Pipeline

Ciclos de Operação da CPU
Estágios do Pipeline
Previsão de Desvio

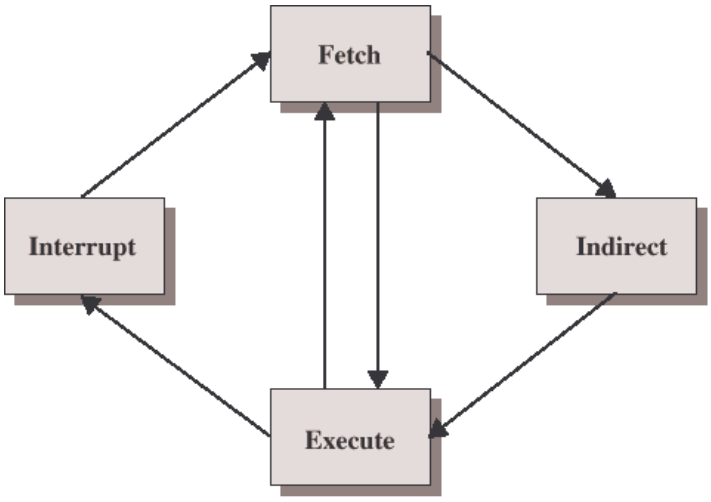
Estrutura da CPU

- Em cada ciclo, a CPU deve:
  - Buscar instruções
  - Interpretar instruções
  - Buscar dados
  - Processar dados
  - Escrever dados na memória

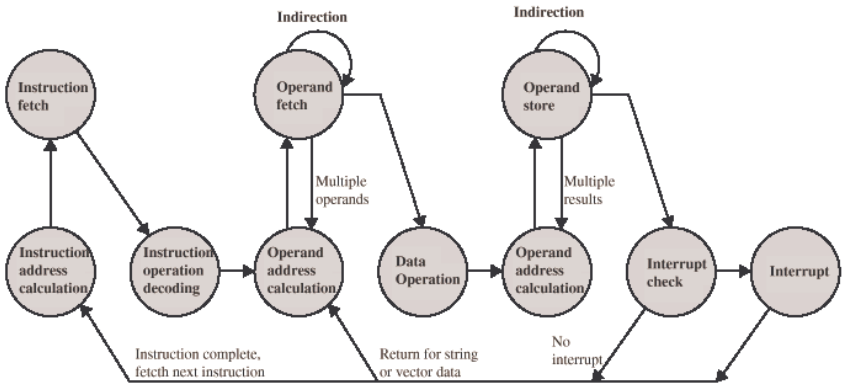
Ciclos de Operação da CPU

- Pode necessitar acesso à memória para buscar Operandos
  - Acesso Indireto
- Pode necessitar um sub-ciclo adicional
- Controle de Interrupção

Ciclos de Operação com Endereçamento Indireto



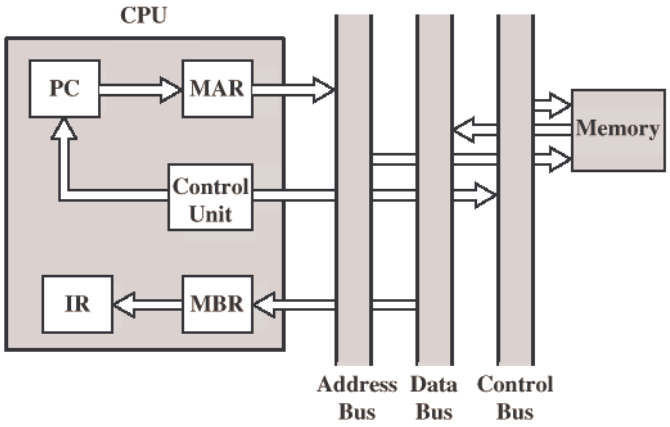
Ciclos de Operação



Data Flow
Ciclo de Busca (Instruction Fetch)

- Depende do projeto da CPU
- Em Geral: Ciclo de Busca (Fetch)
  - PC (Program Counter) contém endereço da próxima instrução
  - Endereço é movido para MAR (Memory Address Reg.)
  - Endereço é posto no Address Bus
  - Unidade de Controle sinaliza *Leitura* para a Memória
  - Resultado é posto no Data Bus
    - Copiado para MBR (Memory Buffer Reg.)
    - Movido para o IR (Instruction Reg.)
  - PC ++

Data Flow - Ciclo de Busca



MBR = Memory buffer register
MAR = Memory address register
IR = Instruction register
PC = Program counter

