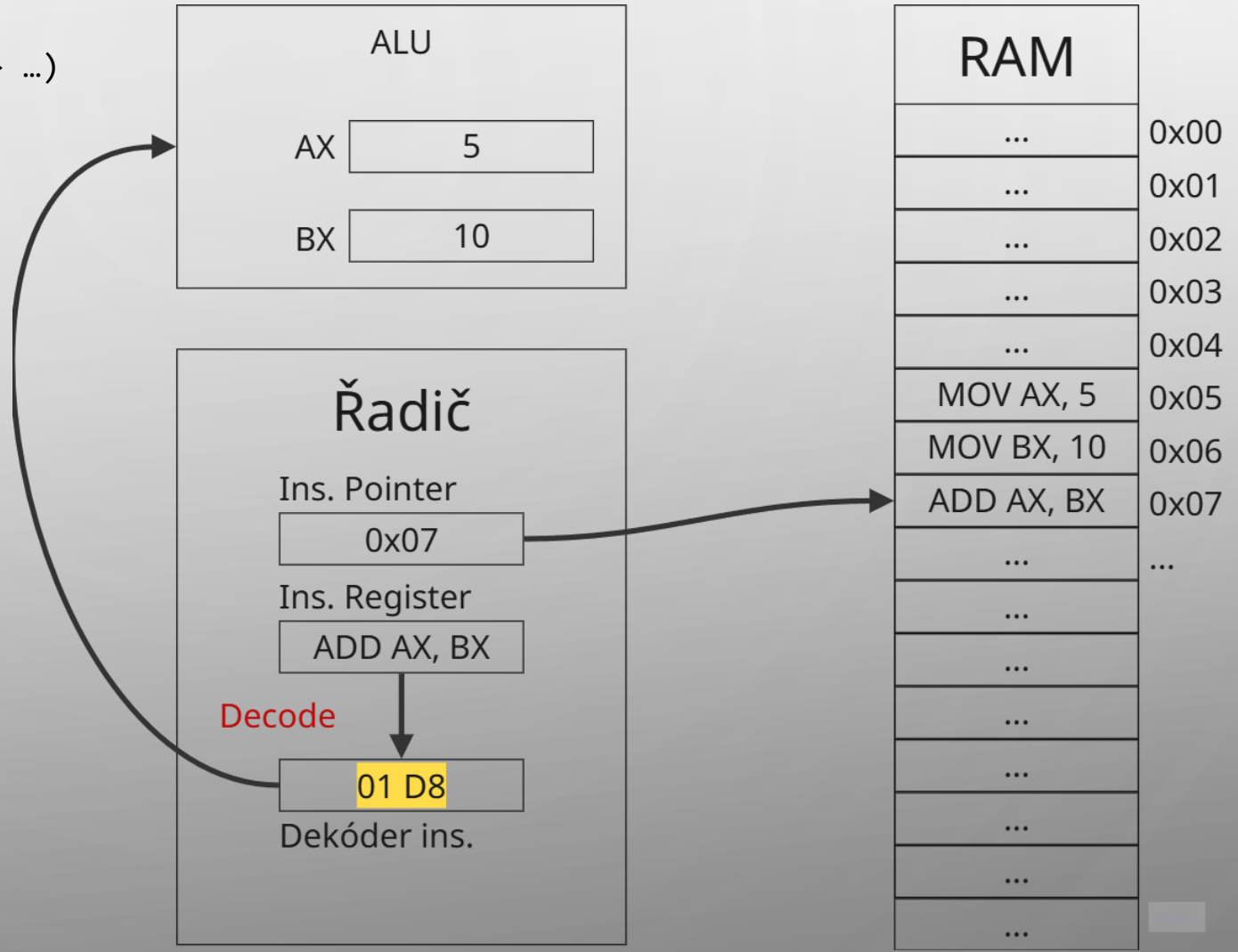
Instrukční cyklus

(… -> Fetch -> Decode -> Execute -> …)



