INE5607 – Organização e Arquitetura de Computadores

Unidade Central de Processamento

Aula 17: Processadores com despacho múltiplo de instruções

Prof. Laércio Lima Pilla

laercio.pilla@ufsc.br

Sumário

- Despacho múltiplo de instruções
- Despacho múltiplo estático
- Despacho múltiplo dinâmico
- Considerações finais

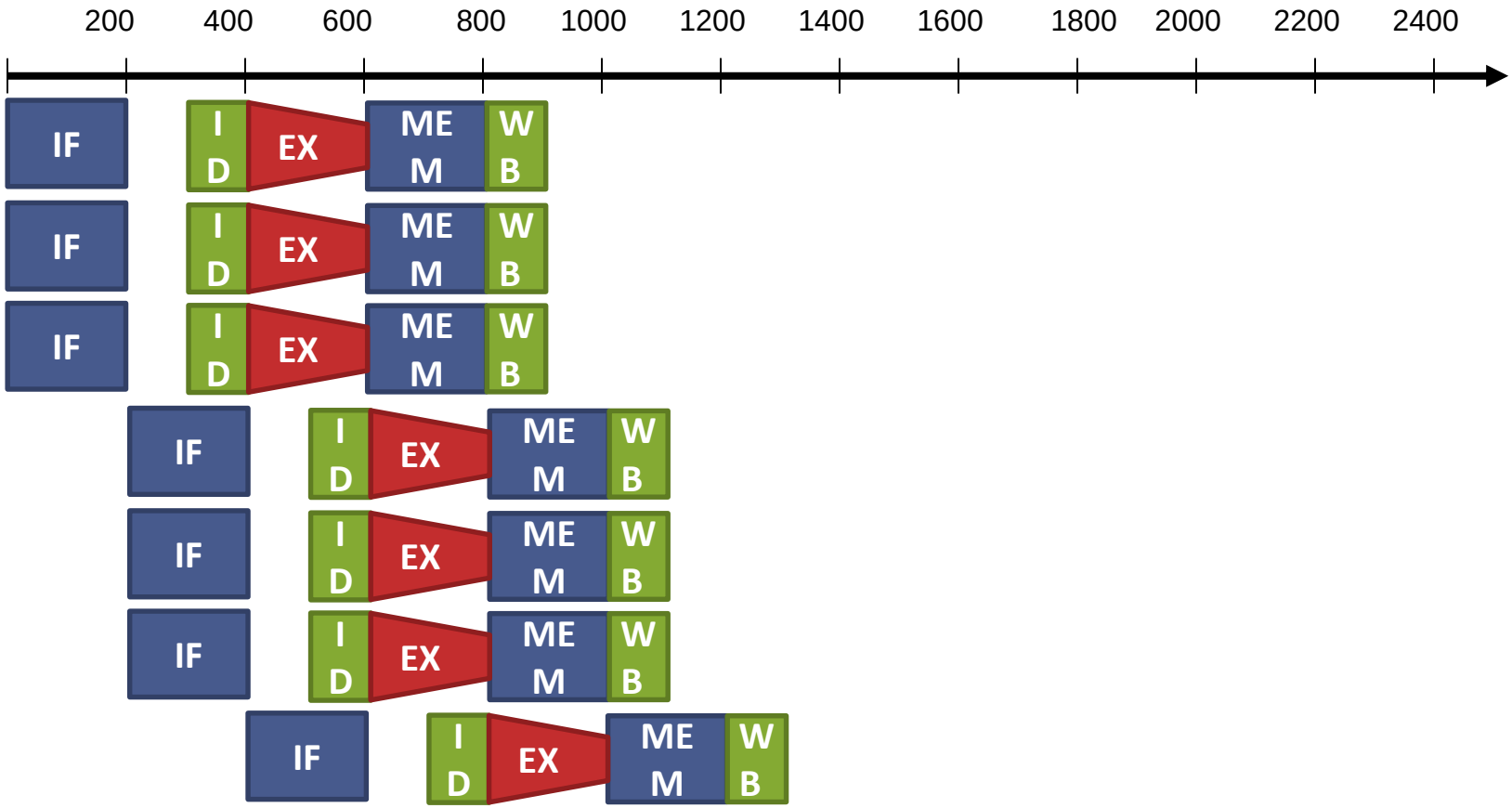
DESPACHO MÚLTIPLO DE INSTRUÇÕES

Despacho múltiplo de instruções

- **Como aumentar o desempenho além do pipeline?**
  - **Mais paralelismo em nível de instrução!**
    - Aumentar a profundidade do pipeline para ter mais instruções em execução
    - Lançar múltiplas instruções para a execução em paralelo

Despacho múltiplo de instruções

- Despacho múltiplo de instruções
  - Múltiplas instruções disparadas por ciclo



Despacho múltiplo de instruções

- Exemplo de despacho múltiplo no MIPS

Instruções\Ciclos	1	2	3	4	5	6	7
ULA ou branch	IF	ID	EX	MEM	WB		
lw ou sw	IF	ID	EX	MEM	WB		
ULA ou branch		IF	ID	EX	MEM	WB	
lw ou sw		IF	ID	EX	MEM	WB	
ULA ou branch			IF	ID	EX	MEM	WB
lw ou sw			IF	ID	EX	MEM	WB