Data Flow - Decodificação (Data Fetch)

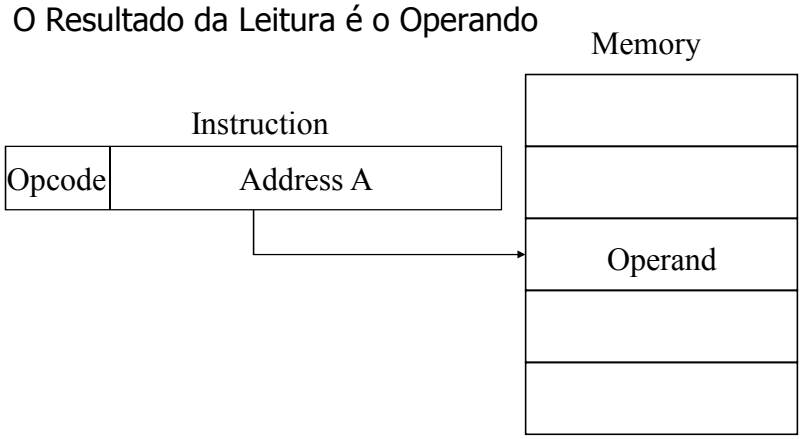
- IR é examinado
- Se Endereçamento Imediato:
  - N bits mais a direita do MBR é o Operando

Instruction

Opcode	Operand

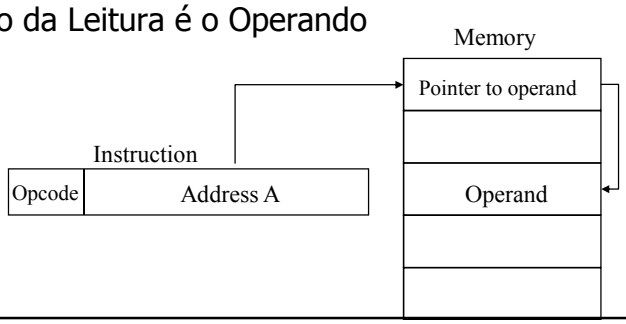
Data Flow - Decodificação (Data Fetch)

- Se Endereçamento Direto:
  - N bits mais a direita do MBR transferidos para o MAR
  - Unidade de Controle sinaliza *Leitura* para a Memória
  - O Resultado da Leitura é o Operando

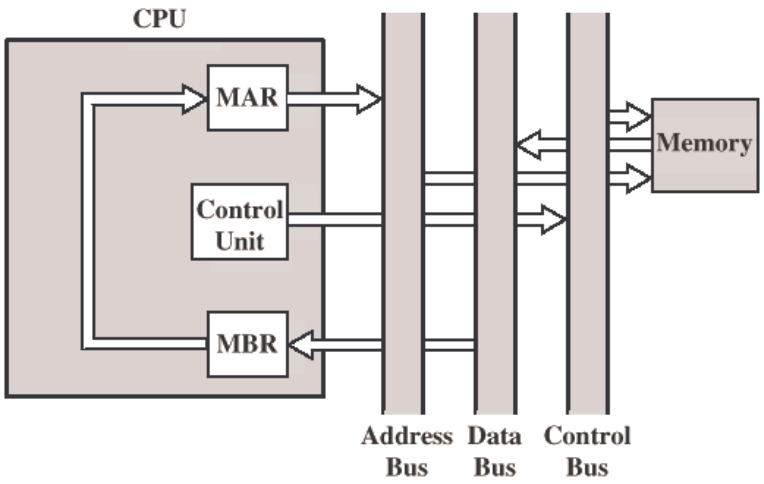


Data Flow - Decodificação (Data Fetch)

- Se Endereçamento Indireto:
  - N bits mais a direita do MBR transferidos para o MAR
  - Unidade de Controle sinaliza *Leitura* para a Memória
  - Resultado (endereço do operando) movido para MBR
  - MBR transferido para o MAR
  - Unidade de Controle sinaliza *Leitura* para a Memória
  - O Resultado da Leitura é o Operando