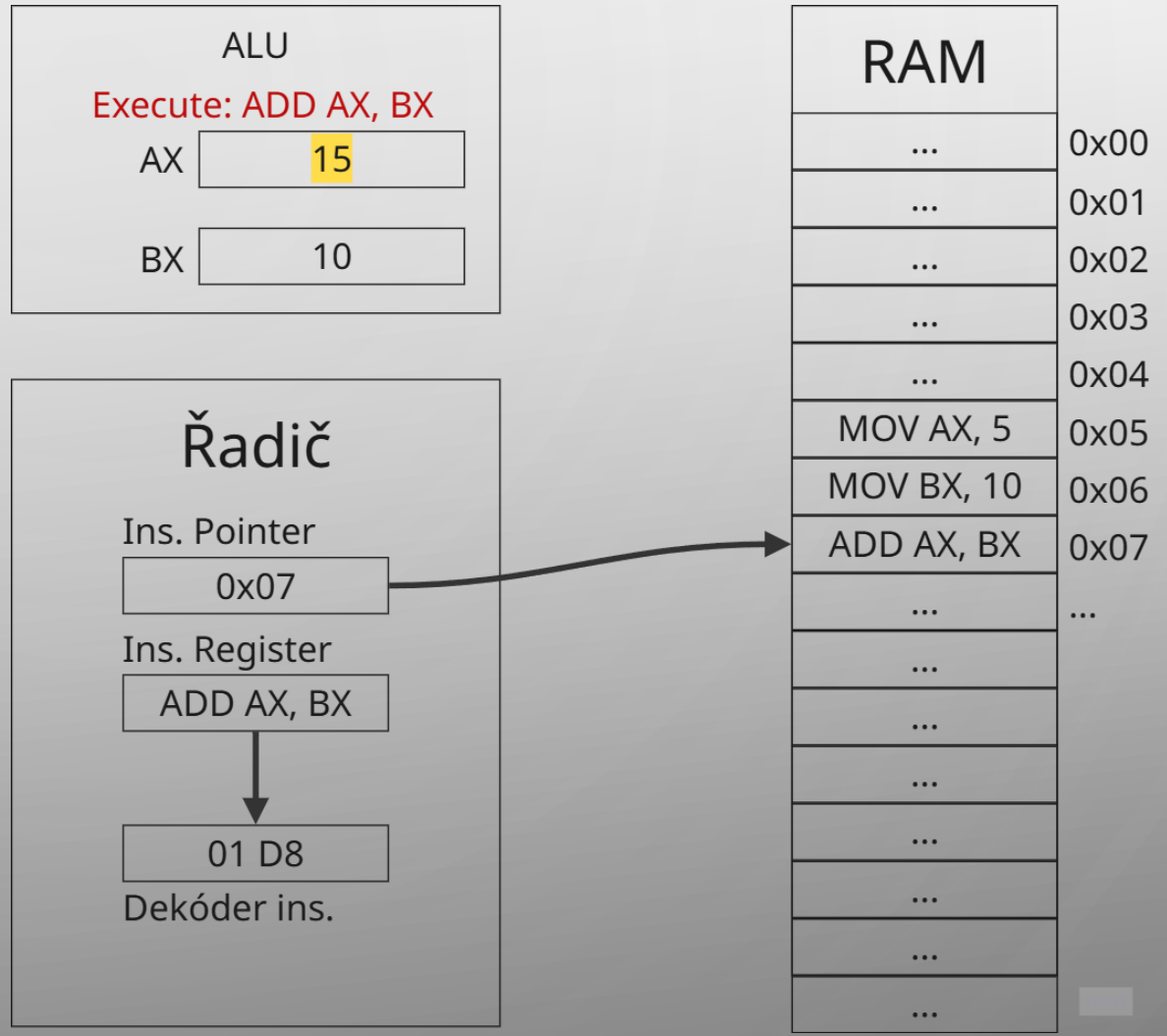
Instrukční cyklus

(… -> Fetch -> Decode -> Execute -> …)



Instrukční cyklus

(… -> Fetch -> Decode -> Execute -> …)



Architektury CPU – Trocha historie

- V 60.–70. letech si výrobci CPU (např. IBM, DEC, Motorola, Intel) navrhovali vlastní procesory s úplně odlišnými instrukčními sadami.
  - tzn. každý procesor mluvil „svým jazykem“ => kód napsaný pro jeden procesor nefungoval na jiném.