Despacho múltiplo de instruções

- Requisito para o despacho múltiplo
  - Replicação de componentes
    - Mais ULAs
    - Mais registradores
    - Etc.
- Resultado do despacho múltiplo
  - **CPI abaixo de 1!**
  - IPC: instruções por ciclo
    - IPC = 4 -> quatro instruções finalizadas por ciclo

Despacho múltiplo de instruções

- **Exemplo**
  - Um programa de **simulação de pipoca estourando** executa **10 bilhões de instruções** em um processador com despacho múltiplo de **5 instruções por ciclo**. Se o processador possui uma **frequência de 2GHz** e consegue usar todos seus recursos na execução do programa, qual será o **tempo total de execução?**

Despacho múltiplo de instruções

- Duas formas de despacho múltiplos
  - Despacho **estático**
  - Despacho **dinâmico**
- Problemas a serem tratados
  - Ordenamento das instruções
  - Antidependências
  - Manutenção dos recursos ocupados

Despacho múltiplo de instruções

- **Ordenamento das instruções**
  - Ordem sequencial precisa ser respeitada
  - Problema de **dependências** similar ao pipeline

Com dependência	**Sem dependência**
add **$t2**, $t1, $t0	add $t2, $t1, $t0
add $t3, **$t2**, $s0	add $t3, $t4, $s0
Ou	Ou
add **$t2**, $t1, $t0	add $t2, $t1, $t0
lw $t3, 0(**$t2**)	lw $t3, 0($t7)

Despacho múltiplo de instruções

- **Antidependências**
  - **Dependências de nome**
  - Mesmo local é usado para armazenar dados

```
add $s0, $s1, $s2
sw $s0, 0($s6)
add $s0, $s3, $s4          #sobrescreve $s0
```

  - Resolvido usando outros registradores
    - Renomeação de registradores

Despacho múltiplo de instruções

- **Renomeação de registradores**
  - De

```
add $s0, $s1, $s2
sw $s0, 0($s6)
add $s0, $s3, $s4
sw $s0, 4($s6)
```

Para

```
add $s0, $s1, $s2
sw $s0, 0($s6)
add $s7, $s3, $s4
sw $s7, 4($s6)
```

Despacho múltiplo de instruções

- **Instruções com dependências**
  - **Não podem ser executadas em paralelo**
- **Instruções com antidependências**
  - **Podem ser executadas em paralelo**
    - Dependem de renomeação de recursos (registradores)

DESPACHO MÚLTIPLO ESTÁTICO

Despacho múltiplo estático

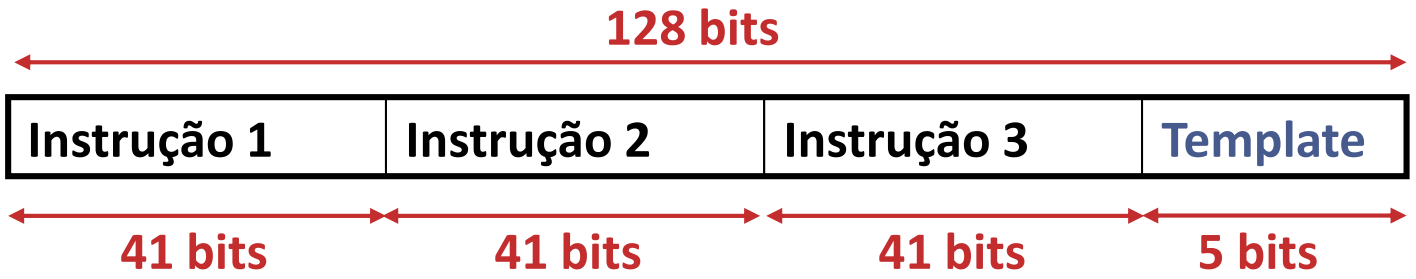
- **Definição**
  - Decisões de paralelismo são tomadas **antes da execução do programa**
    - Durante a **compilação**
  - Paralelismo não muda durante a execução
- Vantagem
  - **Hardware mais simples (- energia, - área)**
- Desvantagens
  - **Menos flexível**
  - **Perda de compatibilidade retroativa**

Despacho múltiplo estático

- Nome comumente usado: **VLIW**
  - *Very long instruction word*
    - Palavra de instrução muito longa
  - Instrução que empacota múltiplas instruções
    - **Pacote de despacho**
    - Múltiplos opcodes, registradores, valores imediatos, etc.

