

コンピュータアーキテクチャ (10)

坂井 修一

東京大学大学院 情報理工学系研究科 電子情報学専攻
東京大学 工学部 電子情報工学科／電気電子工学科

- ・ はじめに
- ・ アウトオブオーダー処理

はじめに

- 本講義の目的
 - コンピュータアーキテクチャの基本を学ぶ
- 時間・場所
 - 火曜日 8:40 - 10:10 工学部2号館241
- ホームページ（ダウンロード可能）
 - url: <http://www.mtl.t.u-tokyo.ac.jp/~sakai/hard/>
- 教科書
 - 坂井修一『コンピュータアーキテクチャ』（コロナ社、電子情報レクチャーシリーズC-9）
 - 坂井修一『実践 コンピュータアーキテクチャ』（コロナ社）

教科書通りやります
- 参考書
 - D. Patterson and J. Hennessy, Computer Organization & Design、3rd Ed.（邦訳『コンピュータの構成と設計』（第3版）上下（日系B P））
 - 馬場敬信『コンピュータアーキテクチャ』（改訂2版）、オーム社
 - 富田眞治『コンピュータアーキテクチャ I』、丸善
- 予備知識： 論理回路
 - 坂井修一『論理回路入門』、培風館
- 成績
 - 試験＋レポート＋出席

講義の概要と予定（1／2）

1. コンピュータアーキテクチャ入門

ディジタルな表現、負の数、実数、加算器、ALU、フリップフロップ、レジスタ、計算のサイクル

2. データの流れと制御の流れ

主記憶装置、メモリの構成と分類、レジスタファイル、命令、命令実行の仕組み、実行サイクル、算術論理演算命令、シーケンサ、条件分岐命令

3. 命令セットアーキテクチャ

操作とオペランド、命令の表現形式、アセンブリ言語、命令セット、算術論理演算命令、データ移動命令、分岐命令、アドレッシング、サブルーチン、RISCとCISC

4. パイプライン処理（1）

パイプラインの原理、命令パイプライン、オーバヘッド、構造ハザード、データハザード、制御ハザード

5. パイプライン処理（2）

フォワードリング、遅延分岐、分岐予測、命令スケジューリング

6. キャッシュ

記憶階層と局所性、透過性、キャッシュ、ライトスルーとライトバック、ダイレクトマップ型、フルアソシアティブ型、セットアソシアティブ型、キャッシュミス

7. 仮想記憶

仮想記憶、ページフォールト、TLB、物理アドレスキャッシュ、仮想アドレスキャッシュ、メモリアクセス機構

講義の概要と予定 (2 / 2)

8. 基本CPUの設計

ディジタル回路の入力、Verilog HDL、シミュレーションによる動作検証、アセンブラ、基本プロセッサの設計、基本プロセッサのシミュレーションによる検証

9. 命令レベル並列処理(1)

並列処理、並列処理パイプライン、VLIW、スーパスカラ、並列処理とハザード

10. 命令レベル並列処理(2)

静的最適化、ループアンローリング、ソフトウェアパイプライン、トレーススケジューリング

11. アウトオブオーダー処理

インオーダーとアウトオブオーダー、フロー依存、逆依存、出力依存、命令ウィンドウ、リザベーションステーション、レジスタリネーミング、マッピングテーブル、リオーダーバッファ、プロセッサの性能

