Homens são de Marte, mulheres são de Vênus, e computadores são do inferno.

Pipelining

Paulo Ricardo Lisboa de Almeida

DSBD

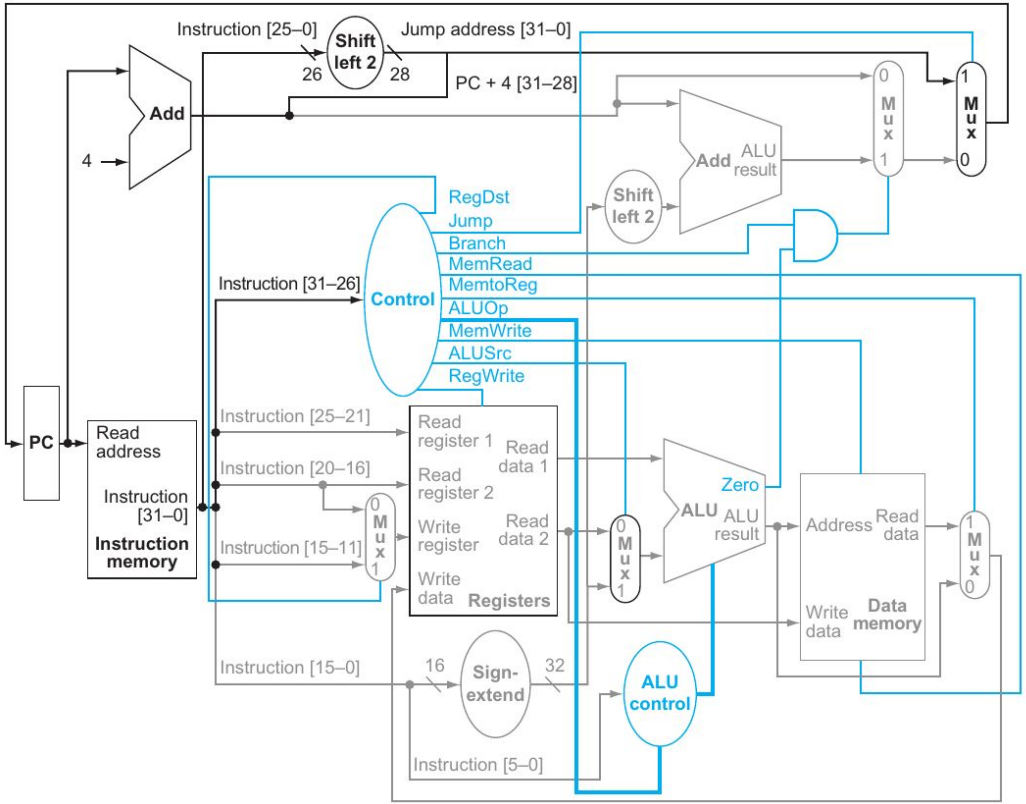
UFPR
UNIVERSIDADE FEDERAL DO PARANÁ

O modelo de ciclo único não é viável!

Pergunta

Qual tipo de instrução nos toma mais tempo (precisa passar por mais lugares)?

E qual tipo mais rápido?



Duração das instruções

Qual tipo de instrução toma mais tempo?

- Provavelmente instruções que lidam com a memória (ex.: loads).
- Passam por cinco unidades funcionais em série:
  - Memória de instruções, banco de registradores, ALU, memória de dados e 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.

Duração das instruções

Temos instruções que em teoria levam tempos diferentes para serem executadas.

Qual o problema com isso?

Duração das instruções

Temos instruções que em teoria levam tempos diferentes para serem executadas.

Qual o problema com isso?

Considere um exemplo com o tempo gasto por cada componente (busca de instruções, banco de registradores, operação na ALU, ...) em picossegundos (1ps = 1 x 10^{-12} segundos)

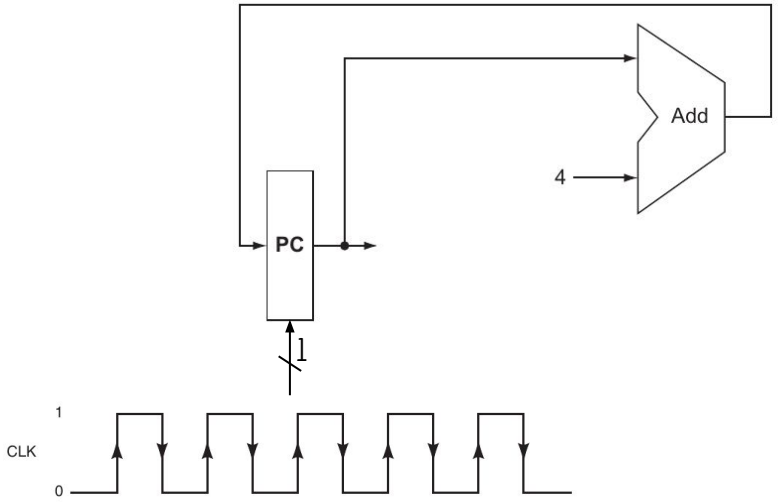
Instrução	Instruction Fetch	Register Read	Operação ALU	Data Access	Register Write	Total
LW	200ps	100ps	200ps	200ps	100ps	800ps
SW	200ps	100ps	200ps	200ps	100ps	700ps
Tipo -R	200ps	100ps	200ps			600ps
Branch	200ps	100ps	200ps			500ps

Problemas do ciclo único

Componentes de estado são sincronizados por um sinal de clock.

Qual deve ser o período de clock?

Instrução	Instruction Fetch	Register Read	Operação ALU	Data Access	Register Write	Total
LW	200ps	100ps	200ps	200ps	100ps	800ps
SW	200ps	100ps	200ps	200ps	100ps	700ps
Tipo -R	200ps	100ps	200ps			600ps
Branch	200ps	100ps	200ps			500ps



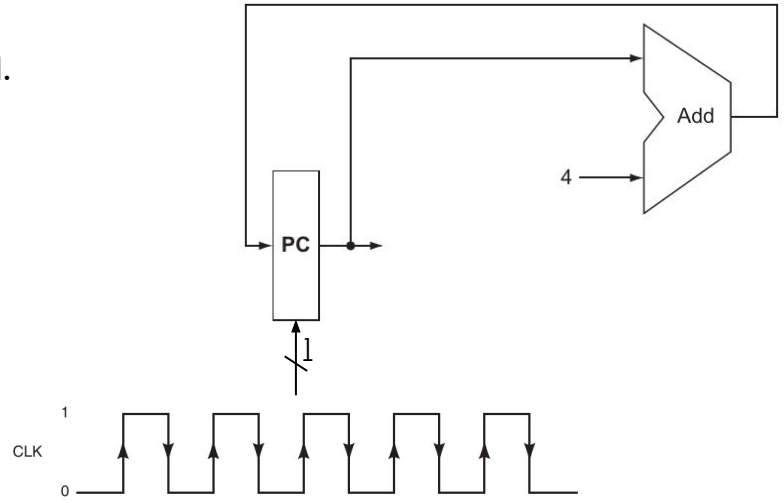
Problemas do ciclo único

Componentes de estado são sincronizados por um sinal de clock.

O clock deve ser longo o suficiente para dar tempo de qualquer instrução ser executada.

- 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 utilizar todos os 800ps, jogamos tempo fora.

