SSC0611
Arquitetura de Computadores

7ª Aula – Pipeline

Profa. Sarita Mazzini Bruschi

sarita@icmc.usp.br

Arquitetura CISC

- CISC – *Complex Instruction Set Computer*
  - Computadores complexos devido a:
    - Instruções complexas que demandam um número grande de ciclos para serem executadas
    - Dezenas de modos de endereçamento
    - Instruções de tamanhos variados
    - Referência a operandos na memória principal
  - Questionamentos quanto à necessidade de certas instruções
    - Levantamentos mostram as instruções mais utilizadas nos programas

Arquitetura CISC

- Estudos de Knuth, Wortman, Tanenbaum e Patterson em várias linguagens com relação a porcentagem de comandos

Comando	Fortran	C	Pascal
atribuicao :=	51%	38%	45%
if	10	43	29
call	5	12	15
loop	9	3	5
goto	9	3	0
outros	16	1	6

Arquitetura CISC

- Portanto:
  - Nas arquiteturas CISC fica mais difícil implementar o pipeline
  - A taxa média de execução das instruções por ciclo tende a ser bem menor do que 1 IPC (*instruction per cycle*)
  - A unidade de controle é microprogramada
  - Instrução complexa significa um maior tempo para decodificar e executar, muitas das quais são raramente usadas
- Surgiu então a arquitetura RISC

Arquitetura RISC

- RISC – *Reduced Instruction Set Computer*
- Características:
  - Instruções mais simples, demandando um número fixo de ciclos de máquinas para sua execução
  - Uso de poucos e simples modos de endereçamento
  - Poucos formatos das instruções
  - Apenas instruções de load/store referenciam operandos na memória principal
  - Cada fase de processamento da instrução tem a duração fixa igual a um ciclo de máquina

Arquitetura RISC

- Portanto:
  - Implementadas com o uso do pipeline
    - Formato fixo das instruções facilita o pipeline
  - As instruções são executadas na sua maioria em apenas um ciclo de máquina
  - A unidade de controle é em geral *hardwired*
    - Não há microprograma para interpretar as instruções
  - Arquitetura orientada a registrador
    - Todas as operações aritméticas são realizadas entre registradores
    - Define-se um grande conjunto de registradores

Arquitetura RISC

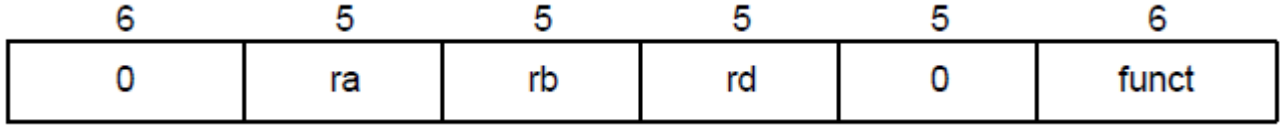
- Primeiros computadores RISC:
  - IBM 801 (1980)
    - É o antecessor do IBM PC/RT (RISC Tecnology)
  - Berkeley RISC I e RISC II (1980 e 1981)
    - Projetado por Patterson e Séquin
    - Inspirou o projeto do processador SPARC, da SUN Microsystem
  - Stanford MIPS (1981)
    - Projetado por Hennessy
    - Originou a MIPS Computer Systems

Arquitetura MIPS

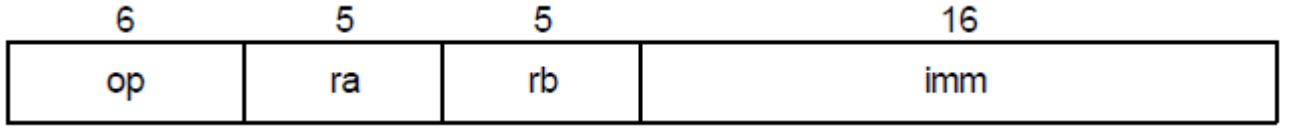
- MIPS: Microprocessor without Interlocked Pipeline Stages
- Conjunto de instruções do tipo RISC (*Reduced Instruction Set Computer*)
  - Usa poucos ciclos do processador para executar as instruções
  - Arquitetura do tipo load/store
- Arquitetura de 32 bits e atualmente de 64 bits
- Possui as seguintes características que simplificam a implementação
  - Todas as operações da ULA são realizadas somente sobre registradores
  - A memória é acessada somente através das instruções de *load* e *store*
  - As instruções possuem poucos formatos e são do mesmo tamanho

