

## Pipelining

Yuri Kaszubowski Lopes

UDESC

Anotações

---

---

---

---

---

---

---

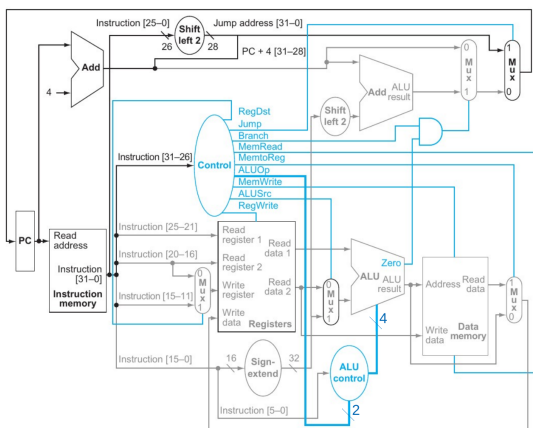
---

YKL (UDESC)

Pipelining

1 / 22

## Duração das instruções



YKL (UDESC)

Pipelining

2 / 22

Anotações

---

---

---

---

---

---

---

---

## Duração das instruções

- Qual tipo de instrução nos toma mais tempo?
  - Provavelmente instruções que lidam com a memória, principalmente loads
    - Passam por cinco unidades funcionais em série: Memória de instruções, banco de registradores, ALU, memória de dados, banco de registradores
- E qual tipo mais rápido?
  - Provavelmente os jumps
    - Após a memória de instruções, basta fazer o shift left do imediato, e concatenar com o PC

Anotações

---

---

---

---

---

---

---

---

YKL (UDESC)

Pipelining

3 / 22

Duração das instruções

- Temos instruções que em teoria levam tempos diferentes para serem executadas
- Considere um exemplo com o tempo gasto por cada componente (busca de instruções, banco de registradores, operação na ALU, ...) em picossegundos
  - ▶  $1\text{ ps} = 1^{-12}$  segundos

Instruction class	Instruction fetch	Register read	ALU operation	Data access	Register write	Total time
Load word (lw)	200 ps	100 ps	200 ps	200 ps	100 ps	800 ps
Store word (sw)	200 ps	100 ps	200 ps	200 ps		700 ps
R-format (add, sub, AND, OR, slt)	200 ps	100 ps	200 ps		100 ps	600 ps
Branch (beq)	200 ps	100 ps	200 ps			500 ps

Anotações

---

---

---

---

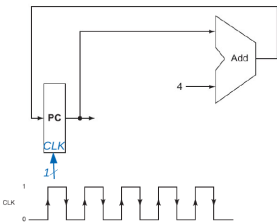
---

---

---

Problemas do ciclo único

- Componentes de estado são sincronizados por um sinal de clock
- O clock deve ser longo o suficiente para dar tempo da instrução ser executada
- Devemos então considerar o pior caso, e ter um período de clock de 800ps (no exemplo)
  - ▶ Esperar 800ps para carregar a próxima instrução
  - ▶ Se a instrução “não necessita” todos os 800ps, jogamos tempo fora



Anotações

---

---

---

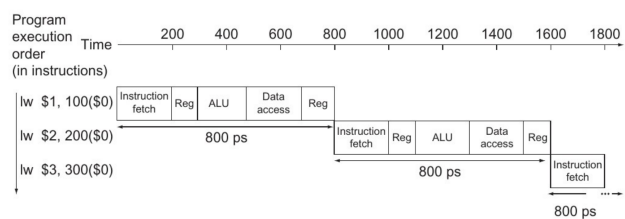
---

---

---

---

Ordem de execução em ciclo único



Anotações

---

---

---

---

---

---

---

Onde ciclo único é usado

- Processadores extremamente simples podem usar ciclo único (talvez como a nossa versão do processador MIPS)
  - Poucas instruções, e o caminho de dados completo é curto
  - Impacto no tempo de ciclo pode ser relativamente pequeno
- Mas se adicionarmos mais complexidade ao nosso circuito, um tamanho de ciclo que considera o pior caso se torna impraticável
  - e.g., unidades para ponto flutuante