Despacho múltiplo estático

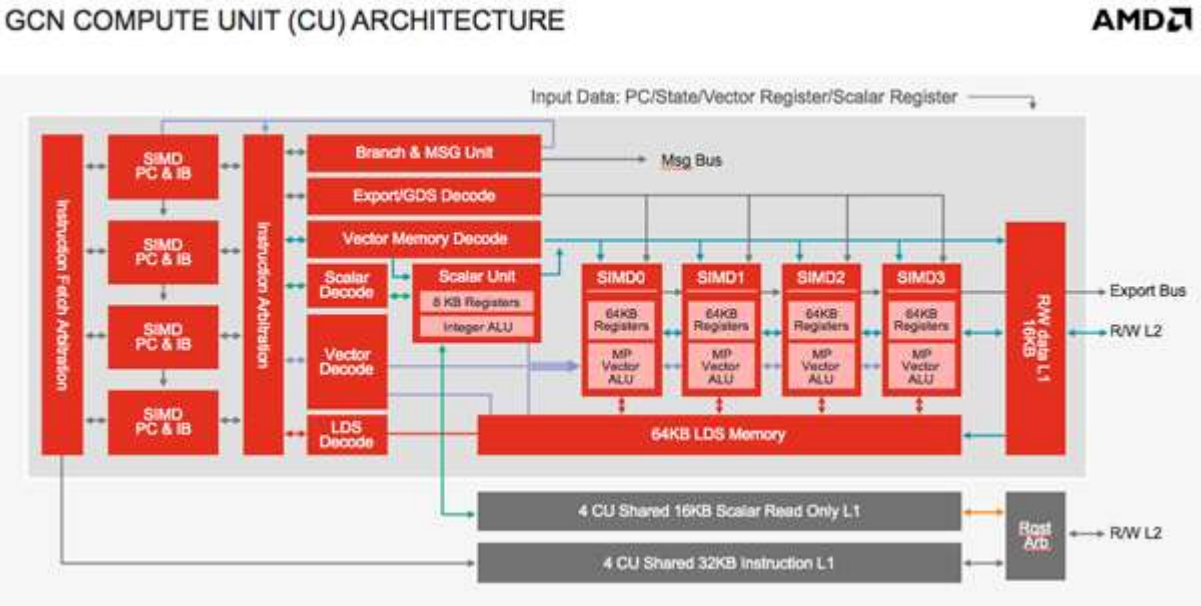
- Exemplo de VLIW
  - Intel IA-64
    - Processadores Itanium e Itanium 2 (2001-)
    - Processador load/store
    - ISA estilo RISC
    - *Bundle* de instruções

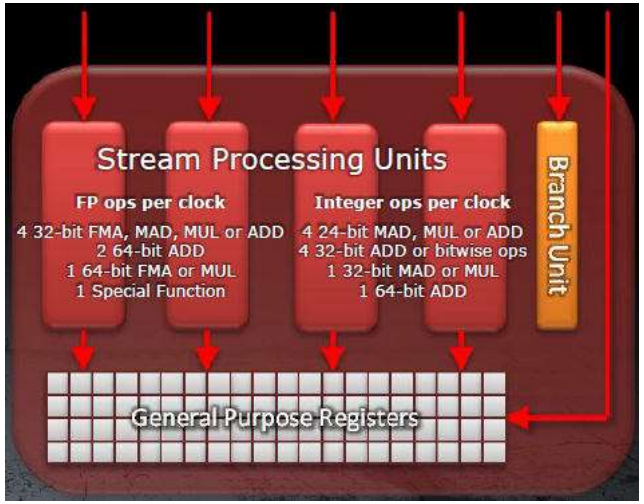


    - Suporte a predicação

Despacho múltiplo estático

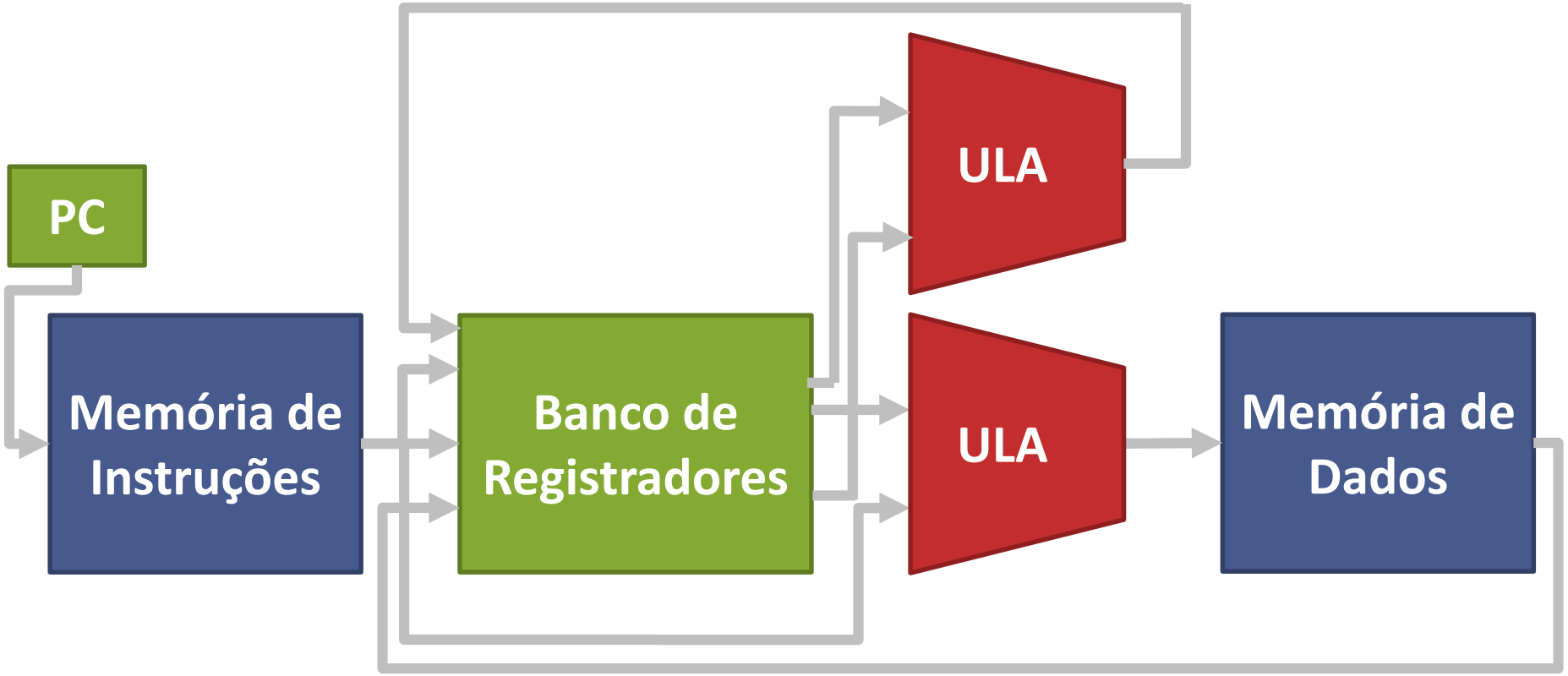
- Exemplo de VLIW
  - Processadores gráficos da AMD
    - VLIW4 ou VLIW5