- Začátkem 80. let (např. Intel 8086, Motorola 68000) se prosadila snaha o sjednocení a standardizaci.
  - Vznik CISC instrukční sady (tehdy se tak ještě nejmenovala)

CISC (Complex Instruction Set Computer)

- Cílem bylo:
  - Zjednodušit psaní kódu pro programátory
  - Vytvořit univerzální a bohatou instrukční sadu
  - Zvýšit kompatibilitu mezi procesory

- Jedna instrukce = více mikrooperací
  - Např. místo 3 jednoduchých instrukcí šlo najednou napsat jednu složitou a CPU si s tím poradil.
  - CPU musel tuto instrukci rozložit na více kroků => složitější zpracování.

Příklad CISC instrukce (x86):

```
MOV AX, [BX+SI+16]
```

Tato jediná instrukce:

1. Spočítá adresu: BX + SI + 16
2. Načte hodnotu z paměti
3. Zapíše ji do registru AX

CISC (Complex Instruction Set Computer)

Použití:

- Většina PC/NTB (Intel Core i5/i7/i9, AMD Ryzen …)
- Servery (Inten Xeon, AMD EPYC …)
- Historické systémy (Commodore, IBM …)

Výhody

Méně instrukcí v programu

Bohatá funkcionalita

Kompatibilita

Nevýhody

Složitý dekodér

Pomalejší vykonání některých ins.

Instrukce různé délky

Menší možnost optimalizace

Architektury CPU – Trocha historie

- Už v 70. letech výzkumy četnosti výskytu instrukcí ukázaly, že programátoři a kompilátory používají strojové instrukce velmi nerovnoměrně.

Instrukce	Použití	Četnost (%)
LOAD	čtení z paměti	26.6
STORE	zápis do paměti	15.6
Jcond	podmíněný skok	10.0
LOADA	načtení adresy	7.0
SUB	odčítání	5.8
CALL	volání podprogramu	5.3
SLL	bitový posun vlevo	3.6
IC	vložení znaku	3.2

Tabulka: Statická četnost instrukcí

RISC (Reduced Instruction Set Computer)

- V polovině 80. let vznikla myšlenka RISC

- Cílem bylo:
  - Stejná délka instrukcí
  - Vykonání instrukce v jednom hodinovém cyklu
  - Zvýšení efektivity pipeline
  - Redukce složitosti hardwaru (jednodušší dekódování)

- Jedna instrukce = jedna jednoduchá operace
  - Stejná délka instrukce usnadňuje dekódování a zpracování
- Základní sada ins. – S pamětí pracují pouze LOAD a STORE instrukce
- Proto také zvýšení počtu registrů.

RISC (Reduced Instruction Set Computer) - porovnání

Příklad CISC:

```
MOV AX, [BX+SI+16]
```

Tato jediná instrukce:

1. Spočítá adresu: BX + SI + 16
2. Načte hodnotu z paměti
3. Zapíše ji do registru AX

Příklad RISC:

```
ADD R1, BX, SI
ADD R1, R1, #16
LOAD AX, 0(R1)
```

Tato posloupnost instrukcí:

1. Spočítá: R1=BX+SI
2. Přičte: R1=R1+16 (Vznikne adresa)
3. Načte pomocí LOAD do AX hodnotu z předem vypočtené adresy

RISC (Reduced Instruction Set Computer)

Použití:

- Mobilní telefony a tablety (ARM Cortex, Apple Silicon M1/M2/M3)
- Embedded systémy (Raspberry Pi, STM32)

Výhody

Jednoduché instrukce

Rychlé provádění ins.

Lepší pipelining

Efektivnější a menši hardware

Nevýhody

Vyšší nároky na paměť

Optimalizace na straně kompilátoru

Více instrukcí pro stejnou činnost

Srovnání CISC a RISC - Pipeline

CISC Pipeline

- Jakmile se provádí jedna instrukce
- => Nemůže se začít provádět druhá
- => Nepoužívané bloky
- => Nižší efektivita

	T1	T2	T3	T4	T5	T6	T7	T8
VI	i1						i2	
DE		i1						i2
VA			i1					
VO				i1				
PI					i1			
UV						i1		

Legenda:
VI - Výběr instrukce
DE - Dekódování
VA - Výpočet adresy
VO - Výběr oprandu
PI - Provedení instrukce
UV - Uložení výsledku

