Pipelining: conceitos básicos

O que é pipelining ?

- É uma técnica de implementação pela qual várias instruções são sobrepostas em execução, ela tira proveito do paralelismo que existe entre as ações necessárias para executar uma instrução.
- É a principal técnica de implementação usada para tornar CPUs rápidas.
- Uma pipeline é como uma linha de montagem, etapas que operam em paralelo. Ex. Linha de montagem de carros.

O que é Pipelining ?

- Cada etapa de uma pipeline é chamada de Estágio de pipe ou Segmento de pipe.
- Os estágios são conectados um ao outro para formar uma pipe;
  - As instruções entram numa ponta, seguem pelos estágios e saem na outra ponta.

Ciclo do Processador.

- A vazão (throughput) de uma linha de montagem é determinada pelo número de produtos montados em um determinado intervalo de tempo.
- A vazão de uma pipeline é dada pela frequência com que uma instrução é terminada, sai da pipe.
- Como os estágios são conectados, todos precisam estar prontos para executar ao mesmo tempo, logo todos têm o mesmo intervalo para executar suas tarefas.
- Este tempo é chamado Ciclo do Processador.

Ciclo do Processador.

- A duração do ciclo do processador deve ser o tempo necessário para o estágio da pipe mais lento ser executado.
- Em um computador, esse ciclo de processador normalmente é 1 ciclo de clock (às vezes 2).

Vantagem da Pipelining.

- Se os estágios da pipe estiverem perfeitamente balanceados o tempo por instrução no processador em pipeline, considerando condições ideais, será igual:

$$\frac{\text{Tempo por instrução em máquinas sem pipeline}}{\text{Número de Estágios da pipe}}$$

- Nessas condições, o ganho de velocidade da pipelining é igual ao número de estágios da pipe.

Desempenho da Pipelining.

- Normalmente os estágios da pipe não são perfeitamente balanceados, além do mais, pipelining envolve algum *overhead*. Logo o tempo por instrução no processador em pipeline não terá seu valor mínimo.
- A pipelining gera uma redução no tempo médio de execução por instruções. A redução pode ser vista no número de ciclos de clock por instrução (CPI).
- A pipelining é vista como redução de CPI.

Instruções RISC

- RISC – Reduced Instruction Set Computer.
- Propriedades de Arquiteturas RISC:
  - Todas as operações sobre dados se aplicam a dados nos registradores,e mudam o registrador inteiro(32 ou 64 bits)
  - Operações sobre memória (carregamento ou armazenamento) movem dados da memória para registradores ou de registradores para memória que podem manipular menos que um registrador inteiro(um byte, 16 bits ou 32 bits).
  - Os formatos de instrução são poucos e geralmente de mesmo tamanho.

Instruções RISC

- Essas propriedades simples ocasionam simplificações drásticas na implementação da pipelining.
- Assim como outras arquiteturas RISC, o conjunto de instruções MIPS oferece 32 registradores, embora o registrador 0 sempre tenha o valor 0. A maioria das arquiteturas RISC possui três tipos de instruções.

Instruções RISC

- Instruções da ALU:
  - Usam dois registradores ou um registrador e um imediato estendido por operador, operam sobre eles e armazenam o resultado em outro registrador.
    - Operações com imediato são chamadas de operações imediatas, possuem um offset de 16 bits.
  - A ALU executa operações como adição, subtração, ou lógico,...

Instruções RISC

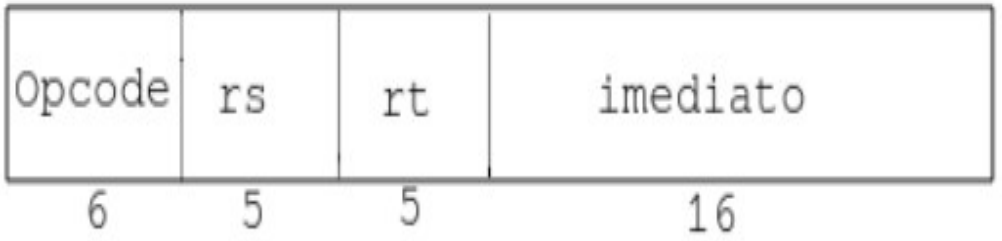
- Instruções de Carregamento e Armazenamento de Dados:
  - Usam um registrador, chamado de registrador Base, e um campo imediato (16 bits no MPIS) chamado de offset, como operandos. A soma, chamada de endereço efetivo, do conteúdo do registrador como o offset estendido, determina um endereço de memória.
    - No carregamento(load), um segundo registrador indica o destino do dado buscado na memória;
    - No armazenamento(store), o segundo registrador indica a origem do dado a ser armazenado.