Despacho múltiplo estático

- **Exemplo didático**
  - MIPS com despacho de duas instruções
    - Uma aritmética ou desvio, outra load ou store



Despacho múltiplo estático

- **Exemplo didático**

Instruções\Ciclos	1	2	3	4	5	6	7
ULA ou branch	IF	ID	EX	MEM	WB		
lw ou sw	IF	ID	EX	MEM	WB		
ULA ou branch		IF	ID	EX	MEM	WB	
lw ou sw		IF	ID	EX	MEM	WB	
ULA ou branch			IF	ID	EX	MEM	WB
lw ou sw			IF	ID	EX	MEM	WB

Despacho múltiplo estático

- Preparação de pacotes de despacho
  - Código base em linguagem de alto nível

```
do {
        vetor[i] = vetor[i] + entrada;
        i++;
} while ( i != tamanho )
```

Despacho múltiplo estático

- Preparação de pacotes de despacho
  - Código base em linguagem de montagem

```
Loop: sll   $t2, $s1, 2
      add   $t2, $t2, $s0#endereço vetor[i]
      lw    $t0, 0($t2)   #t0 = elemento do vetor
      add   $t0, $t0, $s2#soma $s2 ao elemento
      sw    $t0, 0($t2)   #armazena novo valor
      addi  $s1, $s1, 1   #incrementa índice
      bne   $s1, $s3, Loop      #continua enquanto
                                #índice != tamanho
```

Despacho múltiplo estático

- Escalonamento no processador VLIW

	ULA ou desvio	Load ou Store	Ciclo
Loop:	sll $t2, $s1, 2		1
	add $t2, $t2, $s0		2
		lw $t0, 0($t2)	3
	add $t0, $t0, $s2		4
	addi $s1, $s1, 1	sw $t0, 0($t2)	5
	bne $s1, $s3, Loop		6

  - Utilização dos recursos: 7/(2*6) = 7/12 = 0,58

Despacho múltiplo estático

- Preparação de pacotes de despacho
  - ***Loop unrolling:*** **desenrolamento de laço**
    - Feito pelo compilador ou programador
  - Código base em linguagem de alto nível

```
do {    //sabendo que tamanho é múltiplo de 4!
        vetor[i] = vetor[i] + entrada;
        vetor[i+1] = vetor[i+1] + entrada;
        vetor[i+2] = vetor[i+2] + entrada;
        vetor[i+3] = vetor[i+3] + entrada;
        i+=4;
} while ( i != tamanho )
```

Despacho múltiplo estático

- Escalonamento no processador VLIW

	ULA ou desvio	Load ou Store	Ciclo
Loop:	sll $t2, $s1, 2		1
	add $t2, $t2, $s0		2
		lw $t0, 0($t2)	3
	add $t0, $t0, $s2	**lw $t5, 4($t2)**	4
	add $t5, $t5, $s2	**lw $t6, 8($t2)**	5
	add $t6, $t6, $s2	**lw $t7, 12($t2)**	6
	add $t7, $t7, $s2	sw $t0, 0($t2)	7
		sw $t5, 4($t2)	8
	addi $s1, $s1, **4**	**sw $t6, 8($t2)**	9
	bne $s1, $s3, Loop	**sw $t7, 12($t2)**	10

  - Utilização dos recursos: 16/(2*10) = 0,8

DESPACHO MÚLTIPLO DINÂMICO

Despacho múltiplo dinâmico

- Despacho dinâmico: **superescalar**
  - Instruções a serem despachadas em paralelo **selecionadas pelo hardware**
  - **Vantagens** (vs VLIW)
    - Mantém compatibilidade com a mesma ISA
    - Detecção de conflitos e possibilidades em tempo de execução
  - **Desvantagens**
    - Hardware pode se tornar complexo e custoso
      - Por exemplo, possibilidade de execução fora de ordem

Despacho múltiplo dinâmico

- **Funcionamento**
  - Processador avalia uma **janela de instruções**
    - Encontra todas dependências
  - Instruções cujas dependências já foram resolvidas são escalonadas
    - Enviadas para as **estações de reserva**
    - Limitado pela quantidade de recursos disponíveis
  - Quaisquer questões de ordenação são tratadas posteriormente
    - Garantia da **semântica serial** original do código