Anotações

---

---

---

---

---

---

---

Pipelining

- A técnica de pipelining é utilizada em praticamente todos processadores atuais
- Transformamos o processador em uma “linha de montagem”
- Vamos começar com uma analogia de exemplo onde precisamos “lavar roupas”
  - Temos quatro estágios
    - ★ Lavadora de roupas
    - ★ Secadora de roupas
    - ★ Mesa de passar roupas
    - ★ Armário

Anotações

---

---

---

---

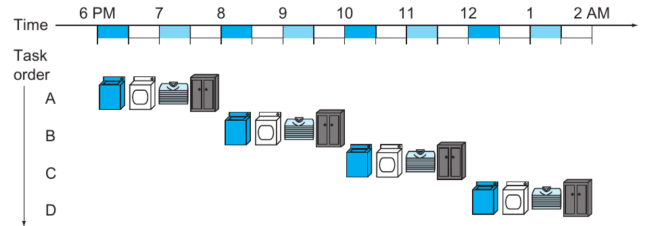
---

---

---

Lavando roupas sem um pipeline

- Considere que temos 4 trouxas de roupa (A, B, C e D) para lavar, e que cada estágio (e.g., lavar a roupa) demora 30 minutos
- Sem aumentar os recursos (ainda temos uma lavadora, uma secadora, ...), como você poderia otimizar esse processo?



Anotações

---

---

---

---

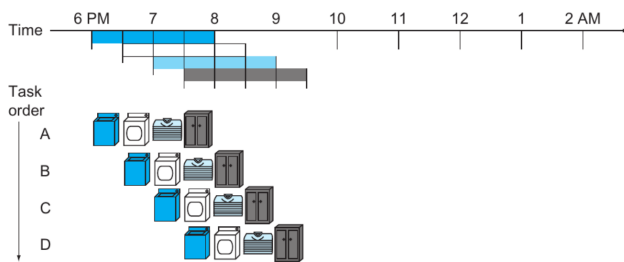
---

---

---

## Lavando roupas com um pipeline

- Após a máquina de lavar terminar a trouxa A, transfere-se a trouxa para a secadora
- A próxima trouxa já é inserida na máquina de lavar, enquanto a secadora continua o processo com a trouxa anterior
  - ▶ A máquina de lavar nunca fica ociosa
  - ▶ Repetimos o processo para os demais componentes



YKL (UDESC)

Pipelining

10 / 22

Anotações

---

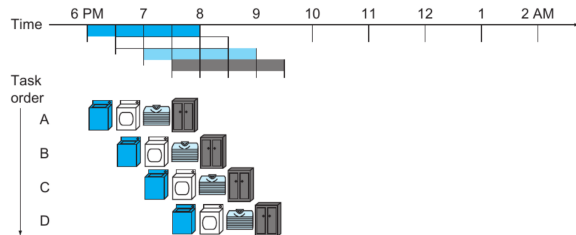
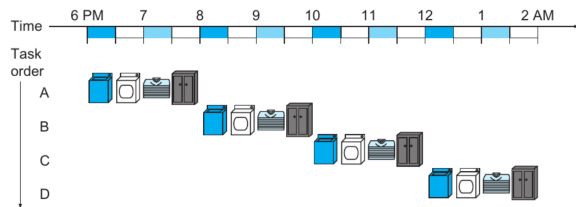
---

---

---

---

---



YKL (UDESC)