Instruções RISC

- Instruções de Desvios e Saltos:
  - Desvios são transferências condicionais de controle, normalmente existem duas formas de especificar a condição:
    - Um conjunto de bits de condição (chamado código de condição); ou
    - Um conjunto limitado de comparações entre um par de registradores ou entre um registrador e o zero.
  - O destino de um desvio é obtido somando-se um offset ao PC atual.
  - Os saltos incondicionais são fornecidos em muitas arquiteturas RISC.

Instruções RISC-Formato MIPS 32

INSTRUÇAO TIPO I

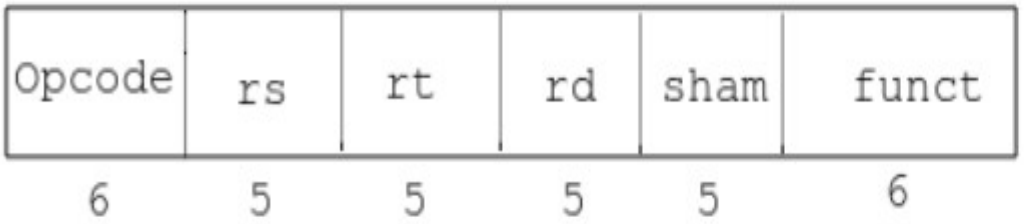


Usado para carregamento, armazenamento de bytes, meias-palavra, palavras, palavras duplas.

Instruçoes de desvio condicional.

Instruções RISC-Formato MIPS 32

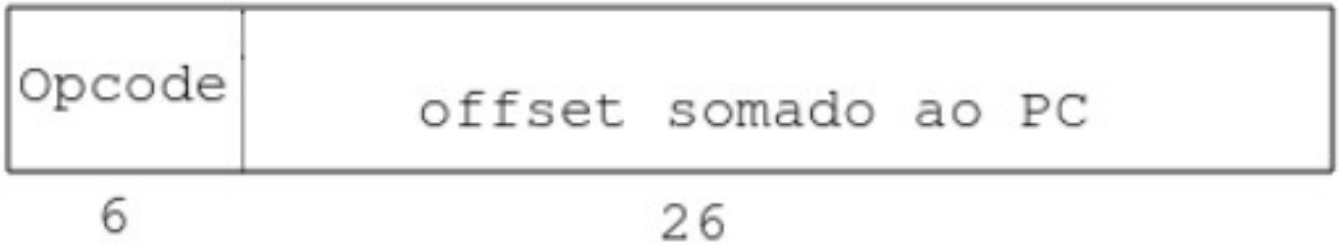
INSTRUÇAO TIPÓ R



Operandos de ALU registrador-regis
trador (rd <- rs funct rt)
funct codifica a operaçao.

Instruções RISC-Formato MIPS 32

INSTRUÇAO TIPO J



Saltos.

Pipeline Clássica de 5 estágios

- Para entender como um conjunto de instruções RISC pode ser implementado em um padrão de pipeline é preciso entender como ele é implementado fora da pipeline.
- Nesse exemplo focamos apenas um subconjunto de uma arquitetura RISC que consiste em carregamento-armazenamento de palavra, desvio e operações da ALU com inteiros.
- **Cada instrução** nesse subconjunto pode ser **implementada em no máximo 5 ciclos** de clock.

Pipeline Clássica de 5 estágios

- Nos 5 ciclos acontecem:

  Ciclo 1: Busca da instrução a ser executada: o PC, contador do programa, é enviado à memória para trazer a instrução a ser executada, depois atualiza o PC para indicar a próxima instrução(soma 4 ao valor em PC.

  Ciclo 2: Decodificação da instrução e busca de registradores: é preciso identificar o formato e os campos da instrução e depois ler os registradores, o banco de registradores; ou estender o valor do imediato caso necessário. O desvio pode ser completado ao final desse ciclo, armazenando o endereço de destino do desvio o PC, se a condição=V.