Instrução	Instruction Fetch	Register Read	Operação ALU	Data Access	Register Write	Total
LW	200ps	100ps	200ps	200ps	100ps	800ps
SW	200ps	100ps	200ps	200ps	100ps	700ps
Tipo -R	200ps	100ps	200ps			600ps
Branch	200ps	100ps	200ps			500ps



Problemas do ciclo único

Componentes de estado são sincronizados por um sinal de clock.

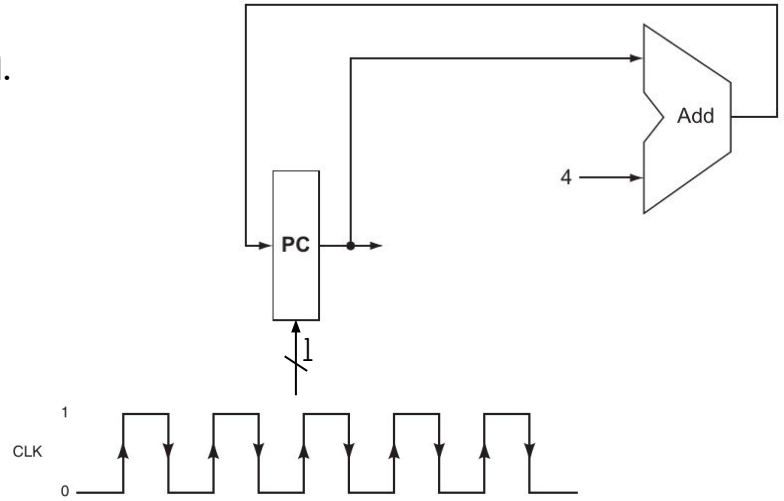
O clock deve ser longo o suficiente para dar tempo de qualquer instrução ser executada.

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 utilizar todos os 800ps, jogamos tempo fora.

Qual a frequência?



Problemas do ciclo único

Componentes de estado são sincronizados por um sinal de clock.

O clock deve ser longo o suficiente para dar tempo de qualquer instrução ser executada.

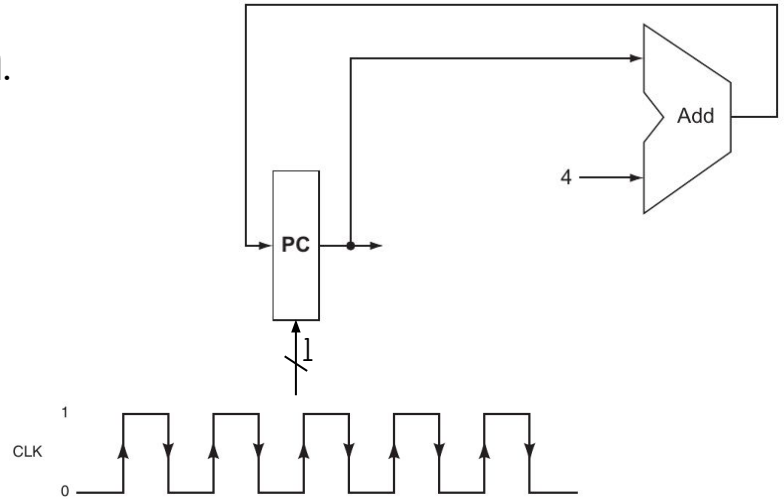
Considerar o pior caso, e ter um período de clock de 800ps (no exemplo).

Esperar 800ps para carregar a próxima instrução.

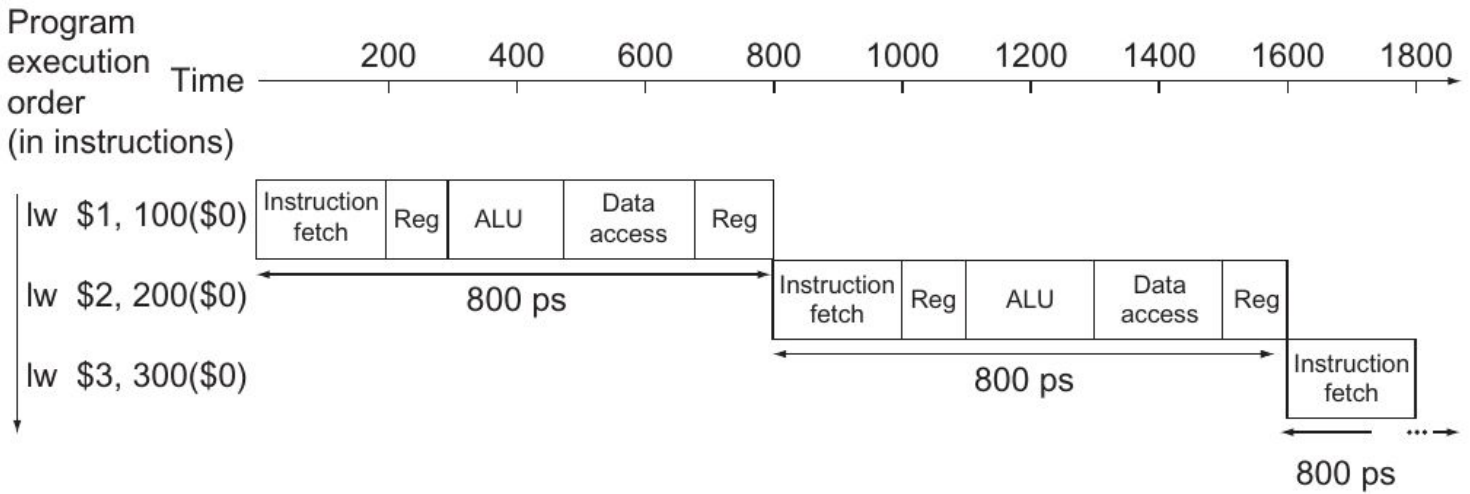
Se a instrução não utilizar todos os 800ps, jogamos tempo fora.

Qual a frequência?

$F = 1/(800*10^{-12}) = 1{,}25*10^{9}$ Hz = 1,25GHz



Exemplo - Ordem de execução em ciclo único



Onde usar o ciclo único

Processadores extremamente simples (talvez como o processador MIPS do exemplo) podem usar ciclo único.

- Poucas instruções, e o caminho de dados completo é curto.

O impacto no período de clock é relativamente pequeno.

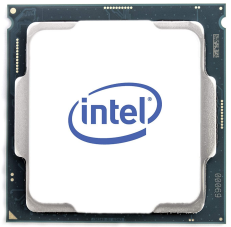
Mas ao adicionar mais complexidade ao circuito, um tamanho de ciclo que considera o pior caso se torna **impraticável**.

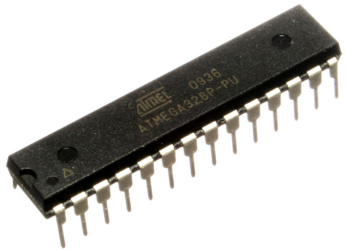
- Exemplo: unidades para ponto flutuante.

Pipelining

A técnica de pipelining é utilizada **em praticamente todos processadores atuais**.

Desde o seu AMD/Intel do notebook ou desktop, até microcontroladores.