Data Flow - Decodificação: Indireto



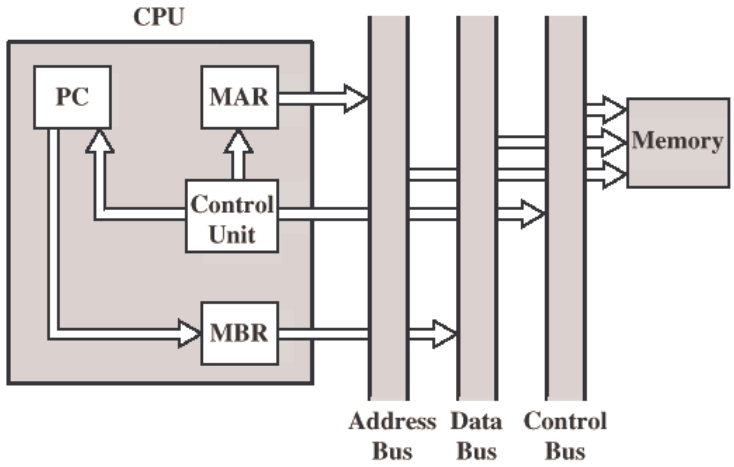
Data Flow - Execução (Execute)

- Varia muito conforme a CPU
- Depende da Instrução sendo executada
- Pode Incluir:
  - Memory read/write
  - Input/Output
  - Transferência entre Registradores
  - Operações da ULA
- Basicamente:
  - Realiza Operação Lógica/Aritmética sobre Operandos
- Pode conter ciclo de Escrita do Resultado na Memória

Data Flow - Interrupção (Interrupt)

- Simples
- PC atual é salvo para permitir *Restore*
  - Conteúdo do PC copiado para MBR
  - Stack pointer (SP) loaded to MAR / SP ++
  - MBR written to memory
- PC carregado com o endereço de atendimento da respectiva interrupção
- A primeira instrução da rotina de tratamento de interrupção é Buscada … (Toda a Rotina)
- RESTORE:
  - SP -- / Stack pointer (SP) loaded to MAR
  - MBR é copiado para PC

Data Flow - Interrupção



CPU- Otimização Busca/Execução

Explorar o
Paralelismo Implícito
na
Execução das Instruções

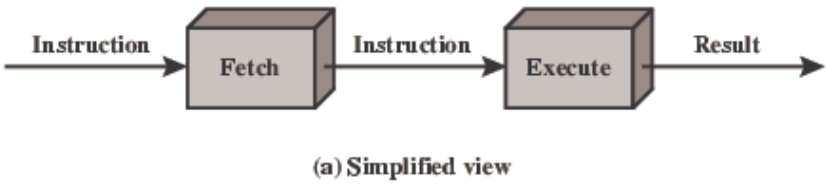
Prefetch

- Ciclo de Busca (Fetch) sempre acessa a Memória
- Ciclo de Execução (Execution) normalmente não acessa a Memória
- Pode-se buscar a próxima instrução durante a execução da Instrução Corrente
- Denominado: *Instruction Prefetch*

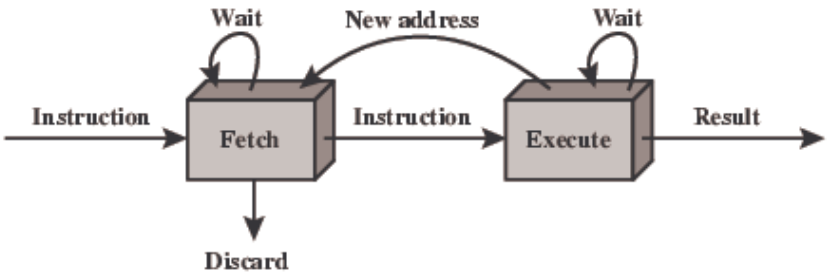
Prefetch - Melhora de Performance

- Mas não Dobra:
  - Fetch normalmente bem mais curto que Execution
    - Prefetch mais de uma instrução ? ? ? ? ?
  - Qualquer JUMP ou BRANCH significa que a instrução "*prefechessada*" não é mais necessária
- Solução:
  - Adicionar mais estágios!!!

Prefetch – Dois Estágios (FIFO)