Srovnání CISC a RISC - Pipeline

RISC Pipeline

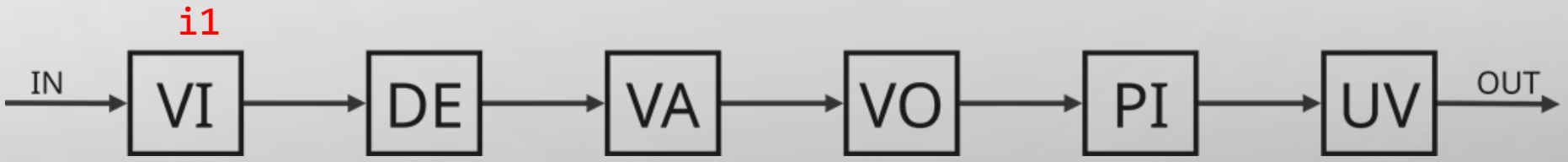
- Jakmile se provádí jedna instrukce
- => Do uvolněného bloku může další
- => „Paralelismus“
- => Vysoká efektivita

	T1	T2	T3	T4	T5	T6	T7	T8
VI	i1	i2	i3	i4	i5	i6	i7	i8
DE		i1	i2	i3	i4	i5	i6	i7
VA			i1	i2	i3	i4	i5	i6
VO				i1	i2	i3	i4	i5
PI					i1	i2	i3	i4
UV						i1	i2	i3

Legenda:
VI - Výběr instrukce
DE - Dekódování
VA - Výpočet adresy
VO - Výběr oprandu
PI - Provedení instrukce
UV - Uložení výsledku

Srovnání CISC a RISC - Pipeline

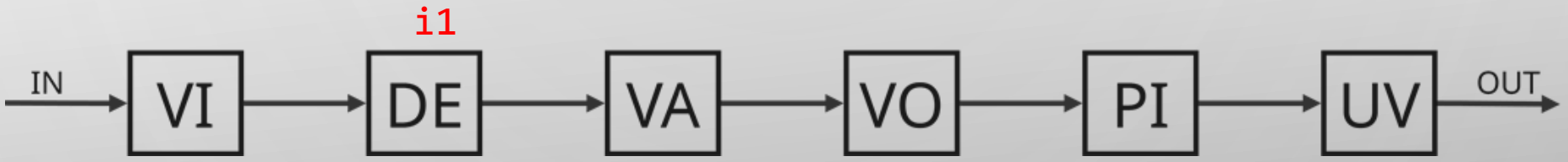
CISC
Čas: T1



Legenda:
VI - Výběr instrukce
DE - Dekódování
VA - Výpočet adresy
VO - Výběr oprandu
PI - Provedení instrukce
UV - Uložení výsledku

Srovnání CISC a RISC - Pipeline

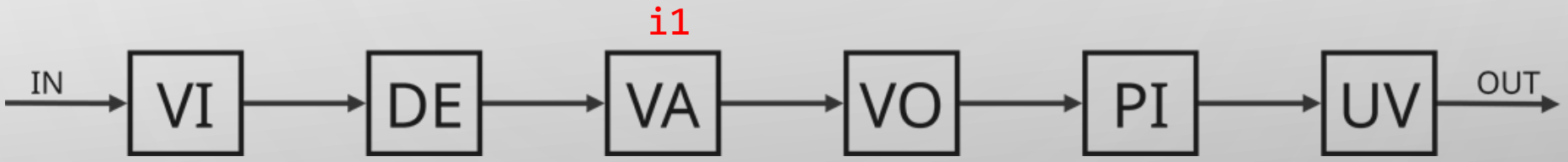
CISC
Čas: T2



Legenda:
VI - Výběr instrukce
DE - Dekódování
VA - Výpočet adresy
VO - Výběr oprandu
PI - Provedení instrukce
UV - Uložení výsledku

Srovnání CISC a RISC - Pipeline

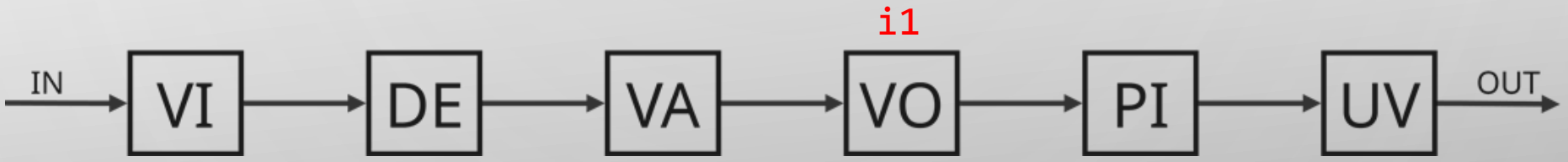
CISC
Čas: T3



Legenda:
VI - Výběr instrukce
DE - Dekódování
VA - Výpočet adresy
VO - Výběr oprandu
PI - Provedení instrukce
UV - Uložení výsledku