12. 入出力と周辺装置

周辺装置、ディスプレイ、二次記憶装置、ハードウェアインタフェース、割り込みとポーリング、アービタ、DMA、例外処理

試験: 7月後半

11. アウトオブオーダー処理

■ 内容

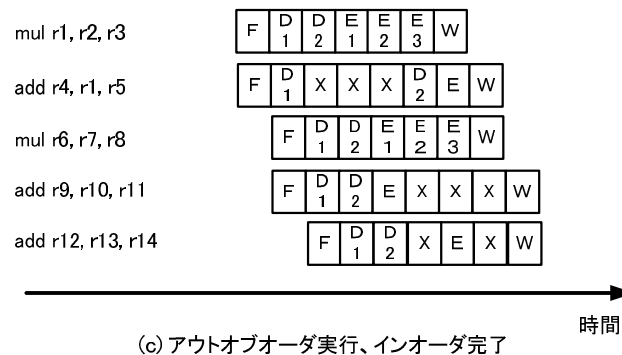
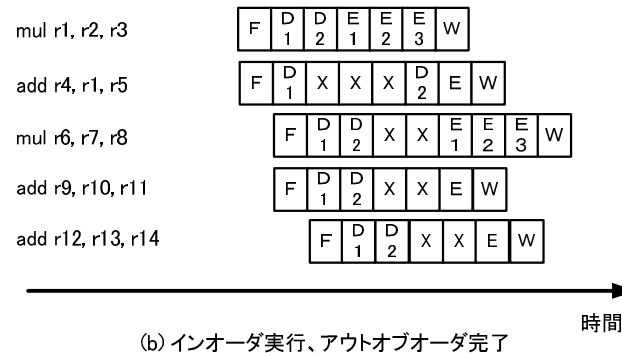
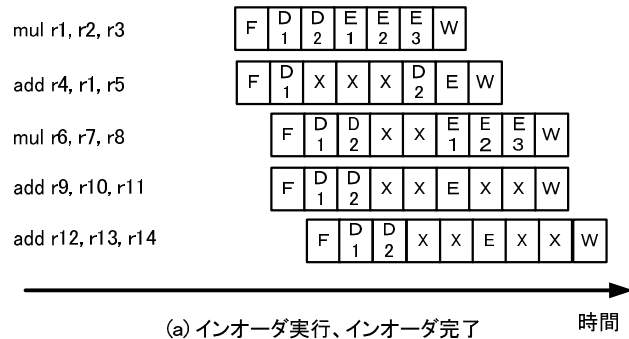
- アウトオブオーダー処理
 - ・ アウトオブオーダー処理とはなにか
 - ・ データ依存再考
 - ・ アウトオブオーダー処理の機構
- レジスタリネーミング
 - ・ ソフトウェアによるレジスタリネーミング
 - ・ ハードウェアによるレジスタリネーミング(1)
 - マッピングテーブル
 - ・ ハードウェアによるレジスタリネーミング(2)
 - リオーダーバッファ
- スーパスカラプロセッサの構成
 - ・ アウトオブオーダー処理を行うプロセッサの構成
 - ・ プロセッサの性能