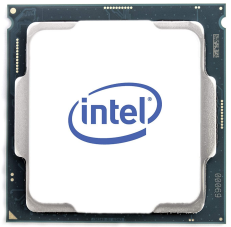


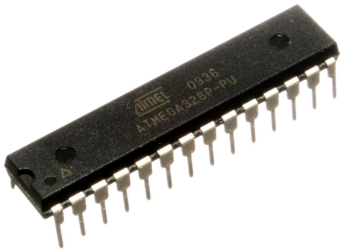


Pipelining

A técnica de pipelining é utilizada **em praticamente todos processadores atuais.**

Desde o seu AMD/Intel do notebook ou desktop, até microcontroladores.

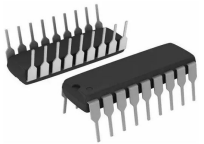




ATMEGA328p
Usado nos Arduinos
e na Indústria automotiva.
Pipeline de 2 estágios.

R$ 53⁵⁰
em 6x R$ 8⁹² sem juros
CI Atmega328p-pu

PIC16F628a
Automação e
Indústria Automotiva.
Pipeline de 2 estágios.



R$ 26⁷⁶
em 5x R$ 5³⁵ sem juros
Microcontrolador Pic16f628a 16f628a
Original Microchip Pic

Pipelining

Transformar o processador em uma “linha de montagem”.

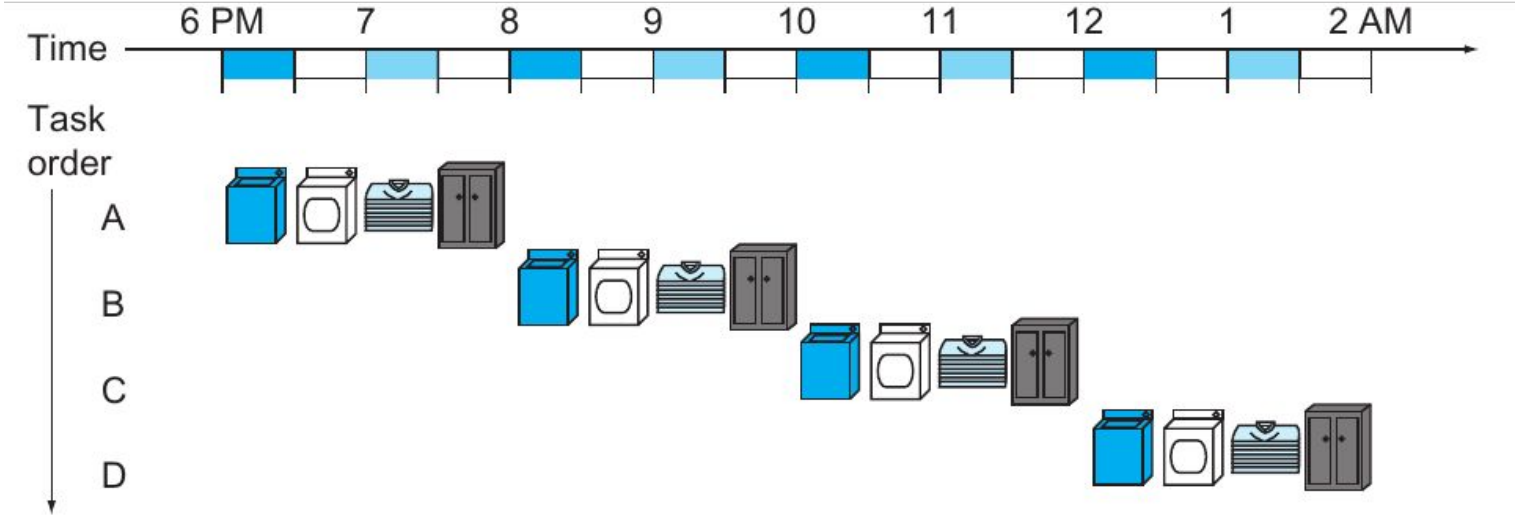
Vamos começar com um exemplo onde precisamos lavar roupas.

Temos quatro estágios:

1. Lavadora de roupas;
2. Secadora de roupas;
3. Mesa de passar roupas;
4. Armário.

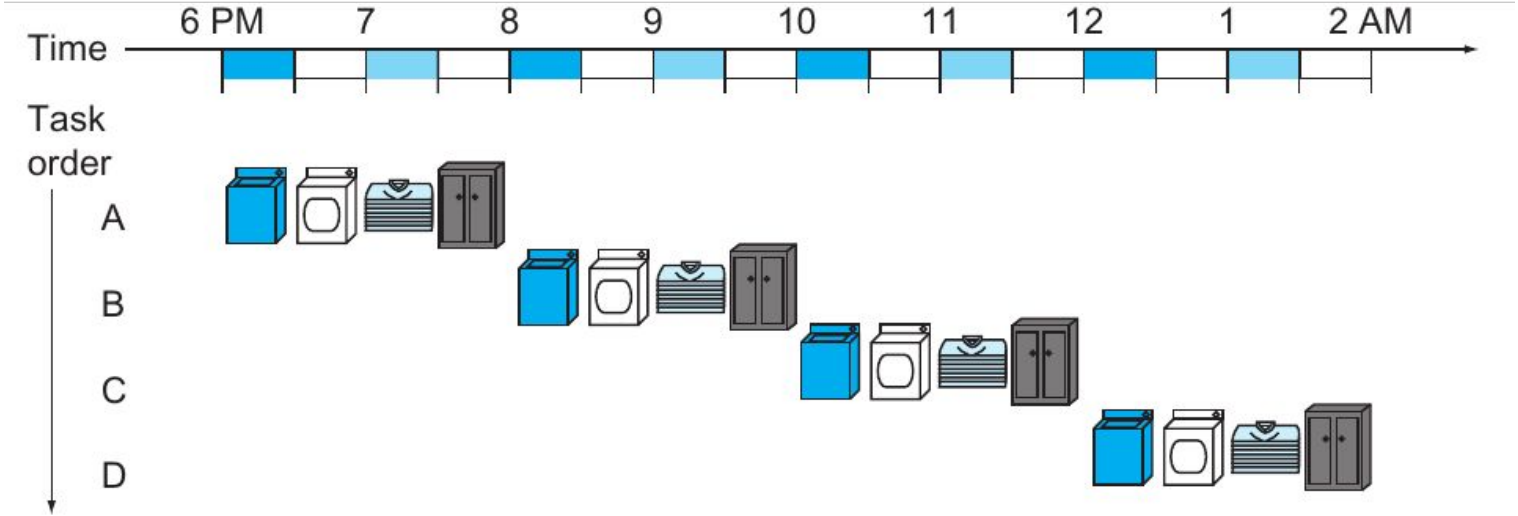
Lavando roupas sem um pipeline

Considere que temos 4 trouxas de roupa (A, B, C e D) para lavar, e que cada estágio (ex.: lavar a roupa) demora meia hora.



Otimizando

Sem aumentar os recursos – ainda temos **uma** lavadora, **uma** secadora, ..., como otimizar esse processo?