Despacho múltiplo dinâmico

- Organização de processador superescalar
  - Versão com execução fora de ordem

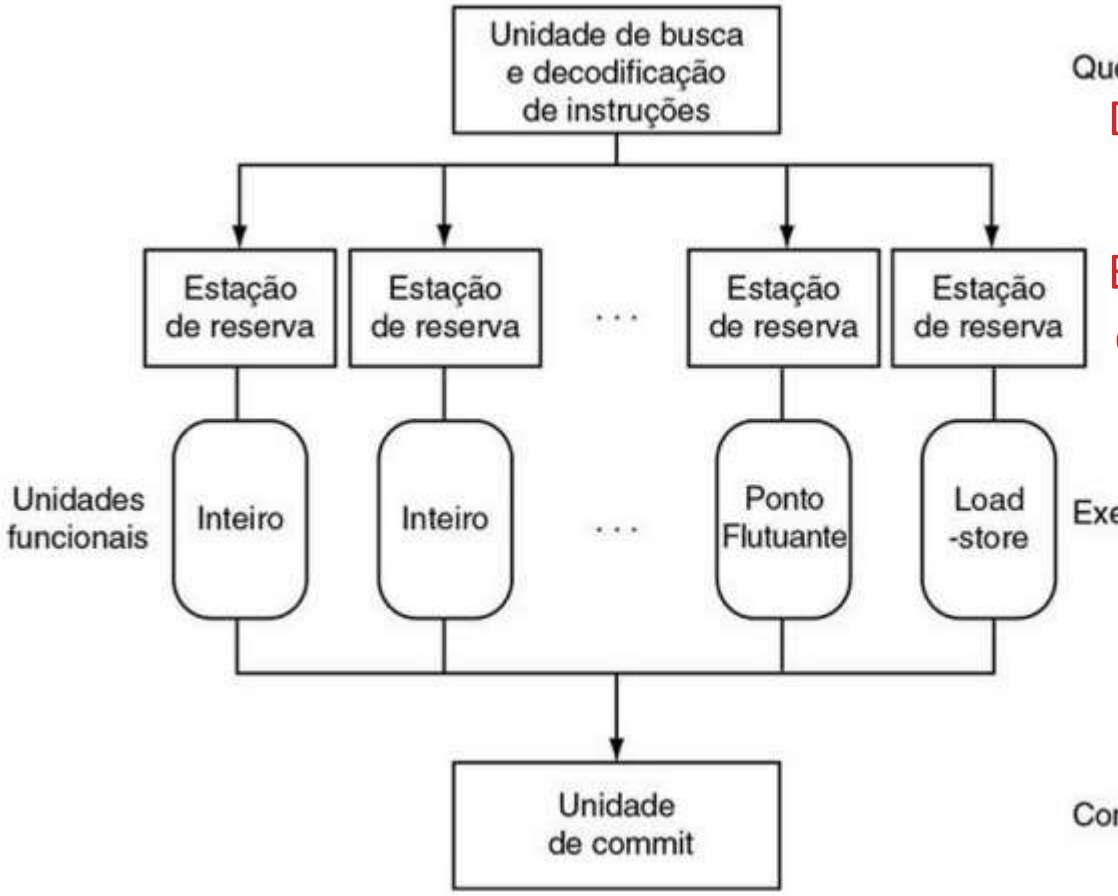
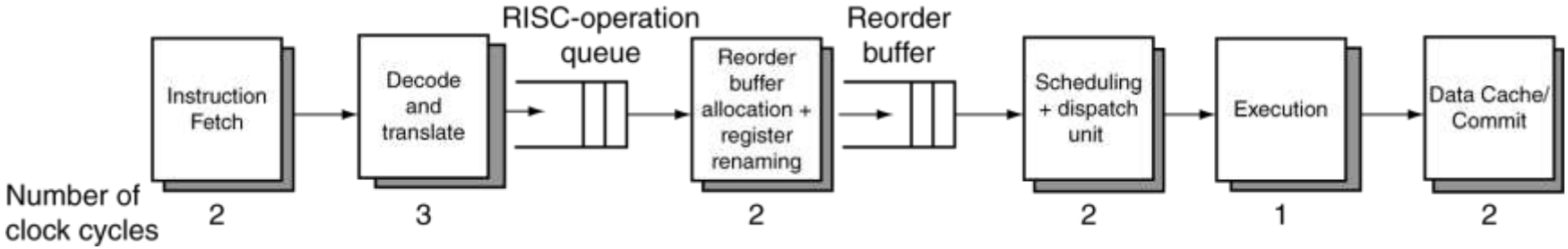


Figura 4.72 do livro *Computer Organization and Design 4th ed.*

Despacho múltiplo dinâmico

- Exemplo de superescalar
  - AMD **Opteron X4**
    - Pipeline
      - 12 estágios para inteiros
      - 17 estágios para ponto flutuante



    - Transforma instruções CISC em operações RISC

Despacho múltiplo dinâmico

- Exemplo de superescalar
  - AMD **Opteron X4**
  - 3 ops/ciclo
  - Pipeline
    - OoO
      - *Out of Order*