Pipelining

11 / 22

Anotações

---

---

---

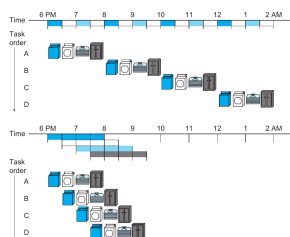
---

---

---

## Onde está o ganho?

- O tempo para executar uma tarefa completa muda?
  - ▶ Não. No exemplo, lavar uma trouxa de roupas ainda demora 2 horas.
  - ▶ Se nosso objetivo fosse lavar uma única trouxa de roupas, não haveria ganho algum.
- Onde está o ganho de tempo?
  - ▶ Temos múltiplas trouxas sendo "processadas" em paralelo, cada uma em um estágio



YKL (UDESC)

Pipelining

12 / 22

Anotações

---

---

---

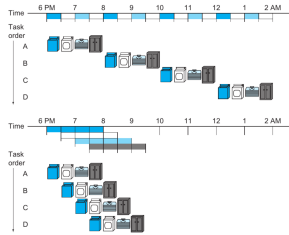
---

---

---

## Onde está o ganho?

- Considerando que todos os estágios duram o mesmo tempo, qual o ganho de tempo se dividirmos o processo em  $n$  estágios (no exemplo  $n=4$ )?
  - Temos o potencial para executar nossas tarefas  $n$  vezes mais rápido
  - Mas para isso nosso pipeline sempre deve estar cheio
  - No início das tarefas, e no fim, sempre temos “unidades” vazias



## Anotações

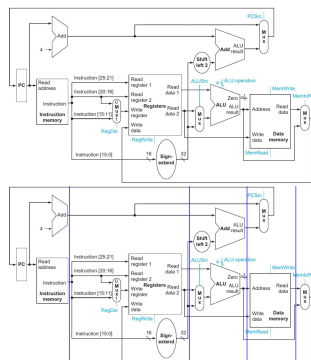
## Pipeline

- Não aumentamos o tamanho de nossa lavanderia
- Se olharmos para uma única tarefa, do início ao fim, o tempo para sua execução não muda
  - O tempo de execução de uma instrução “não muda” (não é bem assim ... detalhes na sequência)
  - O tempo gasto para uma instrução individual também é chamado de **latência**
- Mas se olharmos um período de tempo suficientemente longo, o número de tarefas completadas nesse tempo é muito maior
  - A **vazão (throughput)** pode aumentar em até  $n$  vezes

## Anotações

## E na nossa CPU MIPS?

- Vamos aplicar a ideia da lavadora de roupas no processador MIPS
- Quais são os estágios?
  - Buscar a instrução na memória
    - Instruction fetch (IF)
  - Ler os registradores enquanto a unidade de controle a decodifica
    - Instruction decoding (ID)
  - Executar a operação/cálculo do enredo
    - Execution (EX)
  - Acessar a memória de dados
    - Memory Access (MEM)
  - Escrever o resultado num registrador
    - Write Back (WB)



## Anotações

Estágios do Pipeline

- Nosso processador pode ser então dividido em 5 estágios
  - A quantidade de estágios depende diretamente da arquitetura do processador

Microprocessador	Year	Clock Rate	Pipeline Stages	Issue Width	Out-of-Order/Speculation	Cores/Chip	Power
Intel 486	1989	25 MHz	5	1	No	1	5 W
Intel Pentium	1993	66 MHz	5	2	No	1	10 W
Intel Pentium Pro	1997	200 MHz	10	3	Yes	1	29 W
Intel Pentium 4 Willamette	2001	2000 MHz	22	3	Yes	1	75 W
Intel Pentium 4 Prescott	2004	3600 MHz	31	3	Yes	1	103 W
Intel Core	2006	2930 MHz	14	4	Yes	2	75 W
Intel Core i5 Nehalem	2010	3300 MHz	14	4	Yes	1	87 W
Intel Core i5 Ivy Bridge	2012	3400 MHz	14	4	Yes	8	77 W