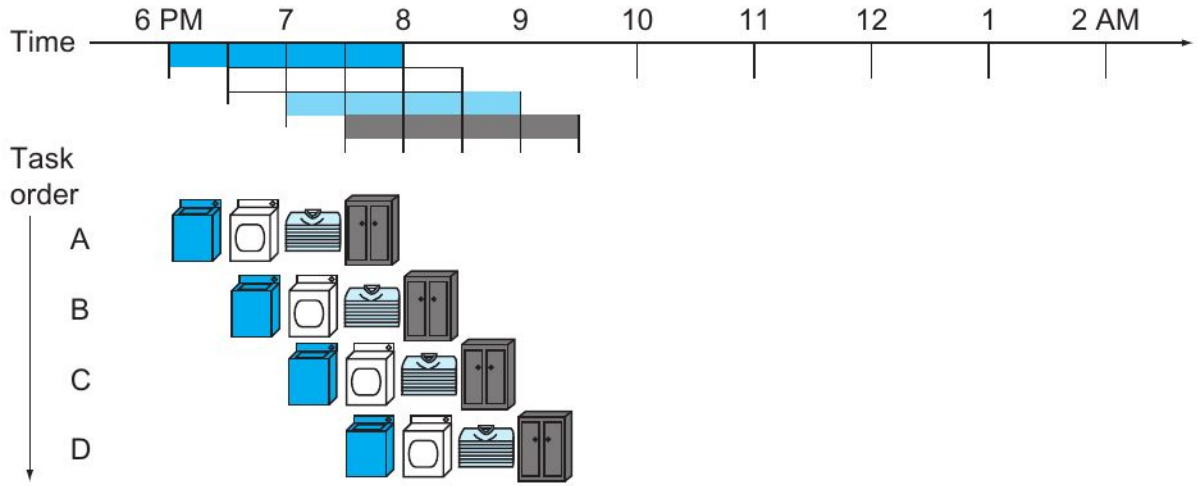
Adicionando um Pipeline

Após a máquina de lavar terminar a trouxa A, transfere-se a trouxa para a secadora.

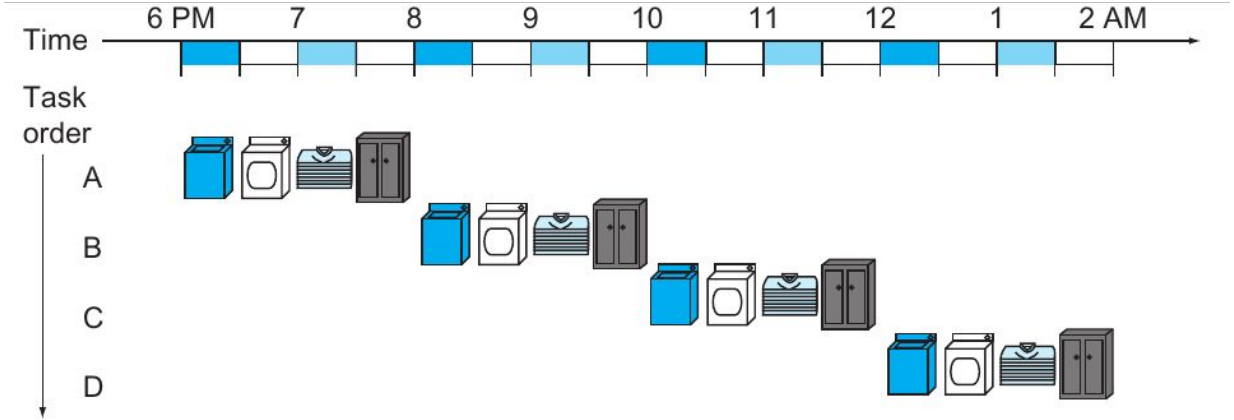
A próxima trouxa é inserida na máquina de lavar, enquanto a secadora continua o processo com a trouxa anterior.

A máquina de lavar (quase) nunca fica ociosa.

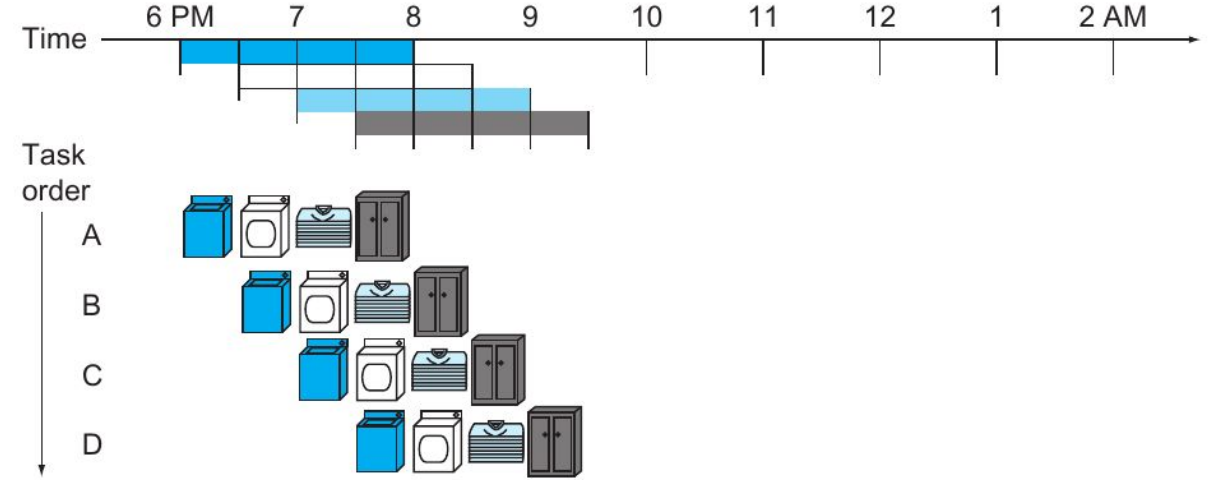
Repetimos o processo para os demais componentes.



Sem Pipeline

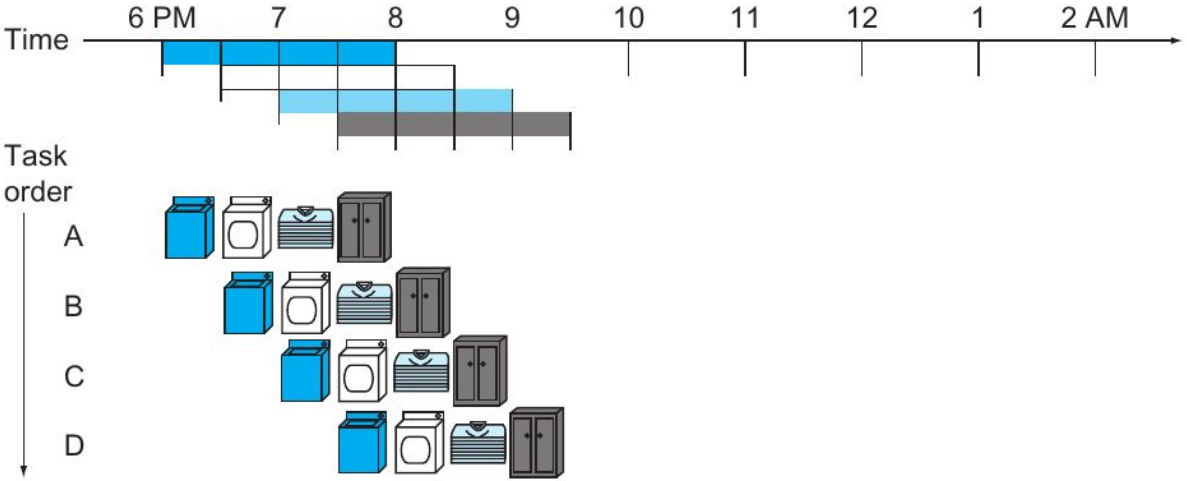


Com Pipeline



Onde está o ganho?