アウトオブオーダー処理とはなにか

■ アウトオブオーダー処理

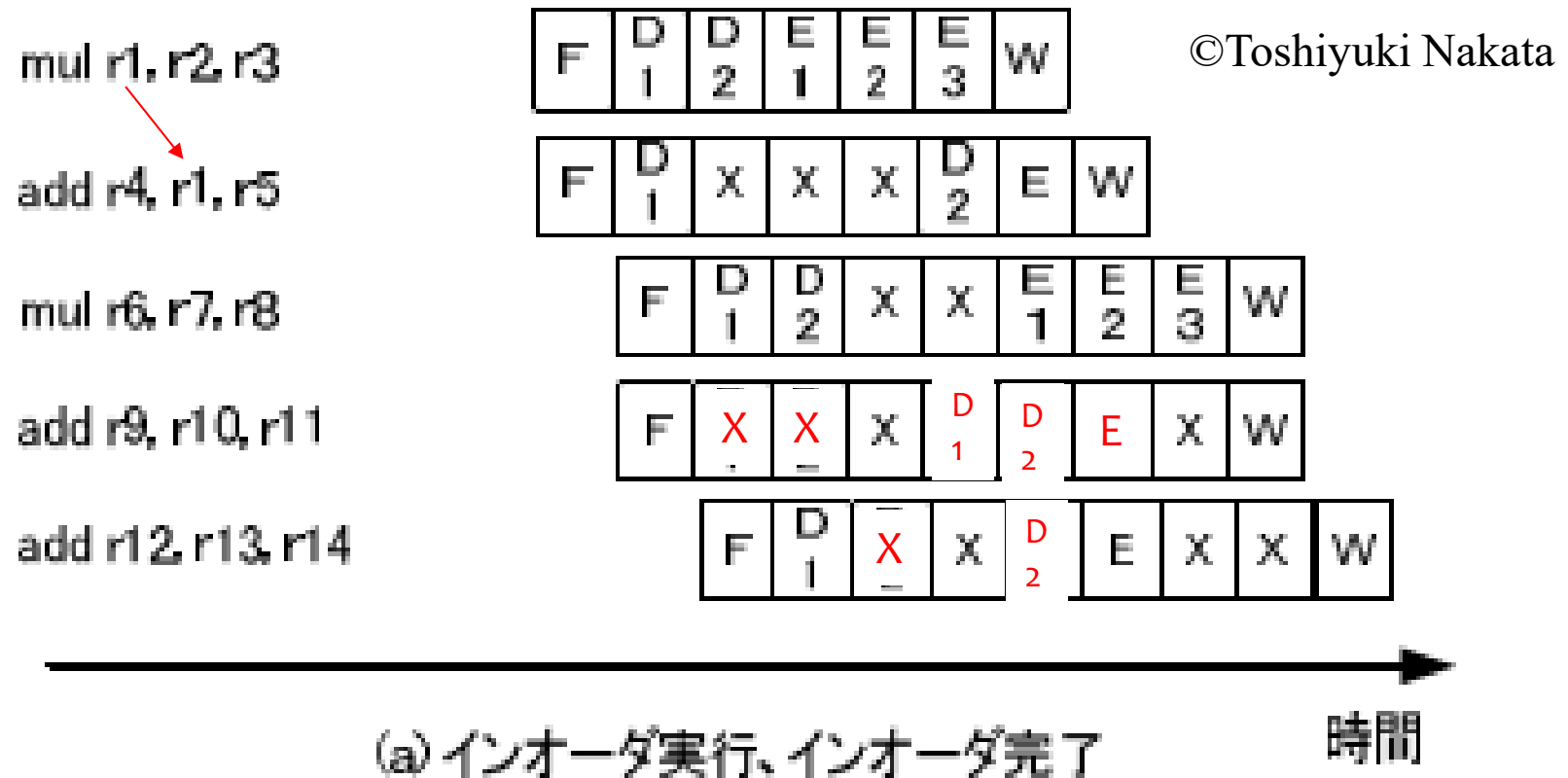
- プログラムの意味を変えない範囲で命令実行・完了の順序を変更し、並列度をあげる処理
 - cf. インオーダー処理: 命令を動的に入れ替えることをしない処理
- 動的スケジューリングの一種
 - ・ 動的スケジューリング: 実行時に行うスケジューリング
- アウトオブオーダー実行
 - ・ 命令をEステージに入れる順番を入れ替える
- アウトオブオーダー完了
 - ・ 実行結果をレジスタに格納する順番を入れ替える

