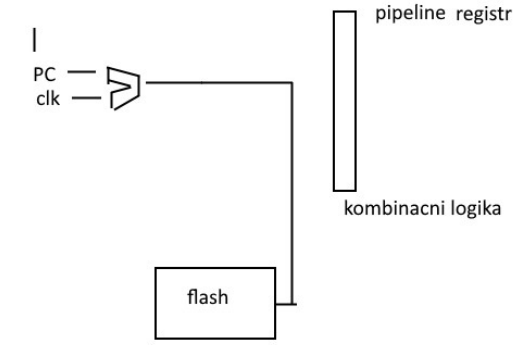
Základní cyklus počítače – MO. 10

Základní cyklus počítače

- skládá se z několika fází:
  - **Fetch**
    - načtení instrukce z paměti
  - **Decode**
    - dekódování instrukce
  - **Execute**
    - provedení operace nebo adresování

------- FETCH --------

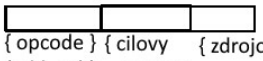
(= umoznuje sekvencne vykonavat program a skakat mezi instrukcemi)
program counter (registr, ktery uchovava adresu prave vykonavane instrukce)



--------- DECODE --------

- decode = zjistime co je to za instrukci
- instrukce je dana instrukcnim formatem

instrukcni format



=> najit vstupy (dane v instrukcnim formatu)

{ opcode } { cilovy registr } { zdrojovy kod }
(add, sub)

---------- EXECUTE --------

- procesor posle dekodovane instrukce jako set kontrolnich signalu k prislusnym komponentum
  - aritmeticko/logicke operace jsou vykonany ALU (jako add, sub, and, or)
  - cteni/zapis z/do pameti je vykonano napriklad pro ukladani nebo nacitani bytu
  - zmeny toku rizeni jsou vykonany napriklad jako jumpovani
    = v tomto kroku, jestli se vykona 'skok', tak misto program counteru, aby se inkrementoval na sousedni pointer, tak skoci na pointer uvedenej v instrukci

- Kolik máme registrů v ATmega16
  - 32 celkem
- Kolik bitů je potřeba k adresování 32 registrů v ATmega16
  - 5 bitů (výpočet log2(32))

- během každého cyklu procesor čte instrukce a provádí operace, což umožňuje vykonávání programů a manipulaci s daty
- základní cyklus se neustále opakuje, zajišťuje neustálé vykonávání instrukcí a řízení toku dat v procesoru

Výjmečné stavy při běhu CPU

- jedním z výjimečných stavů běhu CPU jsou přerušení; může být vyvoláno vnějšími událostmi, jako je například žádost o vstup/výstup, a vyžaduje okamžité přerušení běžícího programu a obsluhu události
- tyto situace se vyskytují během běhu programu, jako je dělení nulou nebo přístup k neplatné paměti
- vyžadují tudíž speciální zacházení a mohou vést k ukončení programu
- tyto výjmečné stavy označují závažné problémy, které mohou způsobit pád systému nebo programu

Operační jednotka

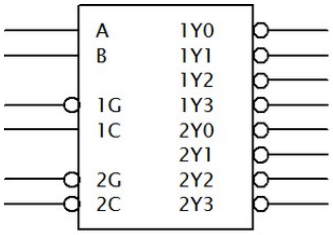
- tato část procesoru provádí aritmetické operace (sčítání, odčítání, násobení, dělení) a logické operace (AND, OR, NOT) nad daty
- může být navržena tak, aby prováděla paralelní výpočty (možnost provádět více operací najednou nebo více částí operace současně), což zvyšuje celkový výkon procesoru
- také může obsahovat speciální instrukce pro práci s čísly s plovoucí řádovou čárkou, logickými operacemi, posuny bitů atd
- tato škála instrukcí poskytuje flexibilitu při zpracování různých typů dat

Řadič

- řadič má za úkol řídit tok dat a instrukcí v procesoru
- interpretuje instrukce z paměti a zajišťuje, aby byly provedeny správně
- také ale řídí registrovou sadu procesoru
- registr je malá paměťová buňka, která uchovává data, na kterých operuje procesor; řadič zajišťuje jejich správné načítání a ukládání

Dekodér

- jeho úkolem je převádět instrukce z instrukční sady procesoru do formy, kterou může operační jednotka interpretovat a provést
- dekodér přebírá binární kód instrukcí a převádí ho do podoby, kterou může operační jednotka interpretovat a provést
- tento proces je klíčový pro správné vykonávání programů a operací
- také zpracovává adresy paměti
- když instrukce vyžaduje přístup k určité části paměti, dekodér identifikuje příslušnou adresu a řídí přístup k danému umístění v paměti