Pipeline Clássica de 5 estágios

Ciclo 3: Execução ou Cálculo de Endereço Efetivo: nesse ciclo a ALU faz uma operação sobre os operandos preparados no ciclo anterior, de acordo com o tipo da instrução:

- Memória: a ALU soma o registrador-base com o offset estedido para obter o endereço efetivo da memória;
- Registrador-Registrador: a ALU efetua a operação indicada pelo Opcode da instrução sobre os registradores;
- Registrador-Imediato:a ALU efetua a operação indicada pelo Opcode da instrução sobre o primeiro registrador e o imediato estendido

Pipeline Clássica de 5 estágios

Ciclo 4: Acesso à memória: se a instrução for um carregamento, a memória faz uma leitura no endereço efetivo calculado; se for um armazenamento, então a memória escreve os dados do segundo registrador no endereço efetivo.

Ciclo 5: Escrita de resultados: se a instrução for de carregamento, o dado lido da memória e escrito no segundo registrador; se for uma instrução da ALU registrador – registrador, o resultado é escrito no terceiro registrador.

- Desta forma, instruções de desvios usam 2 ciclos, as de armazenamento 4 ciclos e as demais 5 ciclos.(Não é a implementação ótima)

Pipeline Clássica de 5 estágios

- Estágios da Pipeline:
  - Baseando-se no número de ciclos necessários para a execução das instruções mais lentas e no que acontece em cada ciclo, a pipeline clássica, para instruções do subconjunto proposto, é formada por 5 estágios:

    1-Instruction Fetch; 2-Instruction Identify; 3-Execution; 4-Memory, 5-Write-back
  - Assim, podemos ter o máximo 5 instruções sendo executadas em um ciclo de clock, estando uma em cada estágio da pipeline.

Pipeline Clássica de 5 estágios

ciclo->	1	2	3	4	5	6	7	8	9
Instr. 1	IF	ID	EX	MEM	WB				
Instr. 2		IF	ID	EX	MEM	WB			
Instr. 3			IF	ID	EX	MEM	WB		
Instr. 4				IF	ID	EX	MEM	WB	
Instr. 5					IF	ID	EX	MEM	WB

Observe: No ciclo 1 apenas a instrução 1 ocupa a pipeline, no estágio 1-IF. No ciclo 5 temos as 5 instruções na pipe, onde a instrução 1 está no último estágio(WB) e a instrução 5 está no primeiro (IF).

Questões Básicas de Desempenho

- Observe que uma instrução executada fora da pipeline precisava de 2 a 5 ciclos de clock para ser encerrada (slide 17) e que na pipeline a cada ciclo, após o ciclo 5, temos uma instrução encerrada.
- O pipelining aumenta o throughput (vazão) de instruções de CPU – o número de instruções completadas por unidade de tempo – mas não reduz o tempo de execução de uma instrução individual.

Questões Básicas de Desempenho

- O aumento do throughput da instrução significa que um programa roda mais rápido e possui tempo de execução menor, embora nenhuma instrução isolada seja mais rápida.
- A verdade, o pipeline aumenta o tempo de execução de cada instrução devido ao overhead causado pelo controle da execução.
- Throughput é o trabalho total feito em um intervalo de tempo;
- Latência é o tempo de início e fim de um evento.

Exercício.

1) Considere o processador em não-pipeline anterior. Suponha que ele tenha um ciclo de clock de 1 ns e que use 4 ciclos para operações da ALU e desvios, 5 ciclos para operações com memória. Suponha que as frequências relativas dessas operações sejam 40%, 20% e 40%, respectivamente. Suponha que, devido a problemas de desvio de clock e preparação, o pipelining do processador acrescente 0,2 ns de overhead ao clock. Ignorado qualquer impacto na latência, quanto ganho de velocidade na taxa de execução da instrução teremos com uma pipeline?