アウトオブオーダー処理の例



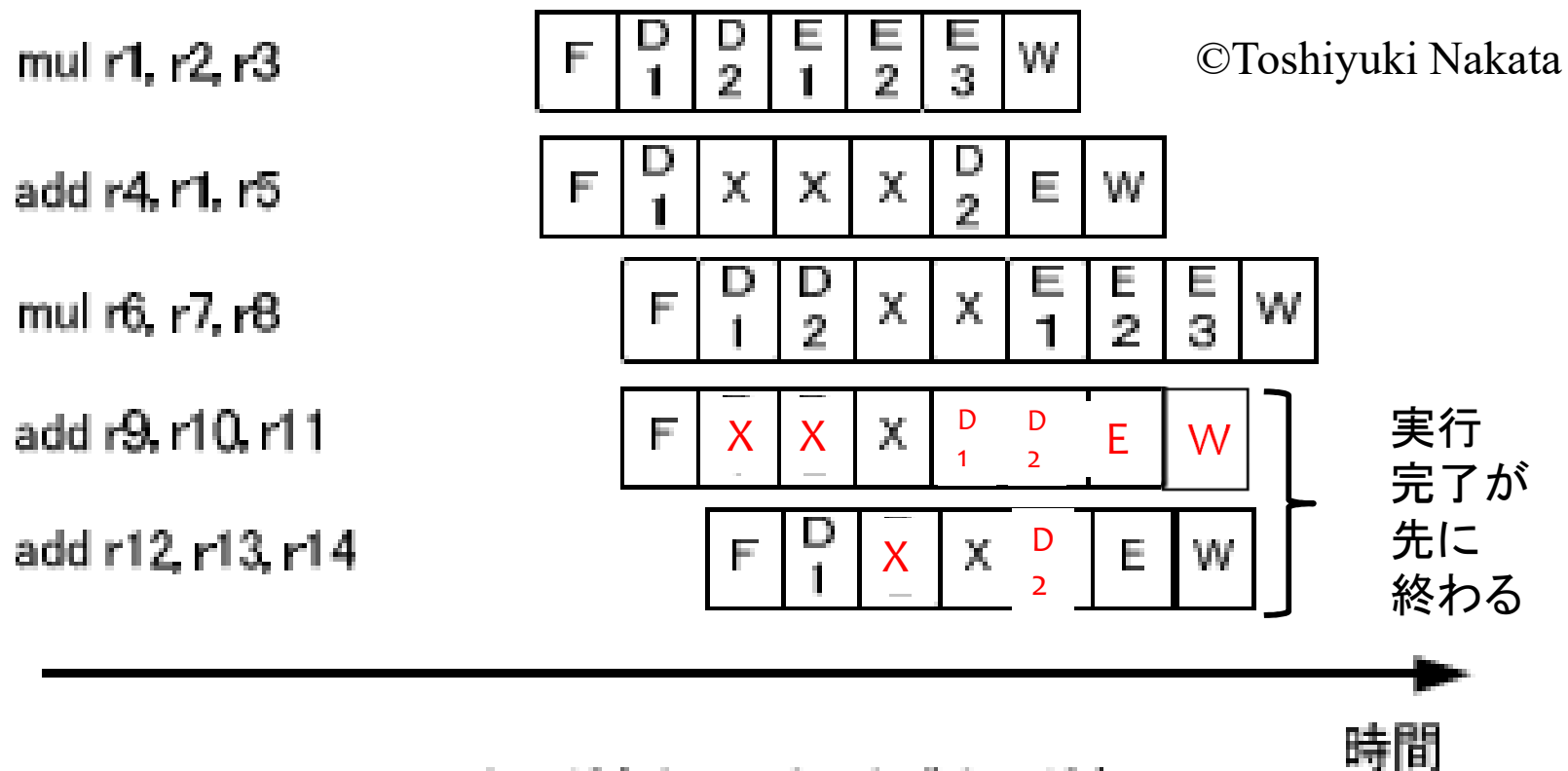
- (a) 11クロック
- (b) 10クロック
- (c) 9クロック
- (d) 8クロック

In Order実行 (Issue)/In order 完了 (Completion)



11クロック

In order実行 Out of Order完了



(b) インオーダー実行、アウトオブオーダー完了

10クロック

Out of Order実行/In Order完了

メモ:あとで、HWの説明で詳述するように、
この場合パイプラインレジスタに関する
制約がHWの追加により軽減される。
また乗算パイプラインはE1/E2/E3の3段の
パイプラインステージとする。



(c) アウトオブオーダー実行、インオーダー完了

Out of Order実行/Out of Order完了

メモ:あとで、HWの説明で詳述するように、
この場合パイプラインレジスタに関する
制約がHWの追加により軽減される。
また乗算パイプラインはE1/E2/E3の3段の
パイプラインステージとする。