Arquitetura MIPS

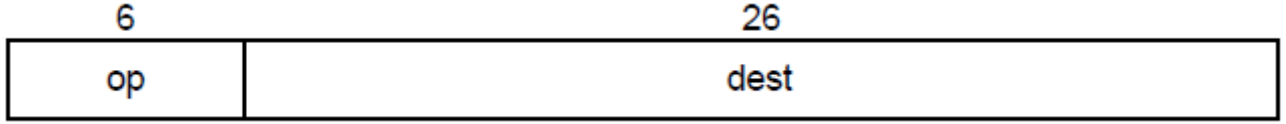
- 3 formatos de instruções:
  - Tipo R: operações registrador – registrador inteiro



  - Tipo I: operações de transferência de dados, desvios e instruções imediatas

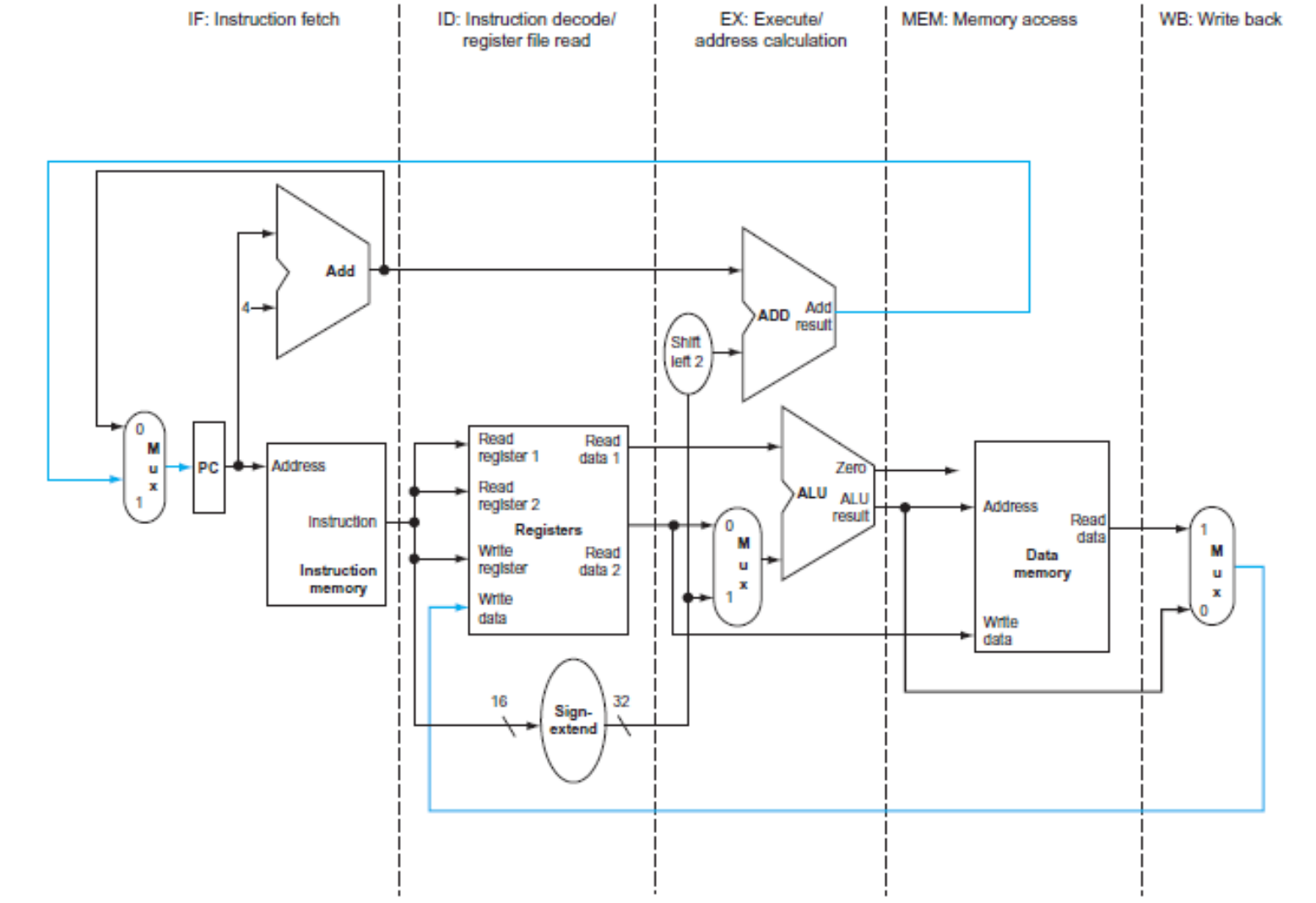


  - Tipo J: operações de saltos



Arquitetura MIPS

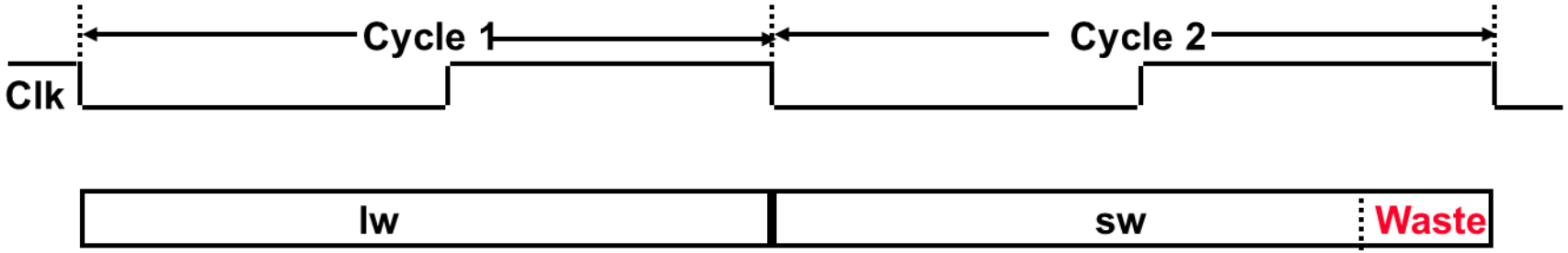
Execução em um único ciclo



1 ciclo de *clock*

Arquitetura MIPS

- Qual a desvantagem de se usar uma arquitetura com um único ciclo?
  - O uso do ciclo de *clock* se torna ineficiente pois este deve ser do tamanho da instrução mais lenta
  - Isso é um grande problema quando se considera instruções mais complexas como por exemplo multiplicação de ponto flutuante