Anotações

---

---

---

---

---

---

---

---

Ajustando o tempo de clock

- Veja mais uma vez o tempo gasto em cada unidade funcional
- Cada unidade nos representa um estágio no pipeline
- Tempo de clock:
  - Sem pipeline, somamos tudo e ajustamos para o pior caso (800ps)
  - Dado que o clock é fixo, qual o tempo de clock se adicionarmos o pipeline?
    - ★ Com o pipeline, ajustamos para o estágio de pior caso
    - ★ Os estágios mais demorados precisam de 200ps
    - ★ Logo, o tempo de clock deve ser de no mínimo 200ps
- Já temos uma limitação
  - O ganho seria de n vezes o número de estágios se todos durassem o mesmo tempo

Instruction class	Instruction fetch	Register read	ALU operation	Data access	Register write	Total time
Load word (lw)	200 ps	100 ps	200 ps	200 ps	100 ps	800 ps
Store word (sw)	200 ps	100 ps	200 ps	200 ps		700 ps
R-format (add, sub, AND, OR, slt)	200 ps	100 ps	200 ps		100 ps	600 ps
Branch (beq)	200 ps	100 ps	200 ps			500 ps

Anotações

---

---

---

---

---

---

---

---

Ajustando o tempo de clock

- Na prática, nosso ganho vai ser:

proporção do aumento de vazão =  $\frac{\text{Tempo do ciclo sem pipeline}}{\text{Tempo do ciclo para a unidade mais lenta}}$

- Em nosso processador MIPS com os números do exemplo:

proporção do aumento de vazão =  $\frac{800}{200} = 4$

- 5 estágios, potencial para até 5 vezes mais rápido
- porém, neste caso é 4 vezes mais rápido

Anotações

---

---

---

---

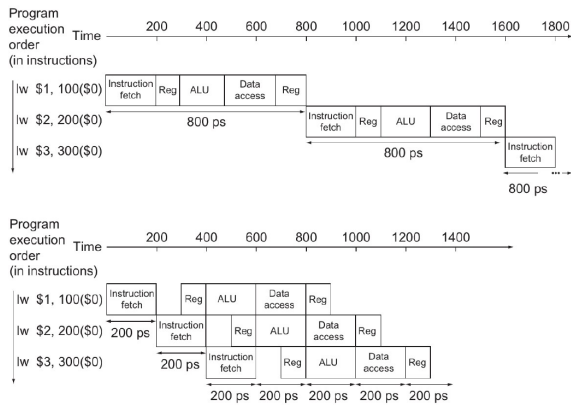
---

---

---

---

## Ciclo único versus Pipeline



### Anotações

## Exercícios

- Considere que criamos um pipeline em um processador qualquer. Ao olhar para o tempo de execução de **uma única instrução (latência)** na versão com pipeline desse processador, esse tempo:
  - Pode ser menor que o tempo do processador sem pipeline?
  - Pode ser igual o tempo do processador sem pipeline?
  - Pode ser maior que o tempo do processador sem pipeline?Confirme ou refute cada uma das afirmações, explicando detalhadamente.
- Toda instrução no MIPS possui o mesmo tamanho. Como isso nos ajuda no pipeline?
  - Se as instruções fossem de tamanho variável, em qual estágio do pipeline precisaríamos estar para só então definir onde está a próxima instrução?
  - Observação: seu processador x86-64 possui instruções de tamanho variável, e precisa tratar essa complexidade extra.

### Anotações

## Referências

- D. Patterson; J. Henessy. **Organização e Projeto de Computadores: Interface Hardware/Software**. 5a Edição. Elsevier Brasil, 2017.
- Andrew S. Tanenbaum. **Organização estruturada de computadores**. 5. ed. São Paulo: Pearson, 2007.
- Harris, D. and Harris, S. **Digital Design and Computer Architecture**. 2a ed. 2012.
- [courses.missouristate.edu/KenVollmar/mars/](https://courses.missouristate.edu/KenVollmar/mars/)

### Anotações