mul r1, r2, r3



©Toshiyuki Nakata

add r4, r1, r5



mul r6, r7, r8



乗算パイプラインは
前の命令でE2に進めば
E1に移れると仮定

add r9, r10, r11



Eステージの実行が逆転
Wステージの完了も逆転

add r12, r13, r14

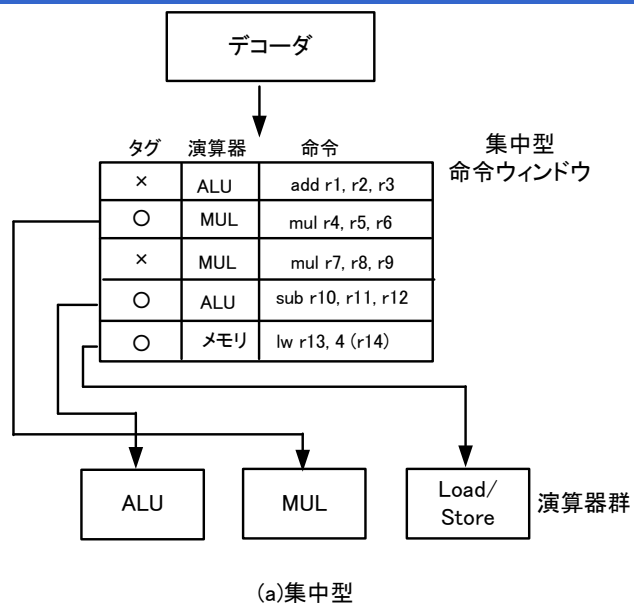


Wステージの完了逆転



(d) アウトオブオーダー実行、アウトオブオーダー完了

アウトオブオーダー処理の機構



アウトオブオーダー処理の実現

⇒ **命令ウィンドウ**

命令ウィンドウ

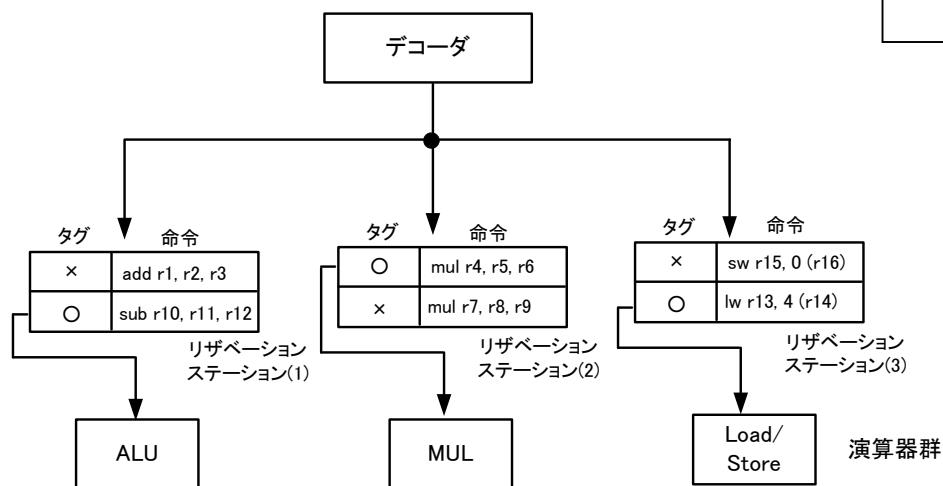
= 実行可能な命令を選び出す機構

命令ウィンドウ

- ・ 集中型

- ・ 分散型

= **リザーベーションステーション**



(b)分散型(リザーベーションステーション)

データ依存性再考

■ データ依存の分類

- フロー依存
 - ・ 命令 A で書き込んだ値を後続の命令 B で読み出すことで起こる $A \Rightarrow B$ の依存関係。真の依存関係ともいう
- 逆依存
 - ・ 命令 A で読み出したレジスタ(メモリ語)に後続の命令 B が書き込みを行うことで起こる $A \Rightarrow B$ の依存関係
- 出力依存
 - ・ 命令 A で書き込んだレジスタ(メモリ語)に後続の命令 B が再度書き込みを行うことで起こる $A \Rightarrow B$ の依存関係