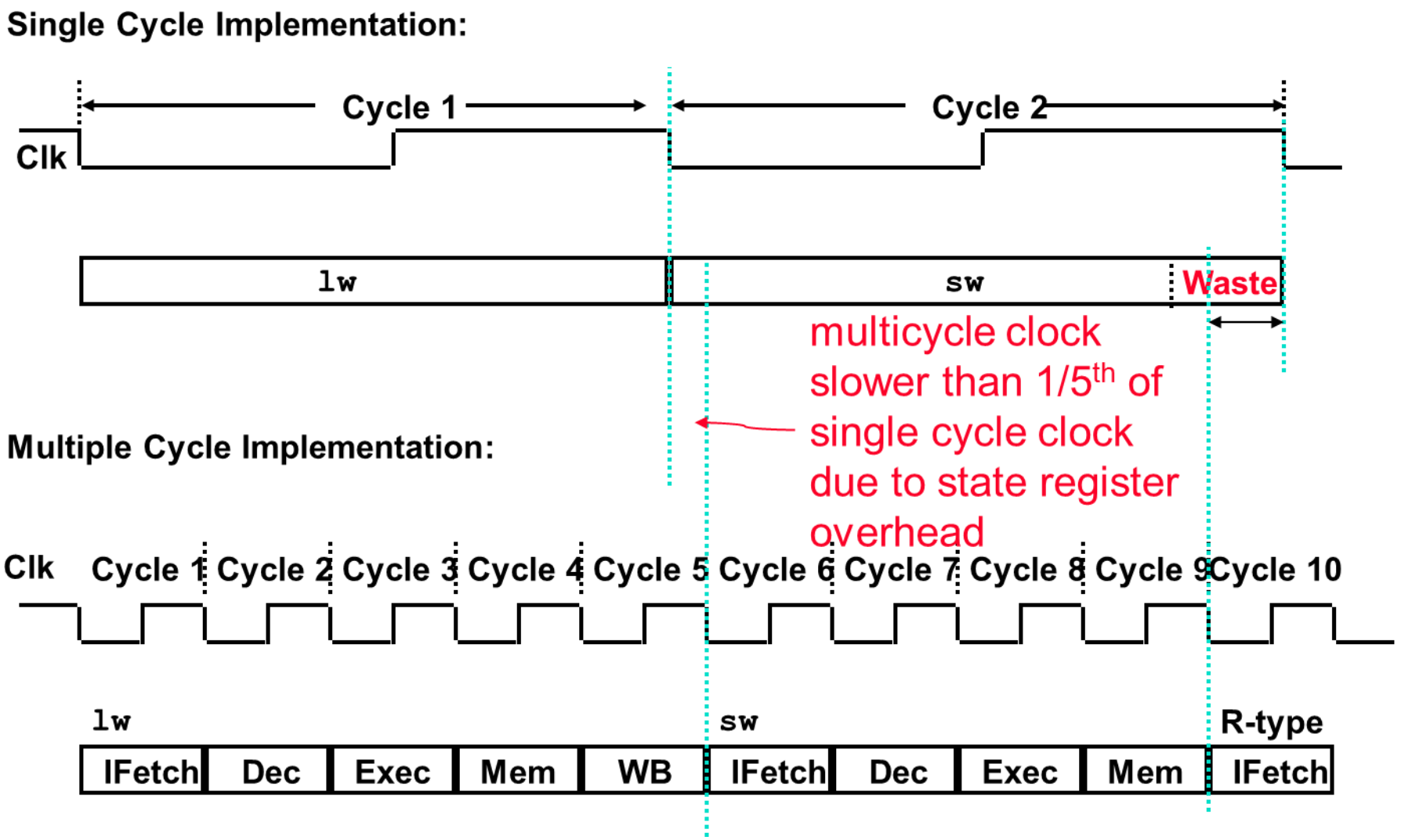


Arquitetura MIPS

- Solução:
  - Fazer com que as instruções demorem mais do que **um** ciclo de clock para finalizar
  - Quebrar as instruções em vários **passos** e cada passo deve consumir um ciclo
    - Nem todas as instruções demoram o **mesmo** número de ciclos de clock para finalizar