Formát instrukce v RISC architektuře

- **Jednotný formát**
  - *Charakteristika*
    - Pevná délka instrukcí: V RISC architektuře mají všechny instrukce stejnou délku
      - To zjednodušuje proces dekódování, protože hardwarová logika může předpokládat konstantní umístění určitých částí instrukce, jako jsou operační kódy nebo adresy registrů

B	A	1G	1C	1Y0	1Y1	1Y2	1Y3
×	×	1	×	1	1	1	1
×	×	×	0	1	1	1	1
0	0	0	1	0	1	1	1
0	1	0	1	1	0	1	1
1	0	0	1	1	1	0	1
1	1	0	1	1	1	1	0

B	A	2G	2C	2Y0	2Y1	2Y2	2Y3
×	×	1	×	1	1	1	1
×	×	×	1	1	1	1	1
0	0	0	0	0	1	1	1
0	1	0	0	1	0	1	1
1	0	0	0	1	1	0	1
1	1	0	0	1	1	1	0

  - *Výhody*
    - Efektivní pipeline: Pevná délka instrukcí umožňuje procesoru efektivněji využít pipeline
      - Každý krok v pipeline může být optimalizován pro práci s instrukcemi stejné délky, což vede k rychlejšímu zpracování
    - Snadnější dekódování: Jednotný formát znamená, že hardwarový dekodér může být jednodušší, což snižuje složitost čipu a zvyšuje jeho rychlost
- **Rozdělení na pole**
  - *Struktura*
    - Operační kód: Tato část instrukce určuje, jakou operaci má procesor vykonat (např. sčítání, odčítání)
    - Registrační adresy: Určují, které registry jsou použity pro danou operaci
      - RISC systémy typicky používají tři adresy registrů v instrukci – jeden pro výsledek a dva pro operandy
  - *Výhody*
    - Modularita a přehlednost: Každá část instrukce je jasně definovaná a má přesně určenou funkci, což zjednodušuje design hardwaru a software
    - Rychlost dekódování: Díky pevnému uspořádání může procesor rychle extrahovat a dekódovat potřebné informace z instrukce
- **Přímé adresování**
  - *Princip*
    - Práce s registry: Většina operací RISC procesoru se provádí přímo v registrech, namísto práce s pamětí
      - To znamená, že instrukce obvykle zahrnují adresy registrů, ve kterých jsou uložena data
  - *Výhody*
    - Rychlost: Práce s registry je mnohem rychlejší než přístup k hlavní paměti
      - Tím se zvyšuje celková rychlost zpracování instrukcí
    - Efektivita: Přímé adresování minimalizuje počet potřebných instrukcí pro přesun dat mezi pamětí a procesorem, čímž se zvyšuje efektivita zpracování

Operační znak

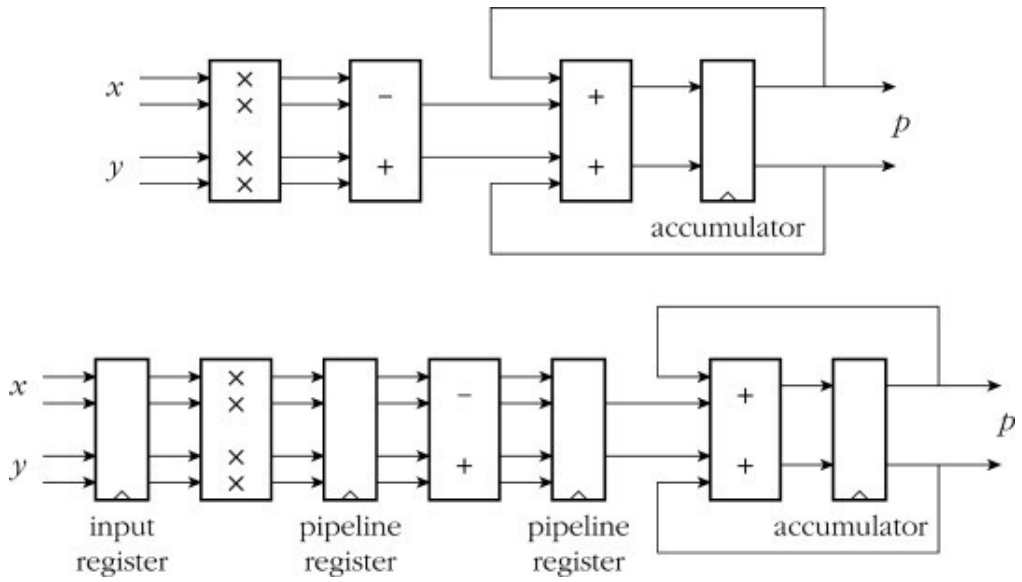
- **Operační kód**
  - Toto je část instrukce, která obsahuje informaci o tom, jaká operace má být provedena V RISC architektuře je operační kód často jednoduchý a jasný, což usnadňuje dekódování instrukcí