O tempo para executar uma tarefa completa muda?

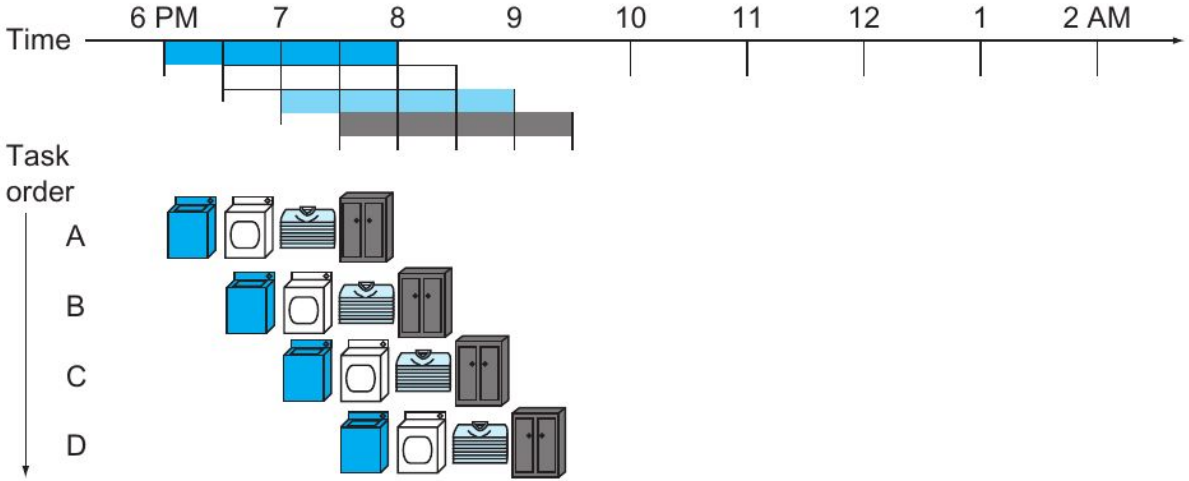


Onde está o ganho?

O tempo para executar uma tarefa completa muda?

Não. No exemplo, lavar uma trouxa de roupas ainda demora 2 horas.

Se o objetivo fosse lavar uma única trouxa de roupas, não haveria ganho algum.



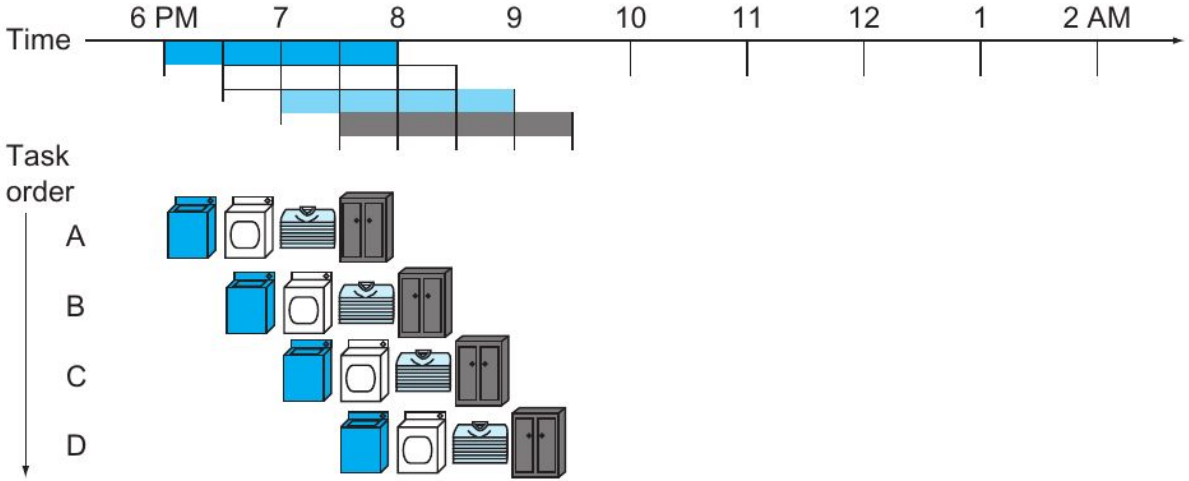
Onde está o ganho?

O tempo para executar uma tarefa completa muda?

Não. No exemplo, lavar uma trouxa de roupas ainda demora 2 horas.

Se o objetivo fosse lavar uma única trouxa de roupas, não haveria ganho algum.

Onde está o ganho de tempo?



Onde está o ganho?

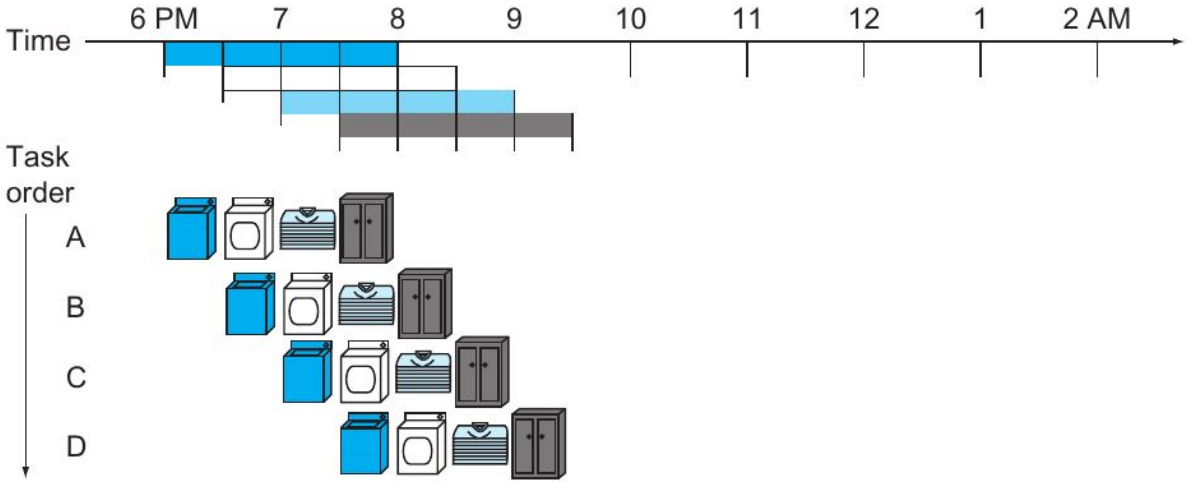
O tempo para executar uma tarefa completa muda?

Não. No exemplo, lavar uma trouxa de roupas ainda demora 2 horas.

Se o objetivo fosse lavar uma única trouxa de roupas, não haveria ganho algum.