逆依存と出力依存は、主に**レジスタ数の不足**からくる依存

cf. インオーダーの場合

- ・ フロー依存だけを意識すればよく、フォワーディングで解決していた

データ依存の例

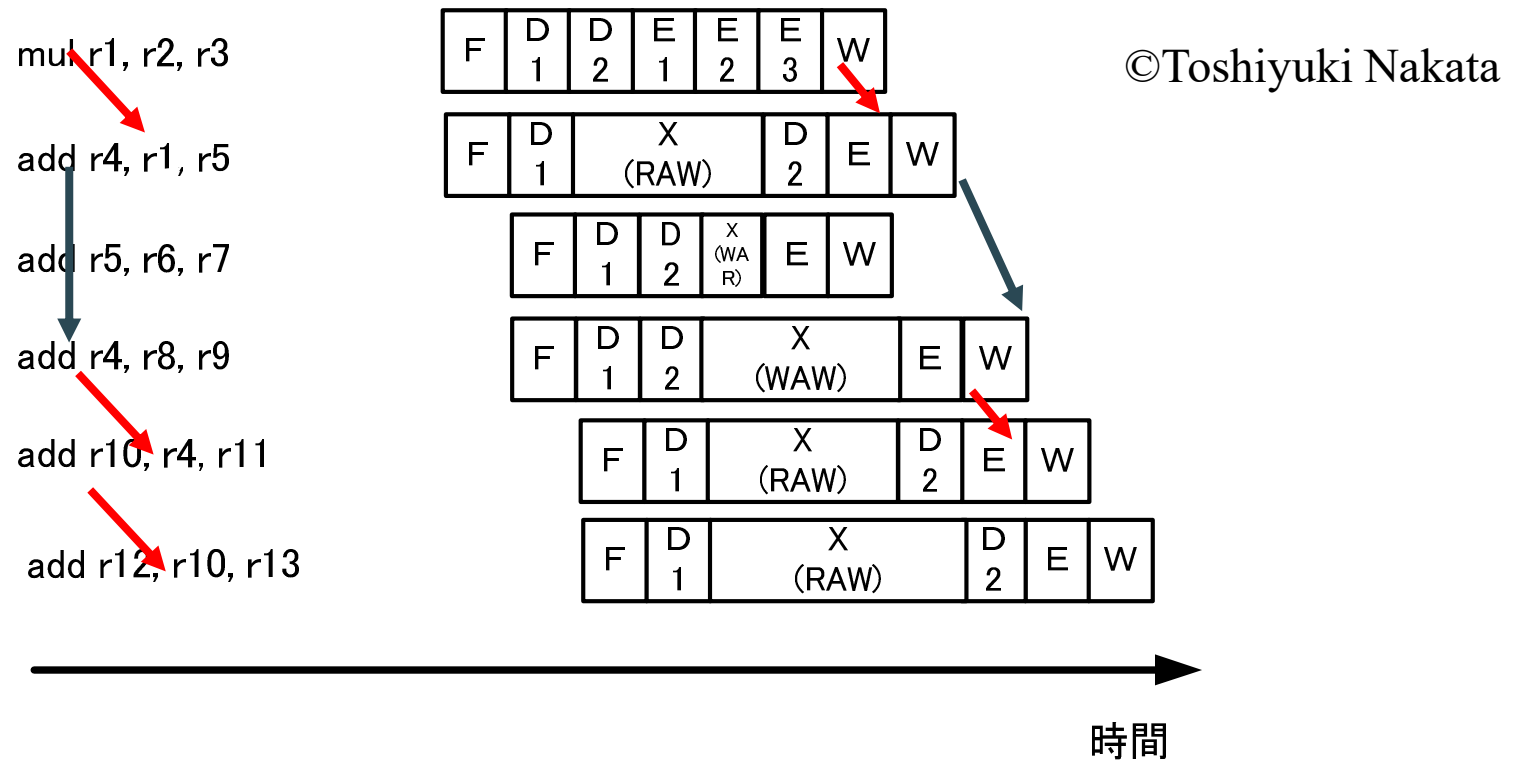
- | | | |
|---|-------|---|
| ①mul r1, r2, r3($r1 \leftarrow -r2 * r3$) | フロー依存 | |
| ②add r4, r1, r5($r4 \leftarrow -r1 + r5$) | 逆依存 |  |
| ③add r5, r6, r7($r5 \leftarrow -r6 + r7$) | 出力依存 |  |
| ④add r4, r8, r9($r4 \leftarrow -r8 + r9$) | | |
| ⑤add r10, r4, r11($r10 \leftarrow -r4 + r11$) | | |
| ⑥add r12, r10, r13($r12 \leftarrow -r10 + r13$) | | |
- Diagram annotations: A red arrow points from r1 in instruction 1 to r1 in instruction 2. A blue arrow points from r4 in instruction 2 to r4 in instruction 4. A red arrow points from r4 in instruction 4 to r10 in instruction 5. A red arrow points from r10 in instruction 5 to r12 in instruction 6.

