Processor

CPU prestanda: $T_{exe} = IC * CPI * T_c$

IC avgörs av ISA, kompilatorn, algoritmen och programspråket.

CPI avgörs av HW.

Instruktionsexekvering

PC → Instruktionsminnet, hämta instruktion

Registernummer → Registerfil, läs reg

Beroende på type:

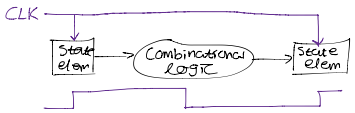
* Använd ALU för att beräkna: Aritmetiska resultat
  Minnesadresser för load/store instr
  Hoppadresser och hoppvillkor
* Läs/skriv dataminnet för load/store instr.
* PC = PC + 4 eller hoppadress

Logikkonstruktorer

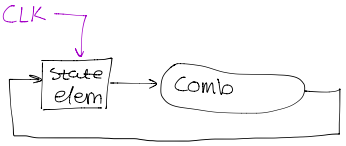
Info kodas binärt

Kombinatoriska element
  Grindar, osv

Tillståndselement
  Lagrar data



Dataväg



Tillståndsmaskin

R-Formcat Instruktioner

Läs två reg operander

Utför aritmetisk/logisk operation

Skriv resultatvärdet till registerfilen

I-format

lw $destreg, offs($basreg)

sw $källreg, offs($basreg)

Behöver teckenutvidga

Branch

beq/bne r_1, r_2, label

Teckenutvidga

Skifta vänster 2-bitpositioner

Addera detta till PC

Icke pipelined

En instruktion per cykel.

Slide Chapter 4 - The processor - 18

Styrenhet

R-type: 0, rs, rt, rd, shamt, funct

L/S 35/43, rs, rt, offset

Jumps

Använder ord-addresser

Uppdatera PC med concat:
  Högsta 4 bitar av PC
  26 bitar från jump-instr

Prestanda

Den längsta signalfördröjningen avgör clockperioden

MIPS Pipeline

Fem pipesteg:

1: IF Instruction Fetch
2: ID Instruction Decode & Reg read
3: EX: Execute instruction eller beräkna adress till DM
4: MEM: Access DM
5: WB: Write Back resultat till reg

Om alla pipesteg är balanserade: Pipeline Speedup = $\frac{T_{c,icke}}{T_{c,Pipe}} \approx$ Antal Pipesteg

Om pipen är obalanserad så är speedup mindre

Speedup pga genomströmning
Latenstiden för en enskild instruktion minskar dock inte

Konflikter (Hazards) i datavägen

Situationer som förhindrar start av instruktion i nästa klockcykel.

- Strukturell: Ett block är upptaget S 34
- Datakonflikt: Föregående instruktion ej klar S 35
- Styrkonflikt: Hoppvillkor och adress evalueras en bit in i pipen. S. 39

Pipeline Summering

☒ **Pipelining ökar prestanda via ökad instruktions- genomströmning**
☒ **Flera instruktioner exekverar parallellt (via överlappning av deloperationer)**
☒ **Pipelinekonflikter (eng. hazards) måste hanteras**
☒ **Strukturkonflikt, datakonflikt (RAW), styrkonflikt**
☒ **Tydligt beroende mellan ISA och komplexitet hos pipeline implementation**

Summering

☒ **Pipelining förbättrar instruktions- genomströmningen via parallellism (överlappning)**
☒ **Fler instruktioner slutförs per sekund**
☒ **Kortare signalvägar möjliggör en högre CPU klockhastighet**
☒ **Hazarder: strukturell, data, styr konflikt**
☒ **Nästa föreläsning: Detaljerad styrning av pipelinen, samt Multiple-Issue och Spekulativa processorer**