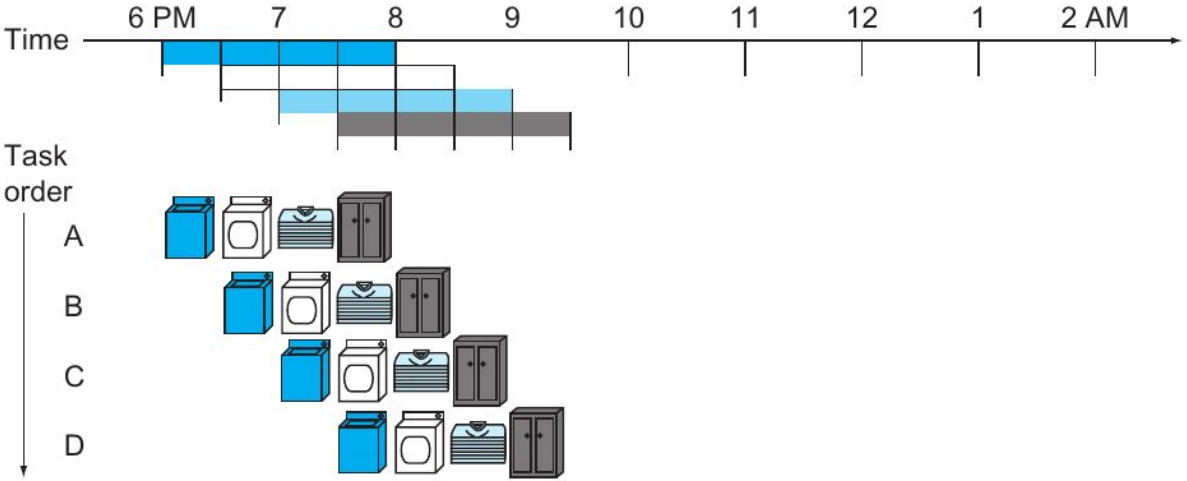
Onde está o ganho de tempo?

Temos múltiplas trouxas sendo "processadas" em paralelo, cada uma em um estágio.



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*)?

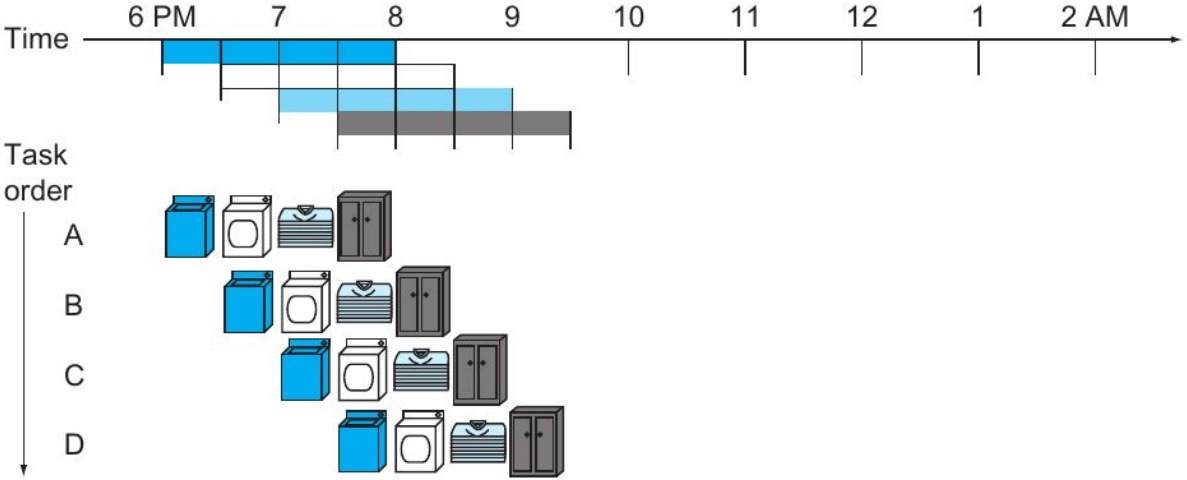


Onde está o ganho?

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.

Para isso o pipeline sempre deve estar cheio. **Isso acontece?**

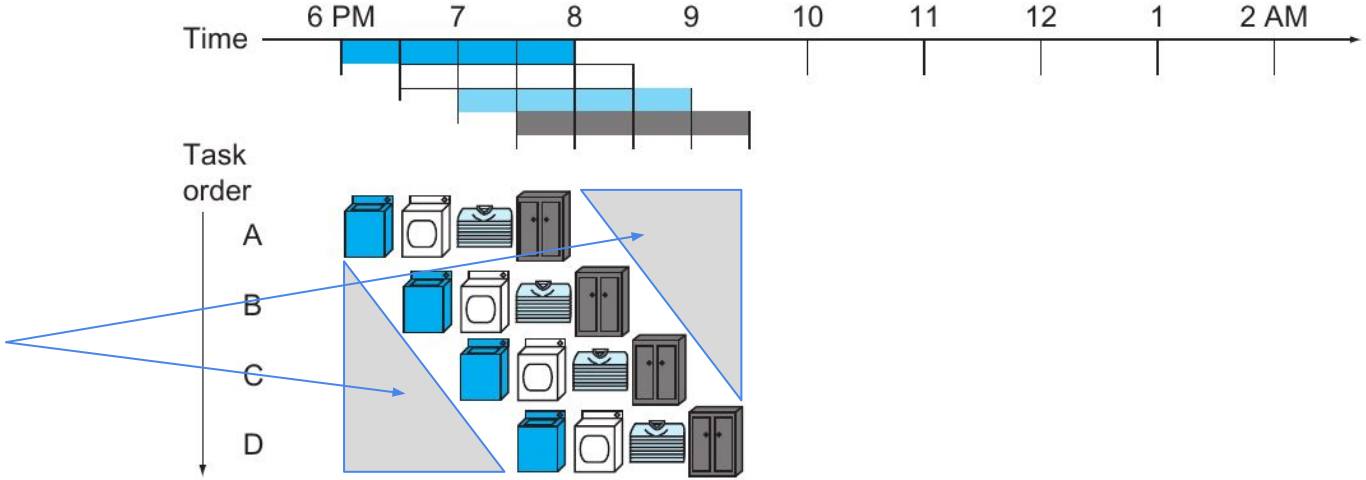


Onde está o ganho?

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.

Para isso o pipeline sempre deve estar cheio.



No início das tarefas, e no fim, temos unidades vazias.

Paradoxo (ou quase)

O tamanho da lavanderia permanece o mesmo.

Uma máquina de lavar, uma secadora, ...

Ao olhar para uma única tarefa, do início ao fim, o tempo para sua execução não muda.

O tempo de execução de uma única instrução não muda.

O **tempo gasto para uma instrução individual** é chamado de **latência**.

Ao olhar para um período de tempo suficientemente longo, **o número de tarefas completadas** é muito maior do que no modelo de ciclo único.

A **vazão (throughput)** pode aumentar em até *n* vezes.

Paradoxo (ou quase)

O tamanho da lavanderia permanece o mesmo.

Uma máquina de lavar, uma secadora, ...

Ao olhar para uma única tarefa, do início ao fim, o tempo para sua execução não muda.

O tempo de execução de uma única instrução não muda.

O **tempo gasto para uma instrução individual** é chamado de **latência**.

Ao olhar para um período de tempo suficientemente longo, **o número de tarefas completadas** é muito maior do que no modelo de ciclo único.