◆フロー依存: ① $\Rightarrow$ ② ($r1$)、
④ $\Rightarrow$ ⑤ ($r4$)、⑤ $\Rightarrow$ ⑥ ($r10$)

◆逆依存: ② $\Rightarrow$ ③ ($r5$)

◆出力依存: ② $\Rightarrow$ ④ ($r4$)

例題のパイプライン実行



アウトオブオーダー実行が可能であっても、3種類の依存関係からくるハザードによって、実行時の並列度が下がる

レジスタリネーミング

- レジスタリネーミング

- レジスタ番地のつけかえによる逆依存、出力依存の解消

- やりかた

- ソフトウェア
 - マッピングテーブル
 - リオーダーバッファ

ソフトウェアによるレジスタリネーミング

- ソフトウェアによるレジスタリネーミング
= 機械語プログラムの書き換え

- ① mul r1, r2, r3
- ② add r4, r1, r5
- ③ add r14, r6, r7
- ④ add r15, r8, r9
- ⑤ add r10, r15, r11
- ⑥ add r12, r10, r13

mul r1, r2, r3

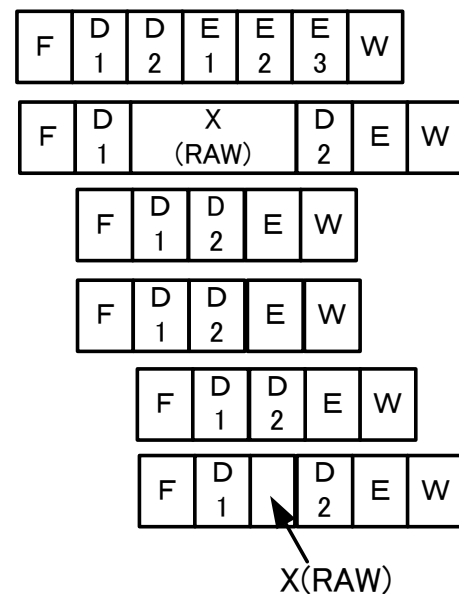
