Pevný a programovatelný řadič. Mikroprogramový automat. Klasická architektura počítače, von Neumannova a harvardská architektura. Struktura CPU, datové a adresní registry, čítač instrukcí, ukazatel zásobníku, typy instrukcí (A0B35SPS)

27.1 Řadiče

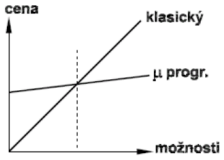
- řadič anglicky control unit
- jasně odlišitelná část systému, která řídí nějaký úkon
  - např.: řadiče displaye, jednotka řídící teplotu vody , atd..
- v CPU se stará o řízení toku dat a o řízení práce všech jednotek, zejména ALU, a to v závislosti na právě vykonávané instrukci

27.1.1 Programovatelný řadič

- varianta sekvenčního obvodu ralizovaná přes paměť
- flexibilní

27.1.2 Pevný řadič = Řadič klasický, též obvodově realizovaný, tedy tzv. obvodový

- rychlejší
- automat realizovaný přes sekvenční obvody
- ve velmi jednoduchých případech levnější

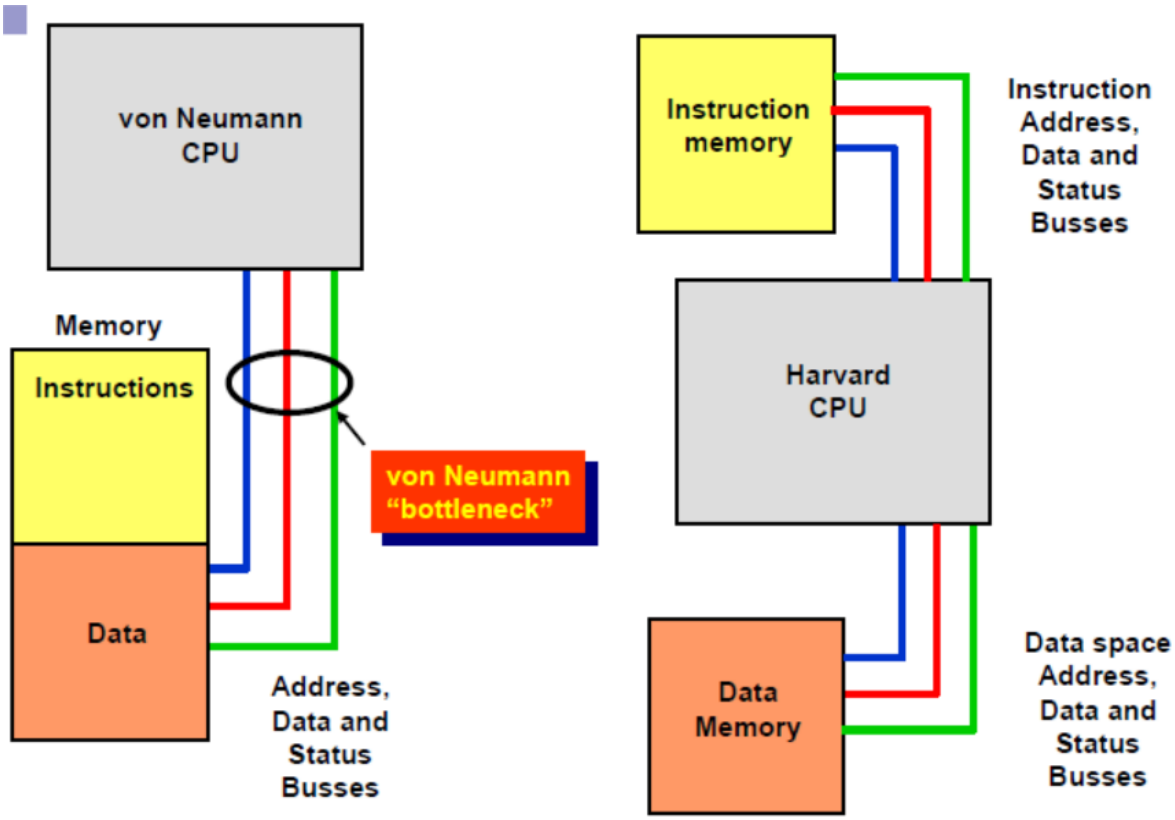


27.2 Mikroprogramovatelný automat

- řadič, který nefunguje s fixní konfigurací, ale používá tzv. mikrokód
- použití v CPU -> instrukce z instrukční sady se překládá na sekvenci mikroinstrukcí, nahrazuje rozsáhlou logiku pevného řadiče
- mikroinstrukce definují, které hardwarové části je potřeba propojit, aby byla vykonána samotná instrukce