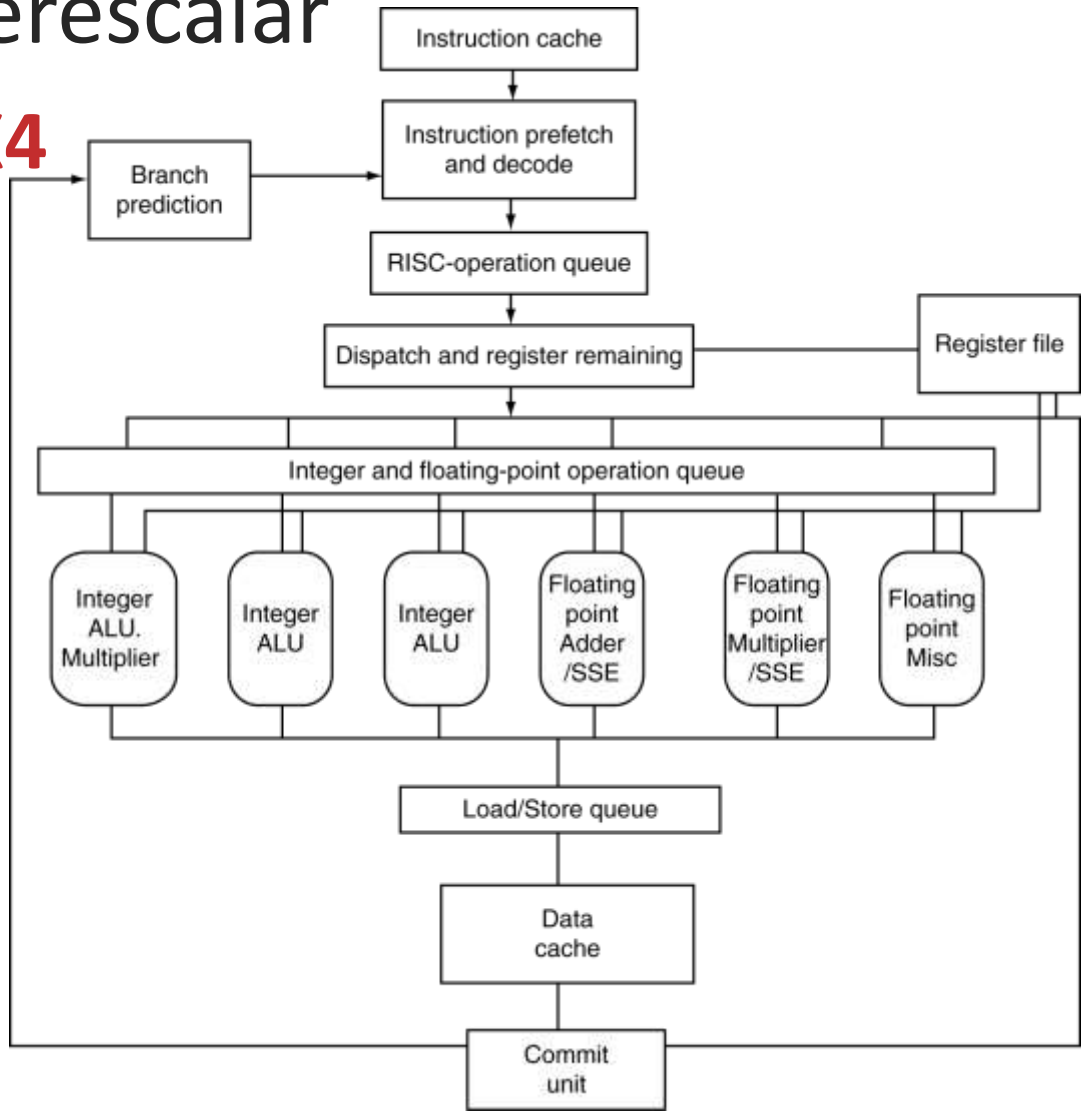
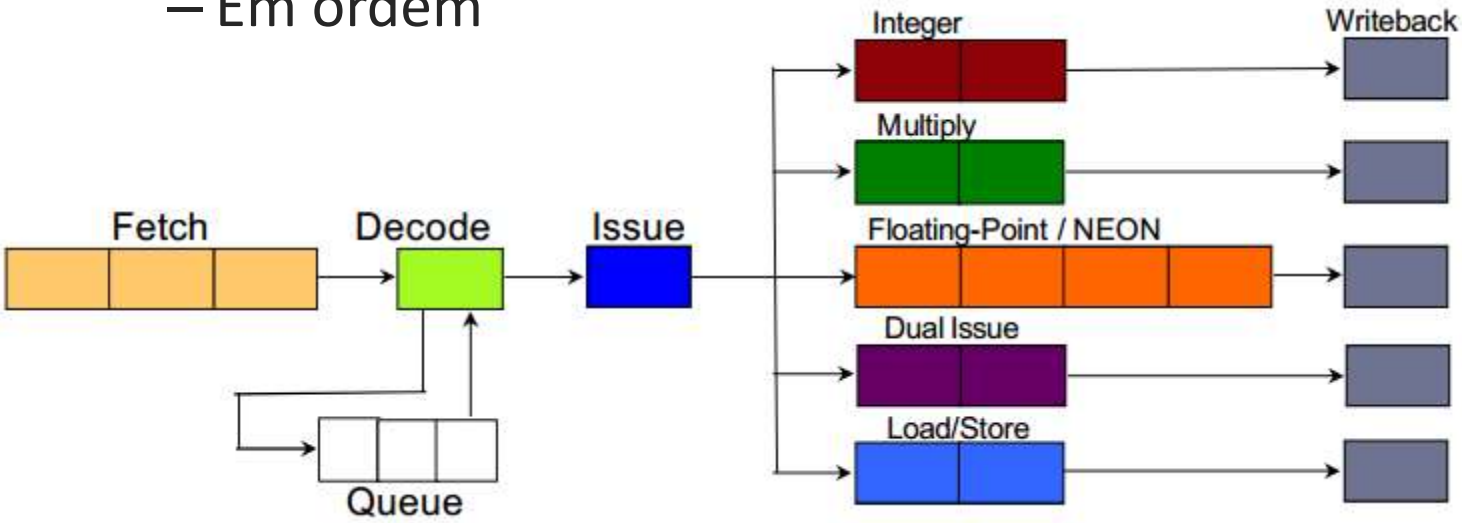