Pipeline registry

- **Využití pipeline**
  - RISC procesory často využívají pipelining, což je technika zpracování instrukcí, kde se různé fáze zpracování instrukcí provádějí současně ve více fázích, tzv pipeline

- **Pipeline registry**
  - Tyto registry jsou umístěny mezi jednotlivými fázemi zpracování instrukcí v pipeliningu Umožňují uchování mezivýsledků a řízení toku instrukcí, což zvyšuje efektivitu zpracování

 # RISC architektura se zaměřuje na jednoduchost a efektivitu zpracování instrukcí Tím, že minimalizuje instrukční sadu a využívá pipelining, se snaží dosáhnout vyššího výkonu a rychlejšího zpracování instrukcí



Jednotlivé instrukce

- **Načtení instrukce z paměti (Fetch):**
  - Tato fáze je kritická pro správný chod procesoru
  - Během ní se procesor přesune k paměti a načte binární instrukce, které jsou uloženy
  - Poté je přesune do interních registrů pro další zpracování
  - Efektivita tohoto kroku ovlivňuje rychlost paměti a vzdálenost, kterou musí procesor překonat pro přístup k instrukcím
  - Rychlý přístup a vhodné umístění instrukcí v paměti může výrazně zvýšit výkon procesoru
- **Dekódování instrukce (Decode):**
  - Tato fáze se zaměřuje na interpretaci binárního kódu instrukce a jeho převod do formy, kterou může operační jednotka rozumět a provést
  - Procesor zde extrahuje operační kód, adresování a cílový registr z instrukce
  - Způsob, jakým jsou tyto informace extrahovány a připraveny pro provedení operace v následující fázi, hraje klíčovou roli v efektivitě a rychlosti zpracování instrukcí
- **Provedení operace nebo adresování (Execute):**
  - V této fázi procesor provede samotné operace nad daty, včetně aritmetických, logických nebo adresovacích úkonů podle interpretované instrukce
  - Provedení těchto operací může vyžadovat různé počty cyklů a může být ovlivněno mnoha faktory, jako je složitost operace nebo přístup k externím zdrojům jako je paměť
- **Uložení výsledku (Write Back):**
  - V této poslední fázi se výsledek operace ukládá zpět do paměti nebo do registru, podle potřeby
  - Uložení výsledku do paměti může být zpožděno kvůli různým faktorům, jako je plnění pipeline, a to může ovlivnit rychlost procesu

Cache pamět (úloha)

- Je rychlá paměť umístěná blízko procesoru, která slouží k ukládání často používaných dat a instrukcí pro rychlejší přístup
- V kontextu základního cyklu počítače může významně zlepšit rychlost načtení instrukcí a dat

Kontextové přepínání (multitasking)

- Moderní procesory umožňují multitasking, což je schopnost současně provádět více procesů
- Kontextové přepínání je proces, při kterém CPU mění z jednoho procesu na jiný, uchovává stavy procesů a umožňuje efektivní sdílení procesoru

Rozšíření instrukční sady (specializované instrukce)

- Kromě základních instrukcí může procesor podporovat specializované instrukční sady pro konkrétní úlohy, jako jsou vektorové operace, šifrování, nebo multimediální zpracování

Řízení energie a tepelné správy (energetická efektivita, chlazení)

- Vzhledem k narůstajícímu výpočetnímu výkonu a miniaturizaci je řízení energie a tepelné správy kritické. Techniky jako dynamické škálování frekvence a napětí (např. Intel's Turbo Boost) a sofistikované systémy chlazení jsou klíčové pro udržení stability a efektivity procesoru.

Vliv na celkový výkon systému (interakce s ostatními součastmi)

- Výkon CPU je silně ovlivněn ostatními komponentami systému, jako jsou rychlost a kapacita RAM, rychlost sběrnice, a účinnost úložiště (např. SSD oproti HDD)
- Synergie mezi těmito komponentami je klíčová pro celkový výkon systému

Výjmečné stavy při běhu CPU

- Jedním z významných aspektů jsou výjimečné stavy běhu CPU, jako jsou přerušení a situace jako dělení nulou nebo přístup k neplatné paměti
- Tyto stavy vyžadují speciální zacházení a mohou vést k ukončení programu