- oproti fixní konfiguraci má výhodu v možnosti opravy chyb procesoru pomocí aktualizace tabulky překladu instrukce -> mikroprogram
- mikroinstrukce se provádějí velmi rychle a lze je paralelizovat
- příklad mikroprogramu jedné instrukce přičítání např.:
  - přiveď registr AX k ALU jako první operand
  - přiveď dočasný registr k ALU jako druhý operand
  - nastav ALU do režimu sčítání
  - nastav carry na 0
  - ulož výsledek do regstru AX
  - nastav příznaky
- Tato sekvence je částí vykonávání jedné instrukce. V kontrastu, pevný řadič by operandy ALU dekódoval a logickým součinem povoloval / zakazoval přímo z operačního znaku instrukce. Stejně tak by podle operačního kódu řídil režim činnosti ALU a umístění výsledku. Pro větší počet instrukcí narůstá velikost potřebné logiky.

27.3 Architektura počítače

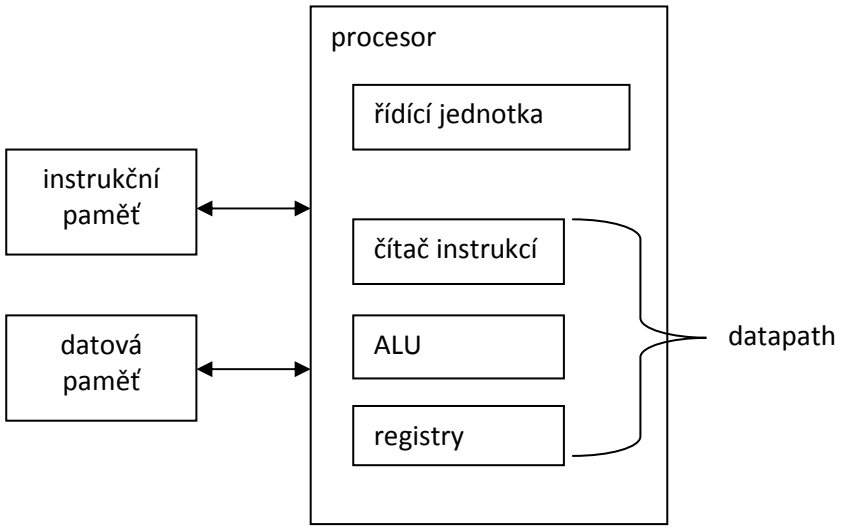


Von_Neumanova jednodušší, pomalejší, možnost rodělit paměť data/instrukce dle potřeby

Harvardská paměť pro instrukce a data fyzicky oddělena

Sběrnice propojující paměti se skládá ze tří sběrnic: adresní, datové, řídící

27.4 Struktura CPU



řídící_jednotka (řadič) zajišťuje součinost jednotlivých částí CPU

ALU aritmeticko-logická jednotka (může jich být i více) zajišťuje všechny aritmetické a logické výpočty

PC program counter - čítač instrukcí - uchovává stav paměti, procesor má vždy po resetu nastavenou určitou hodnotu

registry
- *datové* - ukládání hodnot
- *adresní* - uchovávají adresy odkud mají být dada načítána / kam ukládána
- procesor může vykonávat aritmetické/logické operace pouze nad daty v registrech

27.4.1 ukazatel zásobníku - stack pointer - SP

- uchovává adresu posledního záznamu uloženého na zásobníku
- obvykle "růst dolů": při push se SP zvětší, při pop se SP snižuje
- zásobník pracuje na principu LIFO (last in, first out)
- používá se pro uchování návratové adresy při volání funkcí, nebo zároveň pro ukládání parametrů pro volanou funkci
- sekundární využití je pro ukládání proměnných programu, ke kterým nepřistupujeme instrukcemi PUSH a POP, ale běžnými ukazateli

27.4.2 typy instrukcí

CISC complex instruction set computer

- instrukce se skládá z několika kroků ⟶zpomalování procesoru
- různě dlouhé, různě trvající instrukce

RISC reduced instruction set computing

- snížení počtu instrukcí, snaha dosáhnout 1takt=1instrukce

27.4.3 jeden cyklus cpu - asi není nezbytné

Nevíme, co znamená HP, asi Heap Pointer.

Tak jako tak, prvním krokem je načtení obsahu paměti na adrese PC, asi tím mysleli adresu v hlavní paměti... Tohle je detail.

/newline