(a) Simplified view



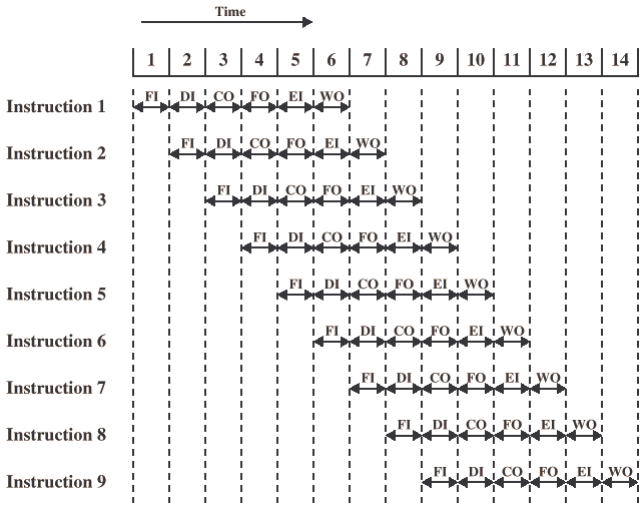
(b) Expanded view

Pipelining

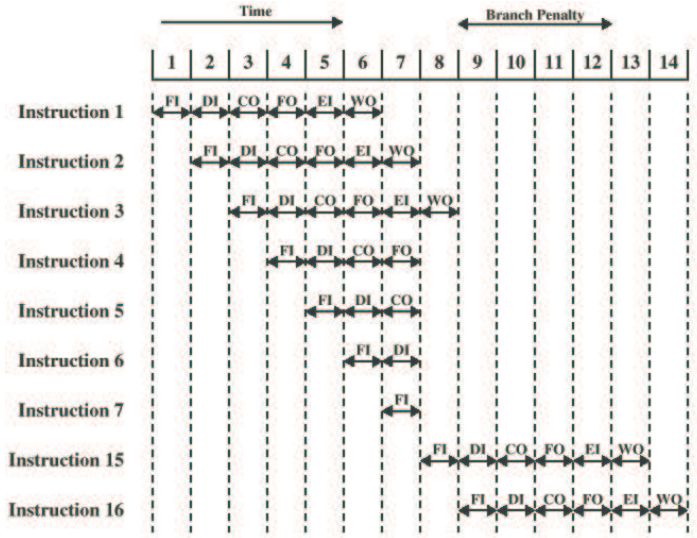
- Fetch instruction - FI
- Decode instruction - DI
- Calculate operands - CO
- Fetch operands - FO
- Execute instructions - EI
- Write result - WO

- Sobrepor essas Operações ! ! ! ! ! !