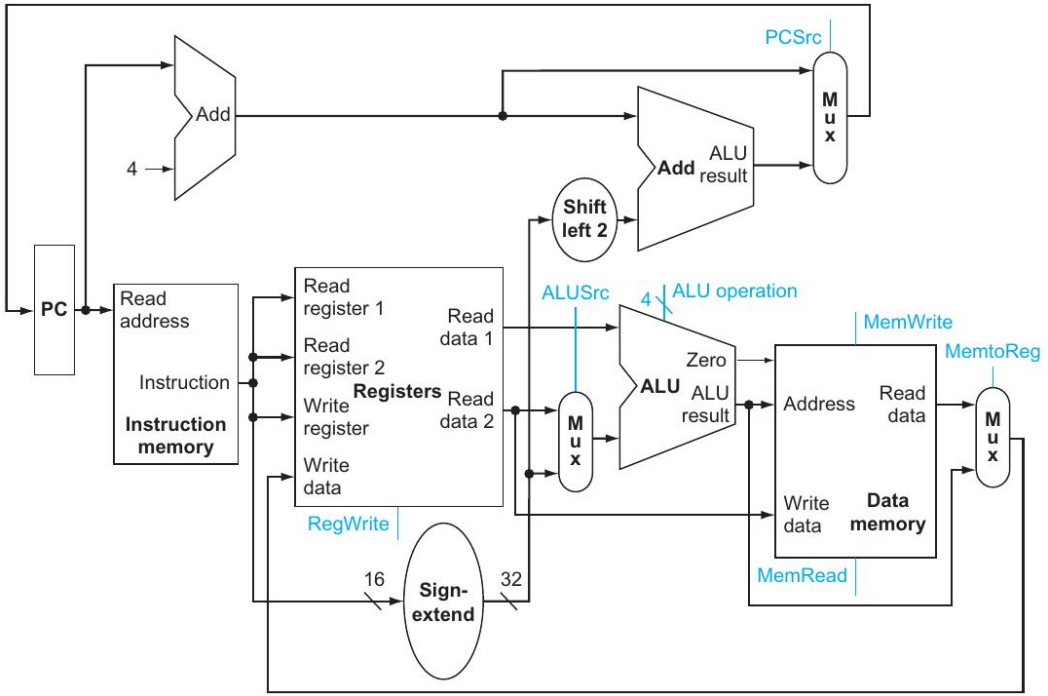
A **vazão (throughput)** pode aumentar em até *n* vezes.

Tente pronunciar corretamente.

Trocando maçãs por laranjas

Vamos aplicar a ideia da lavadora de roupas no processador MIPS32.

Quais são os estágios?



Trocando maçãs por laranjas

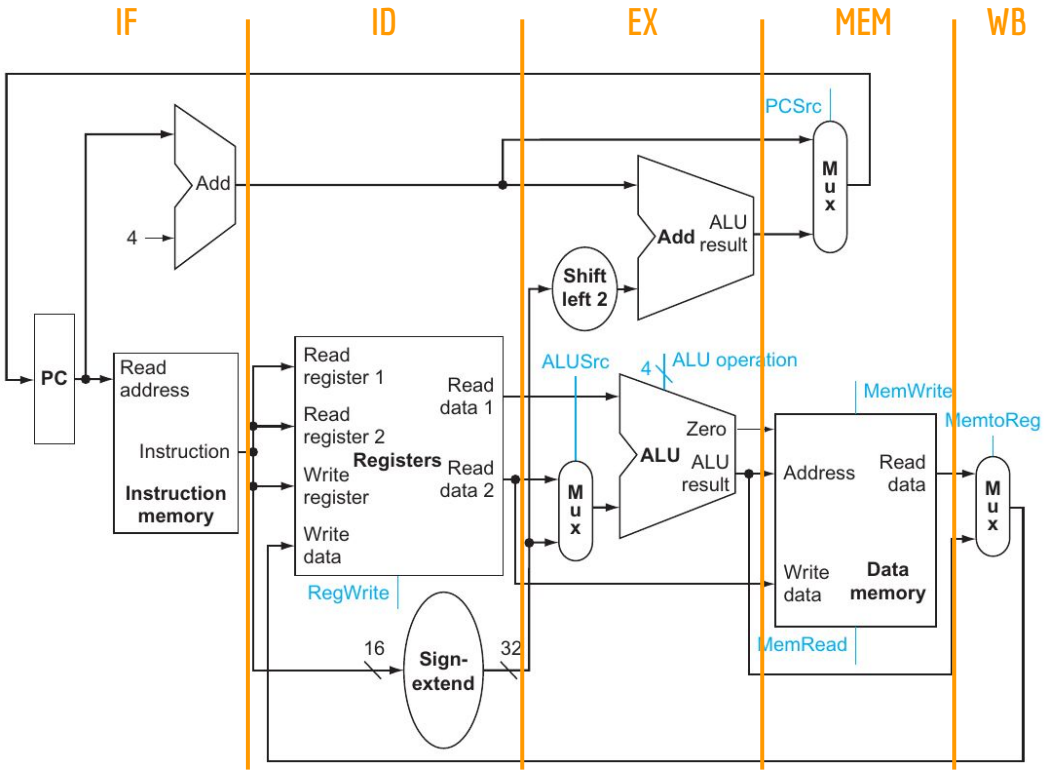
Buscar a instrução na memória.
Instruction fetch - IF.

Ler os registradores enquanto a unidade de controle decodifica.
Instruction decoding - ID.

Executar a operação/cálculo do endereço.
Execution - EX.

Acessar a memória de dados.
Memory Access - MEM.

Escrever o resultado num registrador.
Write Back - WB.



Estágios

O processador foi dividido em 5 estágios.

A **quantidade** de estágios
depende diretamente da arquitetura.

Microprocessador	Ano	Clock	Estágios Pipeline	Tamanho Despacho	Fora de ordem?	CPUs por Chip	Potência
486	1989	0,025 GHz	5	1	Não	1	5 W
Pentium	1993	0,066 GHz	5	2	Não	1	10 W
Pentium Pro	1997	0,2 GHz	10	3	Sim	1	29 W
Pentium 4 Willamette	2001	2 GHz	22	3	Sim	1	75 W
Pentium 4 Prescott	2004	3,6 GHz	31	3	Sim	1	103 W
Intel Core	2006	3 GHz	14	4	Sim	2	75 W
Core i7 Nehalem	2008	3,6 GHz	14	4	Sim	2-4	87 W
Core Westmere	2010	3,73 GHz	14	4	Sim	6	130 W
Core i7 Ivy Bridge	2012	3,4 GHz	14	4	Sim	6	130 W
Core Broadwell	2014	3,7 GHz	14	4	Sim	10	140 W
Core i9 Skylake	2016	3,1 GHz	14	4	Sim	14	165 W
Intel Ice Lake	2018	4,2 GHz	14	4	Sim	16	185 W

Estágios

O processador foi dividido em 5 estágios.

A **quantidade** de estágios **depende diretamente da arquitetura**.