Solução

- sem Pipeline:

  Tempo Médio de execução = Ciclo x CPI médio

  - Tm = 1 ns x ((40% + 20%) x 4 + 40% x 5)
  - Tm = 1 x (0,6 X 4 + 0,4 x 5)
  - Tm = 1 x (2,4 + 2,0)
  - Tm = 1 x 4,4 = 4,4 ns

Solução

- com Pipeline
  - Ciclo = 1 ns + 0,2 ns = 1,2 ns
- Tempo médio de execução de uma instrução em pipeline é 1 Ciclo.
- Ganho de velocidade do pipeline = $\frac{\text{Tm sem pipeline}}{\text{Tm com pipeline}}$
- Assim temos

  (4,4 ns / 1,2 ns) = 3,7 vezes.

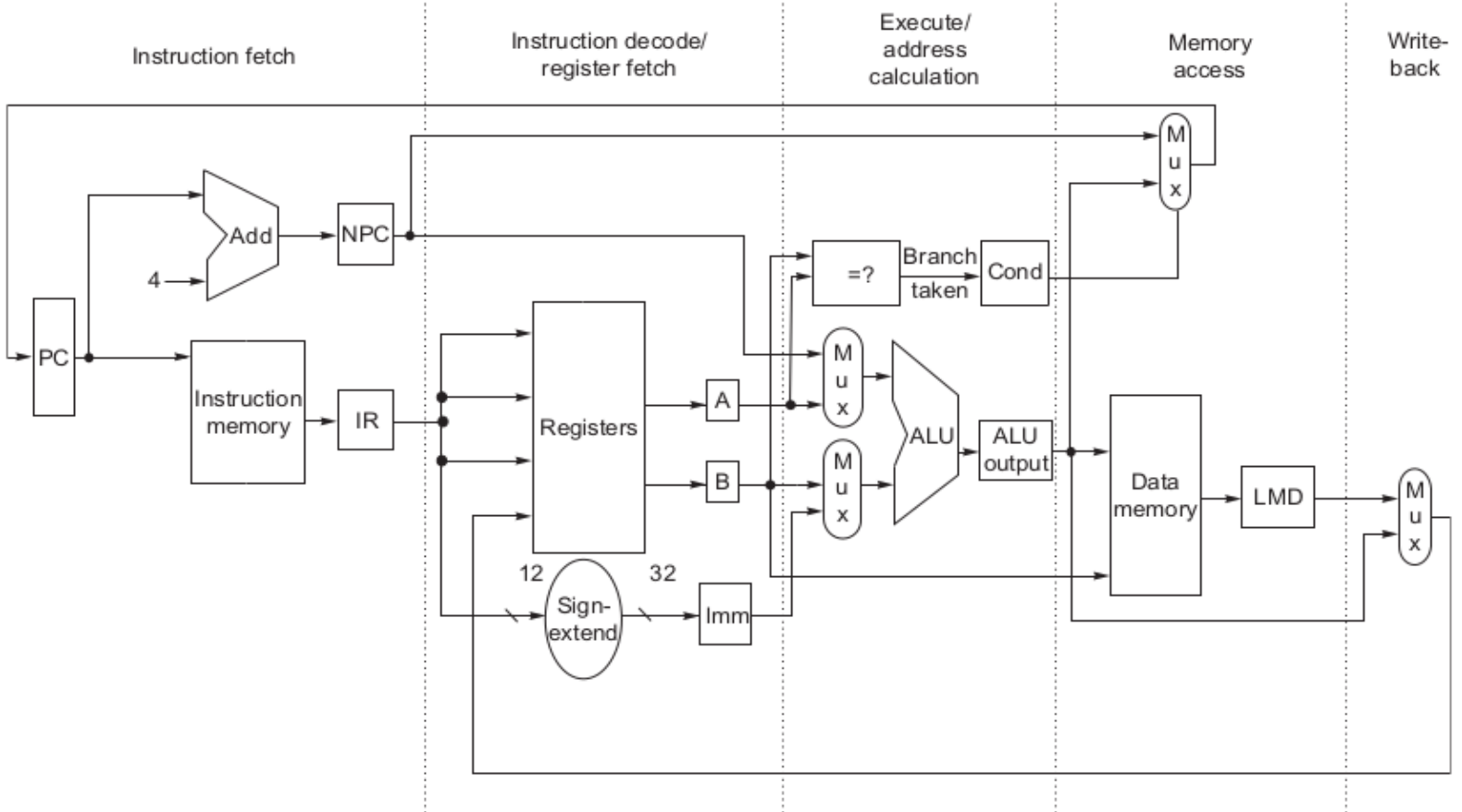
Perigos da Pipeline – (Hazards)

- Existem situações, chamadas de perigos (hazards) que impedem que a próxima instrução no fluxo de instruções seja executada durante seu ciclo de clock.
- Os hazards reduzem o desempenho do ganho de velocidade ideal obtido pelo pipelining.
- Existem 3 tipos de hazards:
  - Estruturais: conflitos de recursos de hardware.
  - de Dados: ocorre quando uma instrução depende de outra.
  - de Controle: ocorre quando o PC é modificado por instruções.