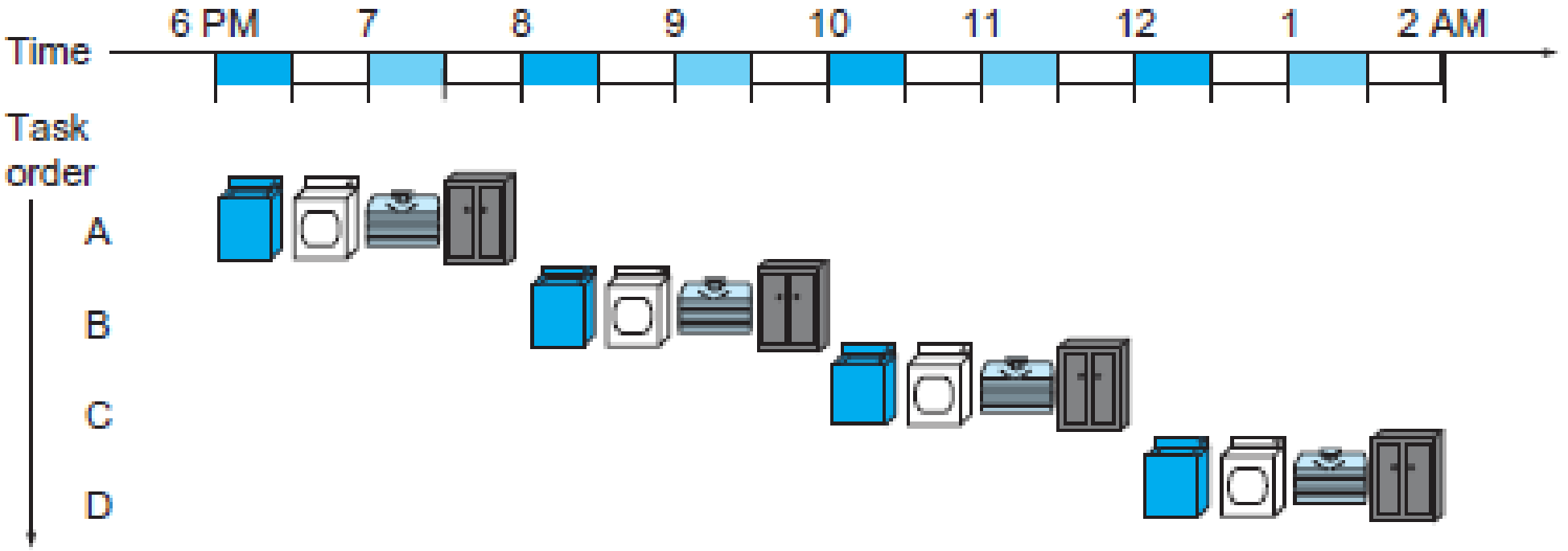
Arquitetura MIPS

Comparação: um único ciclo *x* multiciclo



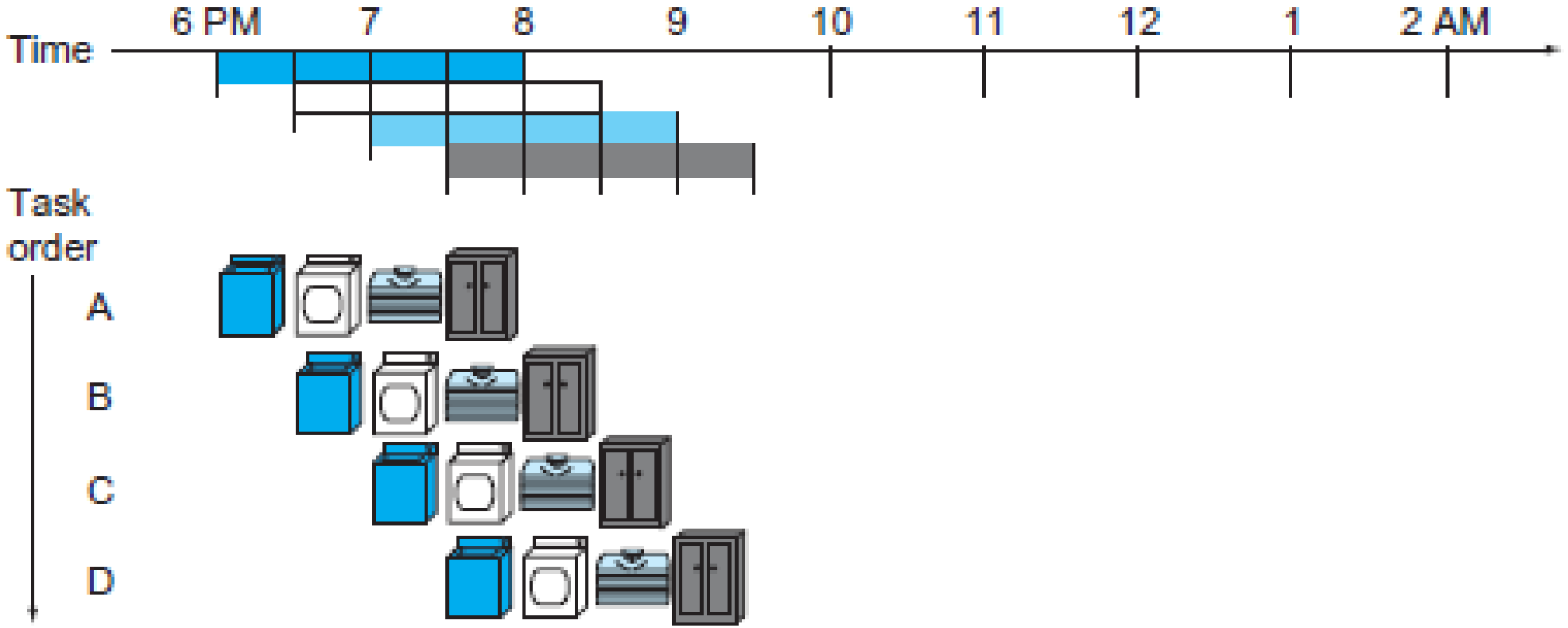
Pipeline

- Exemplo: Lavar roupa de maneira sequencial



Pipeline

- Exemplo: Lavar roupa com pipeline



Pipeline

- Algumas considerações:
  - O pipeline não diminui o tempo de execução de cada tarefa, mas aumenta o *throughput* de toda a carga de trabalho
  - A taxa do pipeline é limitado pelo estágio mais lento
  - Várias tarefas são realizadas simultaneamente utilizando diferentes recursos
  - Um *speedup* potencial é o número de estágios do pipeline
  - O tempo para “encher” o pipeline e para “esvaliá-lo” diminui o *speedup*
  - O pipeline trava se houver dependências
  - Exemplo com a execução de 100 instruções:
    - Computador com um único ciclo: 45 ns/ciclo * 1 CPI * 100 instruções = 4500 ns
    - Computador com multiciclo: 10 ns/ciclo * 4,2 CPI * 100 instruções = 4200 ns
    - Computador com 5 estágios ideais de pipeline: 10 ns/ciclo * (1 CPI * 100 instruções + 4 ciclos para esvaziar) = 1040 ns

CPI: *cycle per instruction*, *clocks per instruction*