Figura 4.74 do livro *Computer Organization and Design 4th ed.*

Despacho múltiplo dinâmico

- Exemplo de superescalar
  - ARM **Cortex-A7**
    - Núcleo LITTLE em big.LITTLE
    - Despacho de até duas instruções em um ciclo
      - Em ordem



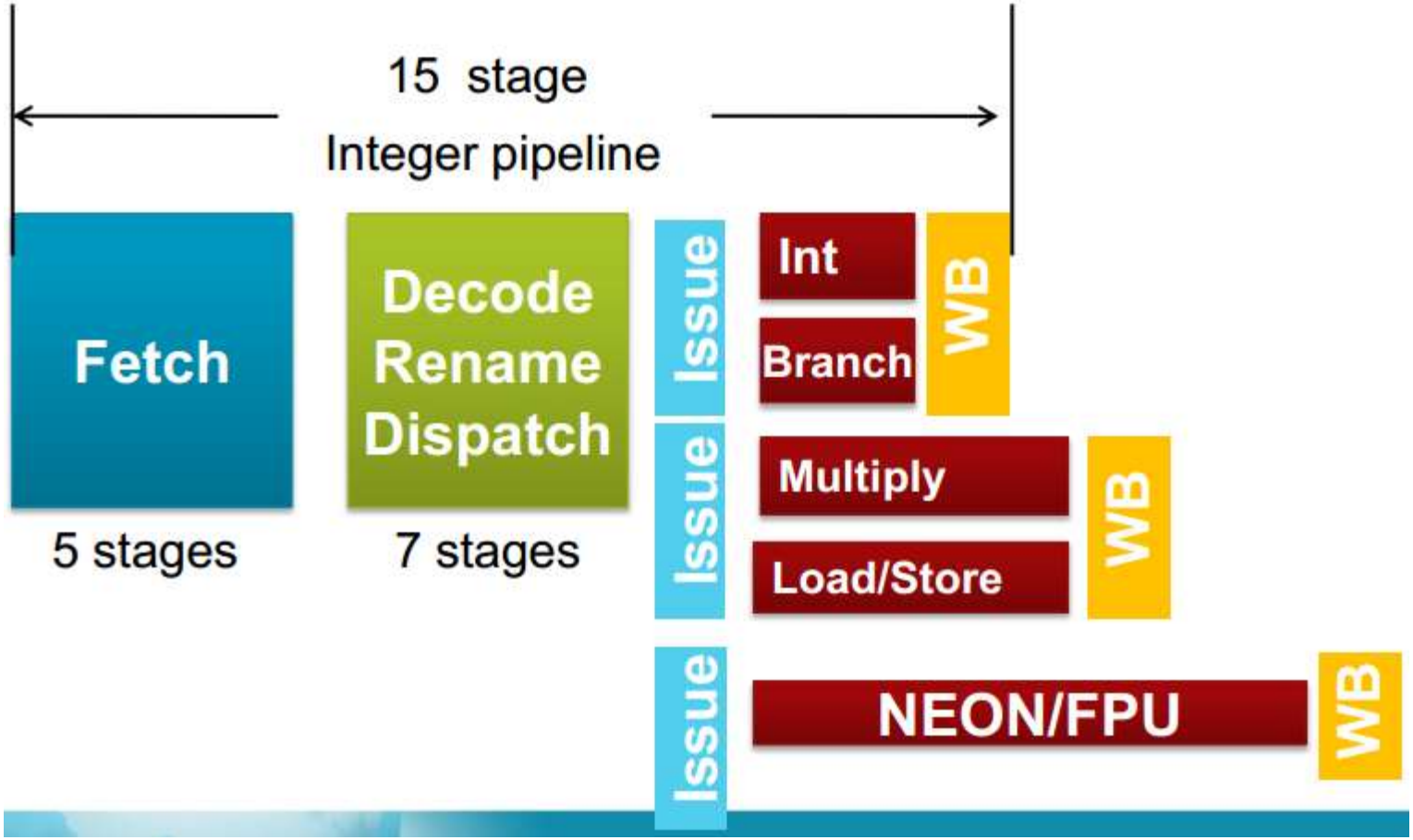
http://community.arm.com/servlet/JiveServlet/previewBody/7612-102-1-11505/Enabling_Mobile_Innovation_with_the_Cortex-A7_Processor.pdf

Despacho múltiplo dinâmico

- Exemplo de superescalar
  - ARM **Cortex-A15**
    - Núcleo big em big.LITTLE
    - Pipeline
      - 15 estágios para inteiros
      - 17 a 25 estágios para ponto flutuante
    - Execução fora de ordem (OoO)
      - Despacho de até três instruções em um ciclo