Srovnání CISC a RISC - Pipeline

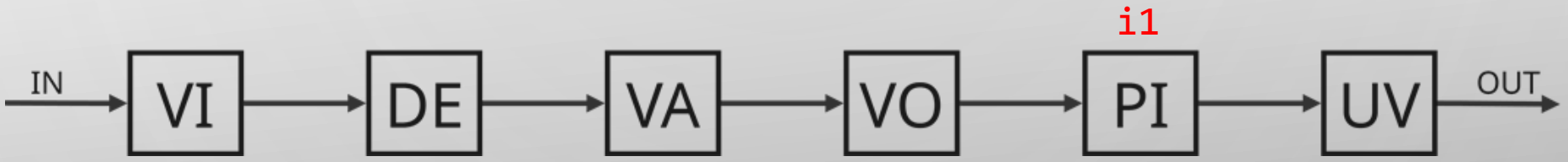
CISC
Čas: T4



Legenda:
VI - Výběr instrukce
DE - Dekódování
VA - Výpočet adresy
VO - Výběr oprandu
PI - Provedení instrukce
UV - Uložení výsledku

Srovnání CISC a RISC - Pipeline

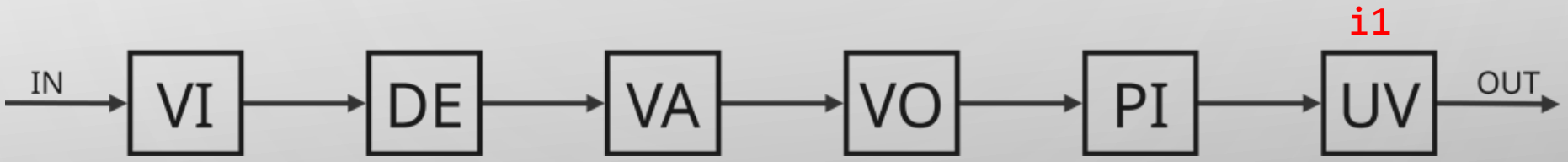
CISC
Čas: T5



Legenda:
VI - Výběr instrukce
DE - Dekódování
VA - Výpočet adresy
VO - Výběr oprandu
PI - Provedení instrukce
UV - Uložení výsledku

Srovnání CISC a RISC - Pipeline

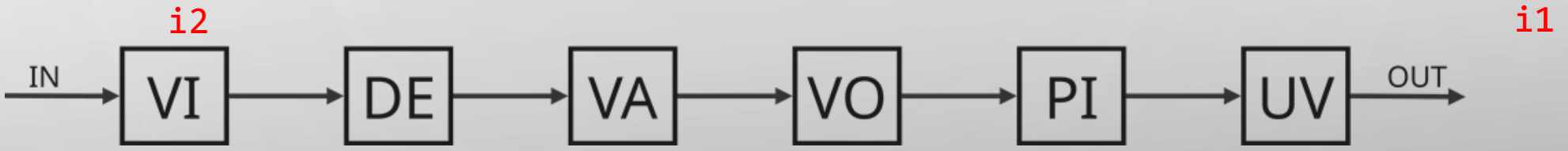
CISC
Čas: T6



Legenda:
VI - Výběr instrukce
DE - Dekódování
VA - Výpočet adresy
VO - Výběr oprandu
PI - Provedení instrukce
UV - Uložení výsledku

Srovnání CISC a RISC - Pipeline

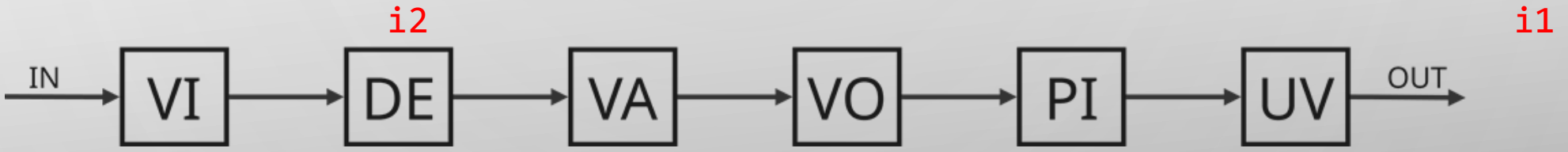
CISC
Čas: T7



Legenda:
VI - Výběr instrukce
DE - Dekódování
VA - Výpočet adresy
VO - Výběr oprandu
PI - Provedení instrukce
UV - Uložení výsledku