Perigos da Pipeline – (Hazards)

- Para evitar um perigo normalmente exige que algumas instruções na pipeline tenham permissão para continuar a execução enquanto outras são adiadas fazendo com que as instruções seguintes na pipeline sejam adiadas também.
- Adiar uma instrução consiste em inserir um *stall* na pipeline.

Uma implementação simples do MIPS



Uma implementação simples do MIPS

- Esse exemplo traz uma implementação da via de dados para um subconjunto do MIPS, que consiste em :
  - carregamento e armazenamento de palavra;
  - branch-equal-zero;
  - e operações da ALU com inteiros.

- Mais adiante serão incorporadas as operações básicas de ponto-flutuante.

Uma implementação simples do MIPS

- Cada instrução do MIPS pode ser implementada no máximo em 5 ciclos de clock. Os 5 ciclos de clock são os seguintes:

  1-Ciclo de busca de instrução (IF):

  IR ← Mem[PC];

  NPC ← PC + 4;

  - Operação: enviar o PC e buscar a instrução da memória para o registrador de instrução(IR), incrementar o PC em 4(bytes) para endereçar a próxima instrução, em NPC. IR é usado nos ciclos subsequentes.

Uma implementação simples do MIPS

2- Decodificador de instrução/busca de registradores (ID).

A ← Regs[rs];

B ← Regs[rt];

Imm ← campo imediato estendido por sinal do IR.

- Operação: Decodificar a instrução e acessar o banco de registradores para ler os registradores(rs e rt).
  - As saídas dos registradores de uso geral são lidas para dois registradores temporários (A e B) para uso em outros ciclos.
  - Os 16 bits inferiores do IR também são estendidos por sinal e armazenado no registrado temporário Imm. A decodificação é feita em paralelo com a leitura dos registradores

Uma implementação simples do MIPS

3- Execução/cálculo endereço efetivo de memória (EX).

A ALU opera sobre os operandos no ciclo anterior, realizando uma das quatro funções dependendo do tipo de instrução MIPS:

- referência memória: faz a soma para definir o endereço efetivo

  ALUOutput ← A + Imm;
- Registrador-Registrador: faz a operação definida pelo código da função nos registradores

  ALUOutput ← A func B;
- Registrador-Imediato: faz a operação definida pelo opcode da instrução

  ALUOutput ← A op Imm;
- Desvio: soma o NPC com o valor do imediato estendido

  ALUOutput ← NPC + (Imm <<2);

  Cond ← (A==0) -

Implementando um Caminho de Dados - Datapath

- Precisamos examinar os principais componentes necessários para executar cada classe de instruções MIPS, e os sinais de entrada e saída desses componentes.
- Para armazenar e acessar as instruções, bem como atualizar o ponteiro para a próxima instrução, são necessários:
  - Memória de Instruções;
  - PC – program counter – ponteiro que indica a instrução a ser executada; e
  - Somador : cuja a função é atualizar o PC para a próxima instrução a ser executa(+4 bytes).

Implementando um Caminho de Dados - Datapath

- Para implementar as operações para a ALU são necessários:
  - Um Banco de Registradores; e
  - Uma Unidade de Lógica e Aritmética.

Implementando um Caminho de Dados - Datapath

- Para implementar loads e stores, além do Banco de Registradores e a ALU, apresentados anteriormente, são necessários:
  - Unidade de memória de dados; e
  - Uma Unidade de Extensão de Sinal.

Implementando um Caminho de Dados - Datapath

- Para implementar as operações de desvios são necessários:
  - Uma ALU para avaliar a condição do desvio;
  - e um somador para calcular o novo valor do PC.