Microprocessador	Ano	Clock	Estágios Pipeline	Tamanho Despacho	Fora de ordem?	CPUs por Chip	Potência
486	1989	0,025 GHz	5	1	Não	1	5 W
Pentium	1993	0,066 GHz	5	2	Não	1	10 W
Pentium Pro	1997	0,2 GHz	10	3	Sim	1	29 W
Pentium 4 Willamette	2001	2 GHz	22	3	Sim	1	75 W
Pentium 4 Prescott	2004	3,6 GHz	31	3	Sim	1	103 W
Intel Core	2006	3 GHz	14	4	Sim	2	75 W
Core i7 Nehalem	2008	3,6 GHz	14	4	Sim	2-4	87 W
Core Westmere	2010	3,73 GHz	14	4	Sim	6	130 W
Core i7 Ivy Bridge	2012	3,4 GHz	14	4	Sim	6	130 W
Core Broadwell	2014	3,7 GHz	14	4	Sim	10	140 W
Core i9 Skylake	2016	3,1 GHz	14	4	Sim	14	165 W
Intel Ice Lake	2018	4,2 GHz	14	4	Sim	16	185 W

Ajustando o período de clock

Veja mais uma vez o tempo gasto em cada unidade funcional. Cada unidade representa um estágio no pipeline.

Tempo de clock.

Sem pipeline, somamos tudo e ajustamos para o pior caso (800ps).

Dado que período de clock é fixo, qual o tempo de clock se adicionarmos o pipeline?

Instrução	Instruction Fetch	Register Read	Operação ALU	Data Access	Register Write	Total
LW	200ps	100ps	200ps	200ps	100ps	800ps
SW	200ps	100ps	200ps	200ps	100ps	700ps
Tipo -R	200ps	100ps	200ps			600ps
Branch	200ps	100ps	200ps			500ps

Ajustando o período de clock

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 estágios durassem o mesmo tempo.**

Instrução	Instruction Fetch	Register Read	Operação ALU	Data Access	Register Write	Total
LW	200ps	100ps	200ps	200ps	100ps	800ps
SW	200ps	100ps	200ps	200ps	100ps	700ps
Tipo -R	200ps	100ps	200ps			600ps
Branch	200ps	100ps	200ps			500ps

Ajustando o período de clock

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.

Qual a frequência?

Ajustando o período de clock

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.

Qual a frequência?

$F = 1/(200*10^{-12}) = 5*10^{9}$ Hz = 5GHz

Ganho

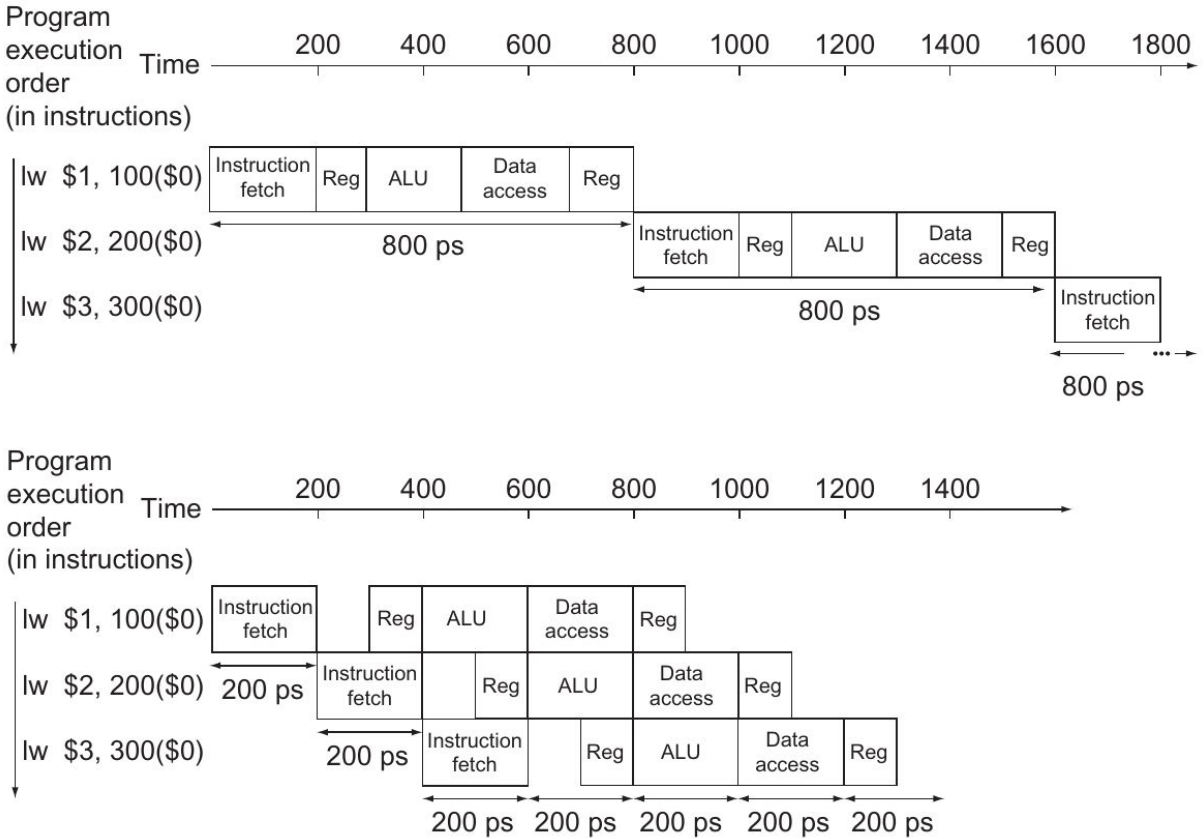
Na prática, nosso ganho de vazão vai ser aproximadamente

$$proporcao\ aumento\ vazao = \frac{Tempo\ de\ ciclo\ sem\ pipeline}{Tempo\ de\ ciclo\ da\ unidade\ mais\ lenta}$$

No processador MIPS32

$$proporcao\ aumento\ vazao = \frac{800}{200} = 4$$

Ciclo único versus Pipeline



Exercícios

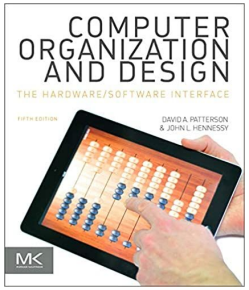
1. Considere que criamos um pipeline em um processador qualquer. Ao olhar para o tempo de execução de uma única instrução (latência) na versão com pipeline desse processador, esse tempo:
    a. Pode ser menor que o tempo do processador sem pipeline?
    b. Pode ser igual o tempo do processador sem pipeline?
    c. Pode ser maior que o tempo do processador sem pipeline?

    Confirme ou refute cada uma das afirmações, explicando detalhadamente.

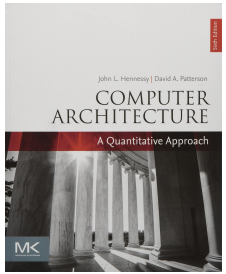
2. Toda instrução no MIPS possui o mesmo tamanho. Como isso nos ajuda no pipeline?
    a. Se as instruções fossem de tamanho variável, em qual estágio do pipeline precisaríamos estar para só então definir onde está a próxima instrução? Observação: seu processador x86-64 possui instruções de tamanho variável, e precisa tratar essa complexidade extra.

Referências

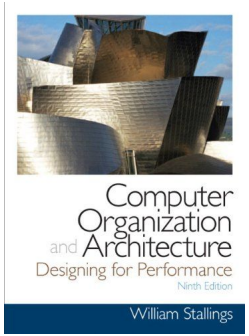
Patterson, Hennessy . Arquitetura e Organização de Computadores: A interface hardware/software. 2014.



Hennessy, Patterson. Arquitetura de Computadores: uma abordagem quantitativa. 2019.



Stallings, W. Organização de Arquitetura de Computadores. 10a Ed. 2016.



Licença