Srovnání CISC a RISC - Pipeline

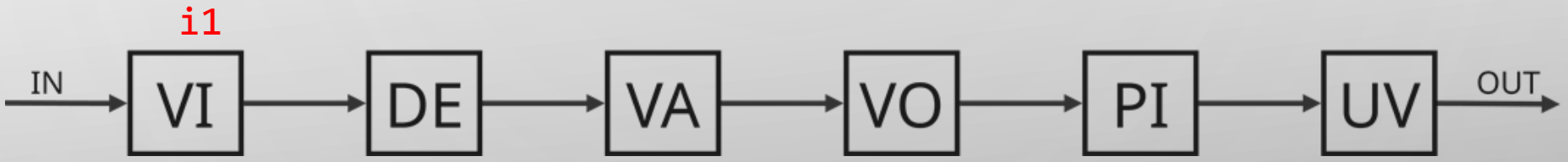
CISC
Čas: T8



Legenda:
VI - Výběr instrukce
DE - Dekódování
VA - Výpočet adresy
VO - Výběr oprandu
PI - Provedení instrukce
UV - Uložení výsledku

Srovnání CISC a RISC - Pipeline

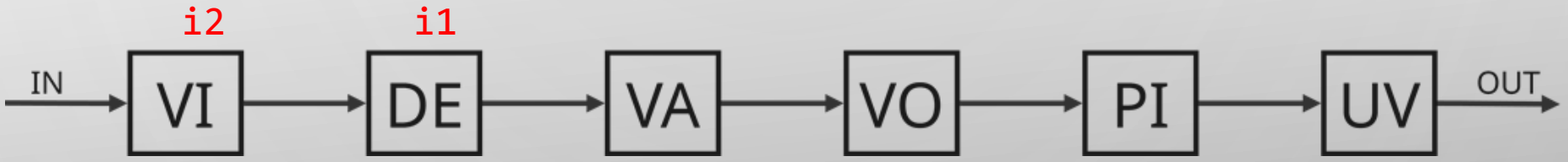
RISC
Čas: T1



Legenda:
VI - Výběr instrukce
DE - Dekódování
VA - Výpočet adresy
VO - Výběr oprandu
PI - Provedení instrukce
UV - Uložení výsledku

Srovnání CISC a RISC - Pipeline

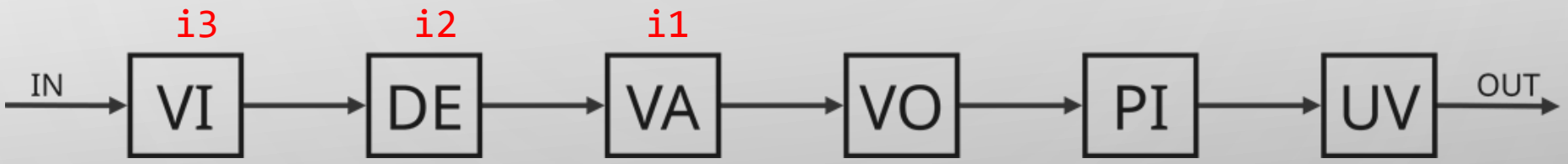
RISC
Čas: T2



Legenda:
VI - Výběr instrukce
DE - Dekódování
VA - Výpočet adresy
VO - Výběr oprandu
PI - Provedení instrukce
UV - Uložení výsledku

Srovnání CISC a RISC - Pipeline

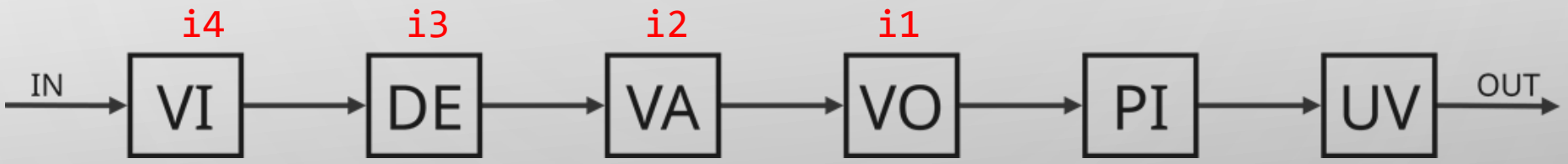
RISC
Čas: T3



Legenda:
VI - Výběr instrukce
DE - Dekódování
VA - Výpočet adresy
VO - Výběr oprandu
PI - Provedení instrukce
UV - Uložení výsledku