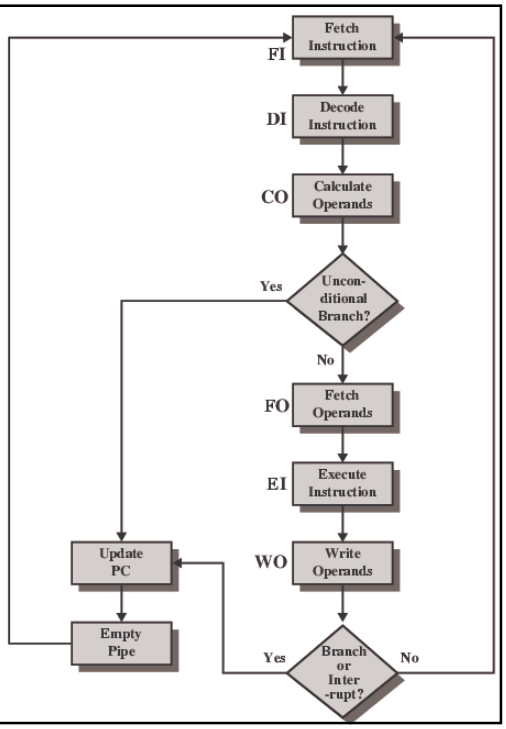
Timing do Pipeline



Desvios (Branch) no Pipeline

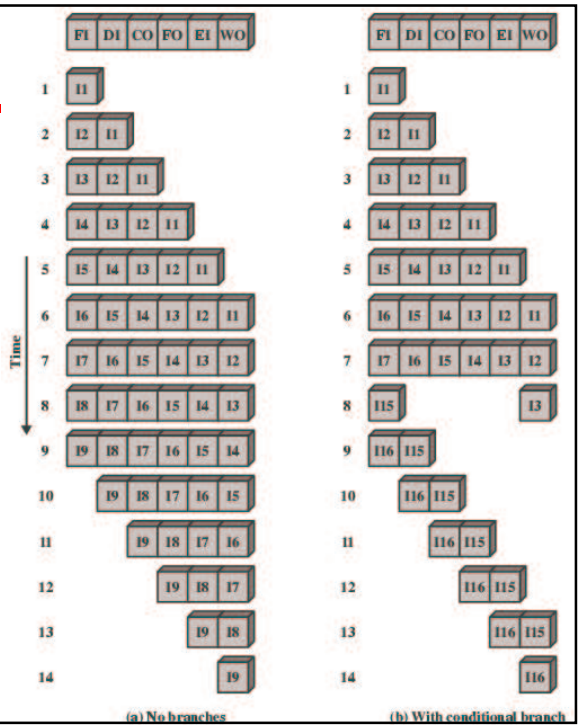


Pipeline de Instrução de 6 Estágios

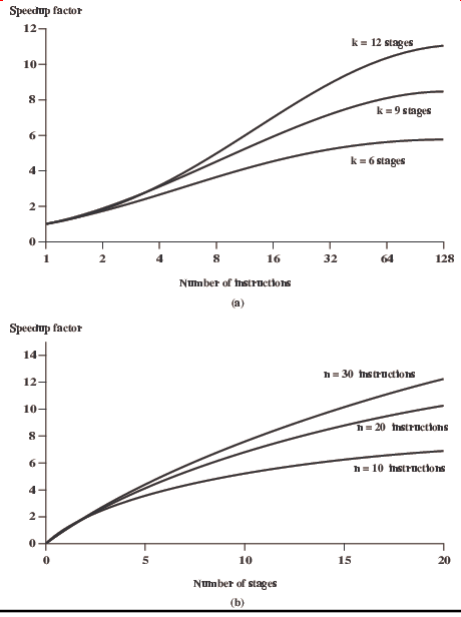


Pipeline de Instrução de 6 Estágios

- I3 é o Branch



Speedup Factors with Instruction Pipelining

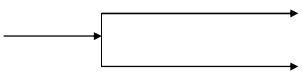


Tratando os Desvios (Branches)

- Multiple Streams
- Prefetch Branch Target
- Loop buffer
- Branch prediction
- Delayed branching

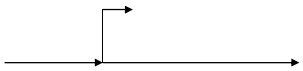
Multiple Streams

- Tem Dois Pipelines ! ! !
  - Essa era Óbvia!!
- Prefetch cada BRANCH em um pipeline separado
- Usa apropriadamente o Pipeline
- Leva à contenção de Barramento e Registradores (bus & register contention)
  - Ex.: Quando os dois lados querem modificar o mesmo Registrador
- Multiplos BRANCHES necessitam de mais Pipelines!!



Prefetch Branch Target

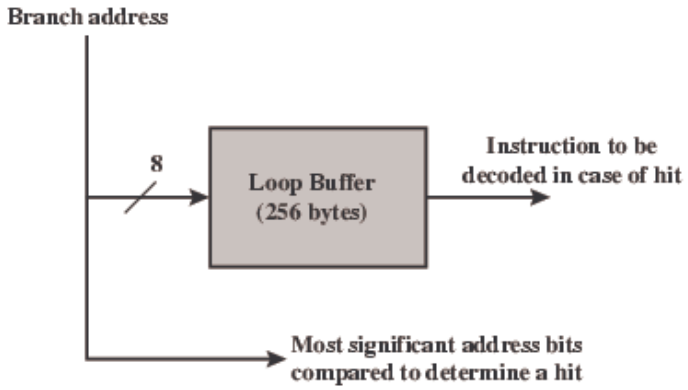
- Instrução *Target* do BRANCH é "prefetchessada" juntamente com instruções normais
- Mantém target até que o BRANCH seja executado
- Utilizado no IBM 360/91



Loop Buffer

- Memória muito rápida
- Mantida pelo estágio de Busca (Fetch) do pipeline
- Checa o buffer antes de Buscar da memória
- Muito eficiente para pequenos loops e jumps
- É como uma cache de instruções (menor)
- Utilizado no CRAY-1
- Atende muito bem Loops com IF – THEN – ELSE
- Motorola 68010 (tamanho 3 instruções):
  - Decrementa
  - Compara
  - Branch

Loop Buffer Diagram



Branch Prediction (1)

- Predict never taken
  - Assume que o BRANCH nunca vai ocorrer
  - Sempre busca a próxima instrução
  - 68020 & VAX 11/780
- Predict always taken
  - Assume que o BRANCH vai ocorrer sempre
  - Sempre busca a instrução alvo

Branch Prediction (2)

- Predict by Opcode
  - Algumas instruções tem mais chances de resultar em BRANCH que outras
  - Alcança até 75% de sucesso
- Taken/Not taken switch
  - Baseado na história anterior
  - Associa um ou mais bits a cada tipo de BRANCH em tabela para guardar se deu ou não salto na última vez
  - Eficiente para loops
    - For I = 1 to 10 (compara se I = 10 e jump se não for)
    - Acerta 10 vezes e erra uma (na saída)
  - Com um bit, um erro vai ocorrer ao entrar no loop e ao sair!

Branch Prediction (3)

- Delayed Branch
  - Não tomar o desvio até que seja necessário (mais tarde)
  - Rearranjar as instruções de um programa
  - Vamos ver em RISC

Branch Prediction State Diagram