Despacho múltiplo dinâmico

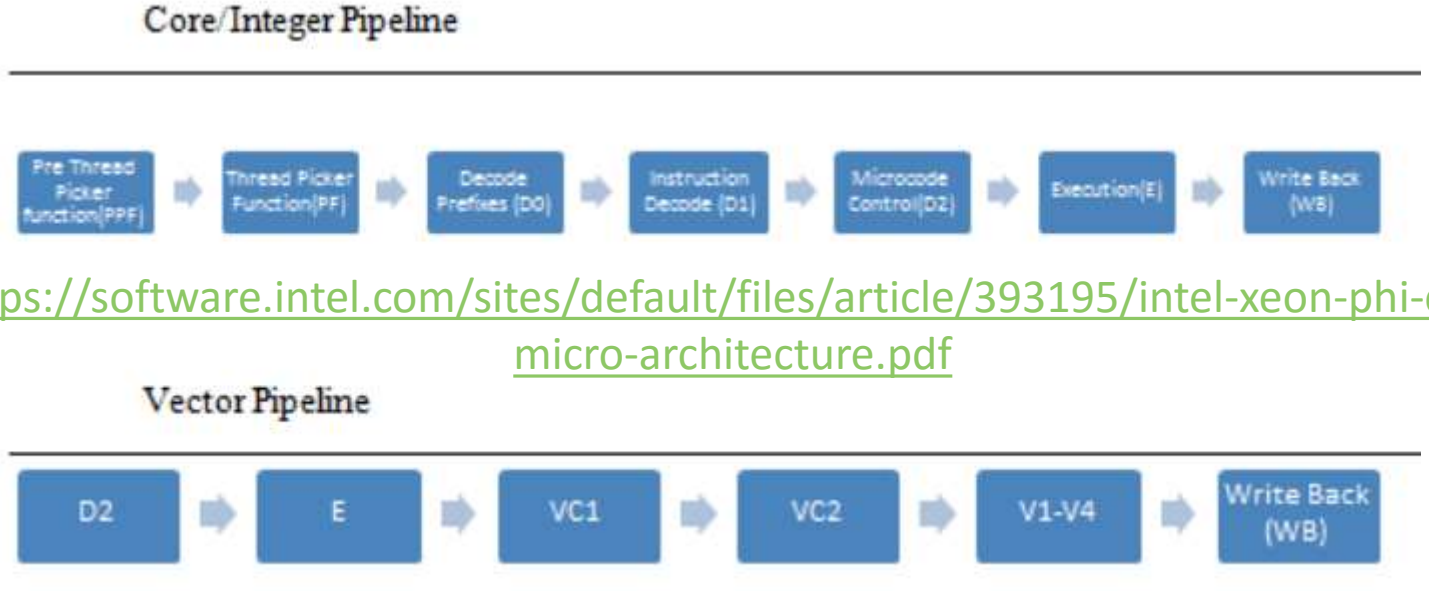
- Exemplo de superescalar
  - ARM **Cortex-A15**



http://www.arm.com/files/pdf/at-exploring_the_design_of_the_cortex-a15.pdf

Despacho múltiplo dinâmico

- Exemplo de superescalar
  - Acelerador **Intel Xeon Phi**
  - Despacho de duas instruções em ordem
  - Pipeline de 7 estágios + 6 para operações sobre vetores



https://software.intel.com/sites/default/files/article/393195/intel-xeon-phi-core-micro-architecture.pdf

Despacho múltiplo dinâmico

- Eficiência energética e pipelining avançado

Processador Intel	Ano	Freq. (MHz)	Estágios de pipeline	Largura do despacho	OoO/especulação	Núcleos	Potência (W)
486	1989	25	5	1	Não	1	5
Pentium	1993	66	5	2	Não	1	10
Pentium Pro	1997	200	10	3	Sim	1	29
Pentium 4 Willamette	2001	2000	22	3	Sim	1	75
Pentium 4 Prescott	2004	3600	31	3	Sim	1	103
Core	2006	2930	14	4	Sim	2	75
Core i5 Nehalem	2010	3300	14	4	Sim	2-4	87
Core i5 Ivy Bridge	2012	3400	14	4	Sim	8	77

Figura 4.73 do livro *Computer Organization and Design 5th ed.*

Despacho múltiplo dinâmico

- Exercício
  - Dado um procedimento de 2000 instruções, quantos ciclos ele levaria em cada um dos seguintes processadores em condições ideais?
    - **Processador superescalar de 10 estágios, IPC médio de 4.**
    - **Processador VLIW de 8 estágios, despacho múltiplo de 5 instruções.**
  - Sabendo que os processadores consomem **80pJ** e **100pJ** por ciclo, respectivamente, qual processador é o mais econômico?

CONSIDERAÇÕES FINAIS

Considerações finais

- **Pipelining**
  - Técnica usada em praticamente todos os processadores atuais
- Despacho múltiplo: mais instruções/ciclo
  - Estático: **VLIW**
    - Tempo de compilação, menos hardware, sem compatibilidade retroativa
  - Dinâmico: **Superescalar**
    - Tempo de execução, mais hardware, com compatibilidade retroativa

Considerações finais

- Próxima aula
  - **Processadores multithread e multicore**

INE5607 – Organização e Arquitetura de Computadores

Unidade Central de Processamento

Aula 17: Processadores com despacho múltiplo de instruções

Prof. Laércio Lima Pilla

laercio.pilla@ufsc.br