Srovnání CISC a RISC - Pipeline

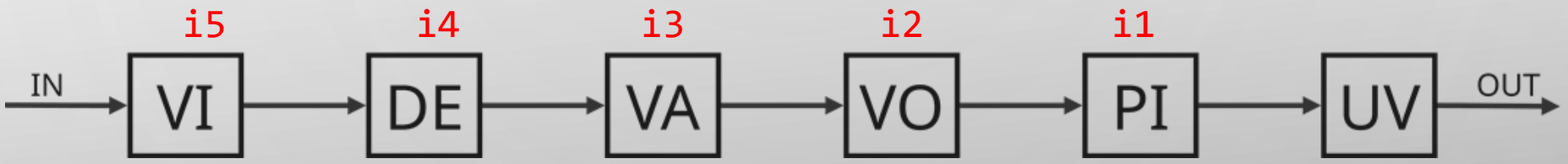
RISC
Čas: T4



Legenda:
VI - Výběr instrukce
DE - Dekódování
VA - Výpočet adresy
VO - Výběr oprandu
PI - Provedení instrukce
UV - Uložení výsledku

Srovnání CISC a RISC - Pipeline

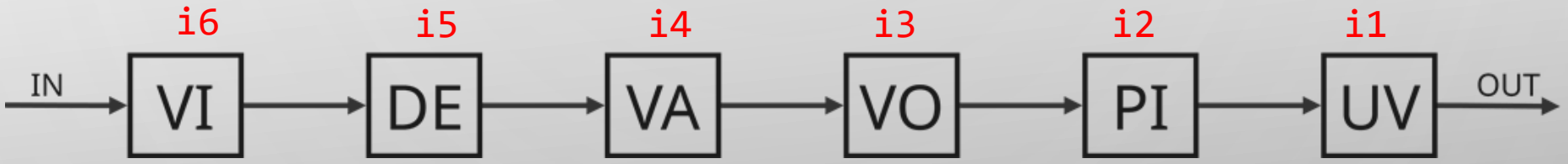
RISC
Čas: T5



Legenda:
VI - Výběr instrukce
DE - Dekódování
VA - Výpočet adresy
VO - Výběr oprandu
PI - Provedení instrukce
UV - Uložení výsledku

Srovnání CISC a RISC - Pipeline

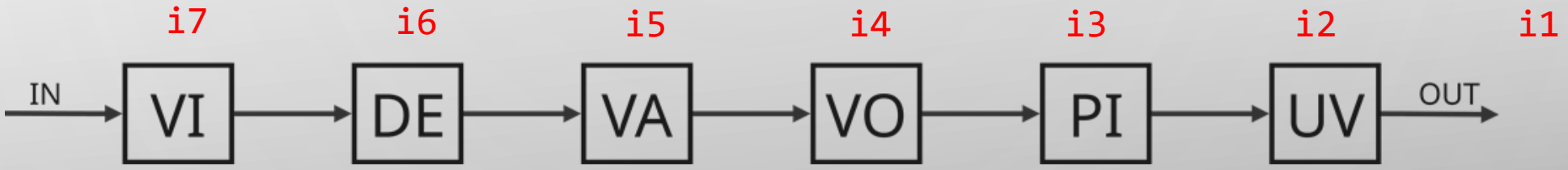
RISC
Čas: T6



Legenda:
VI - Výběr instrukce
DE - Dekódování
VA - Výpočet adresy
VO - Výběr oprandu
PI - Provedení instrukce
UV - Uložení výsledku

Srovnání CISC a RISC - Pipeline

RISC
Čas: T7



Legenda:
VI - Výběr instrukce
DE - Dekódování
VA - Výpočet adresy
VO - Výběr oprandu
PI - Provedení instrukce
UV - Uložení výsledku

Děkuji za pozornost

Prezentaci vytvořil: Hanke Matěj