add r4, r1, r5

add r14, r6, r7

add r15, r8, r9

add r10, r15, r11

add r12, r10, r13



時間

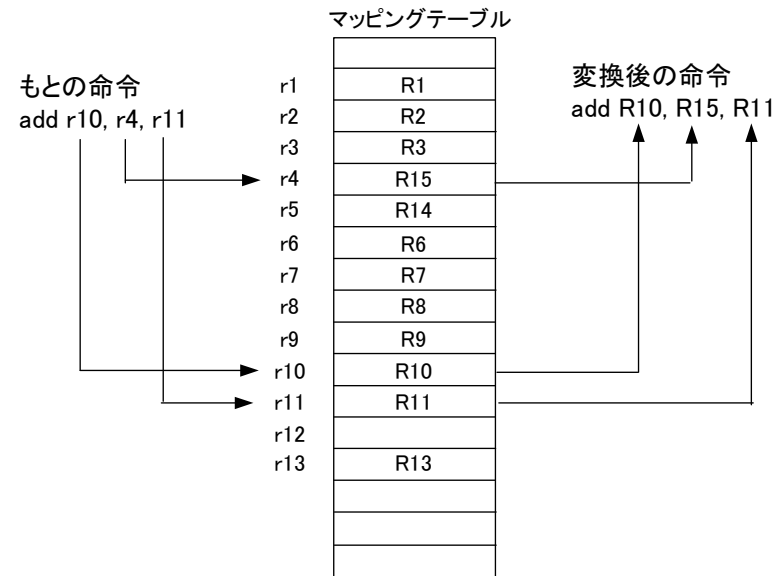
⇒ 3クロックの実行時間短縮

ソフトウェアによるリネーミングの問題点

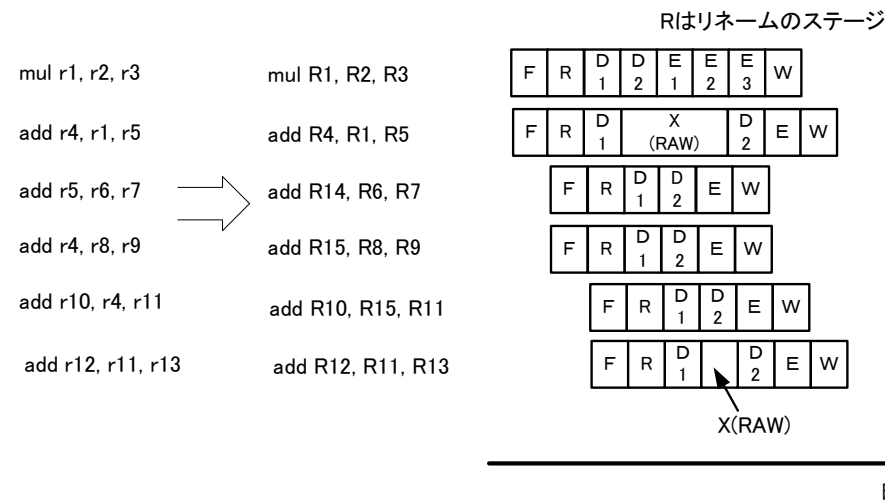
- (1) 機械語プログラムで指定できるレジスタ数には限界がある
- (2) CPUのアーキテクチャの細部(特に並列動作可能なユニット数)にプログラムが影響を受けるため、透過性・互換性が失われる
- (3) 機械語プログラムの変換の手間がかかる

ハードウェアによるレジスタリネーミング(1)

マッピングテーブル



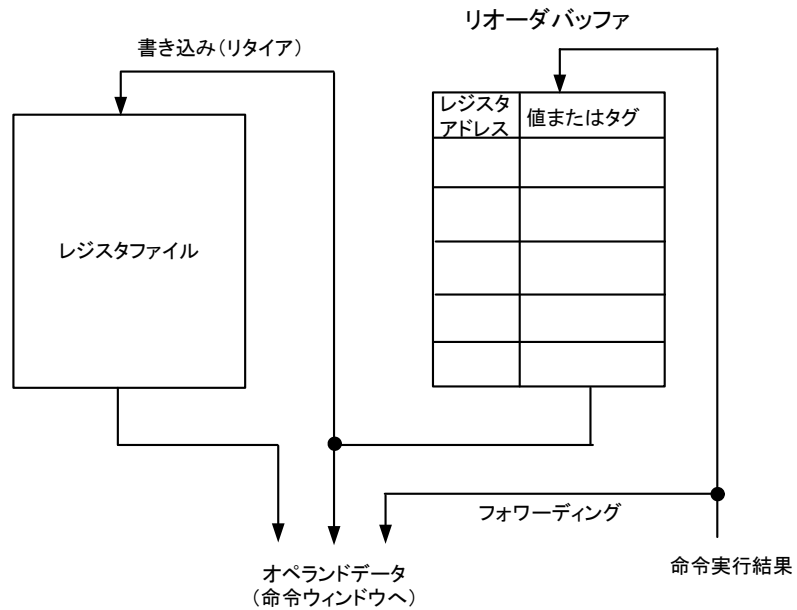
(a) マッピングテーブルによる命令の変換



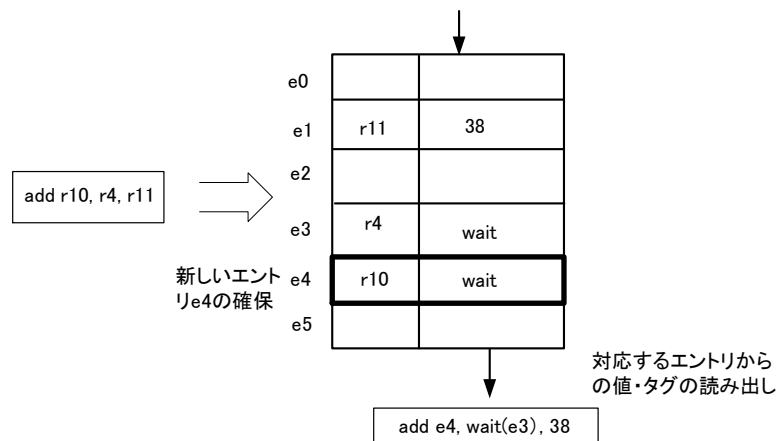
(b) マッピング機構を入れたパイプライン

ハードウェアによるレジスタリネーミング(2)

リオーダーバッファ



(a)リオーダーバッファによるリネーミング



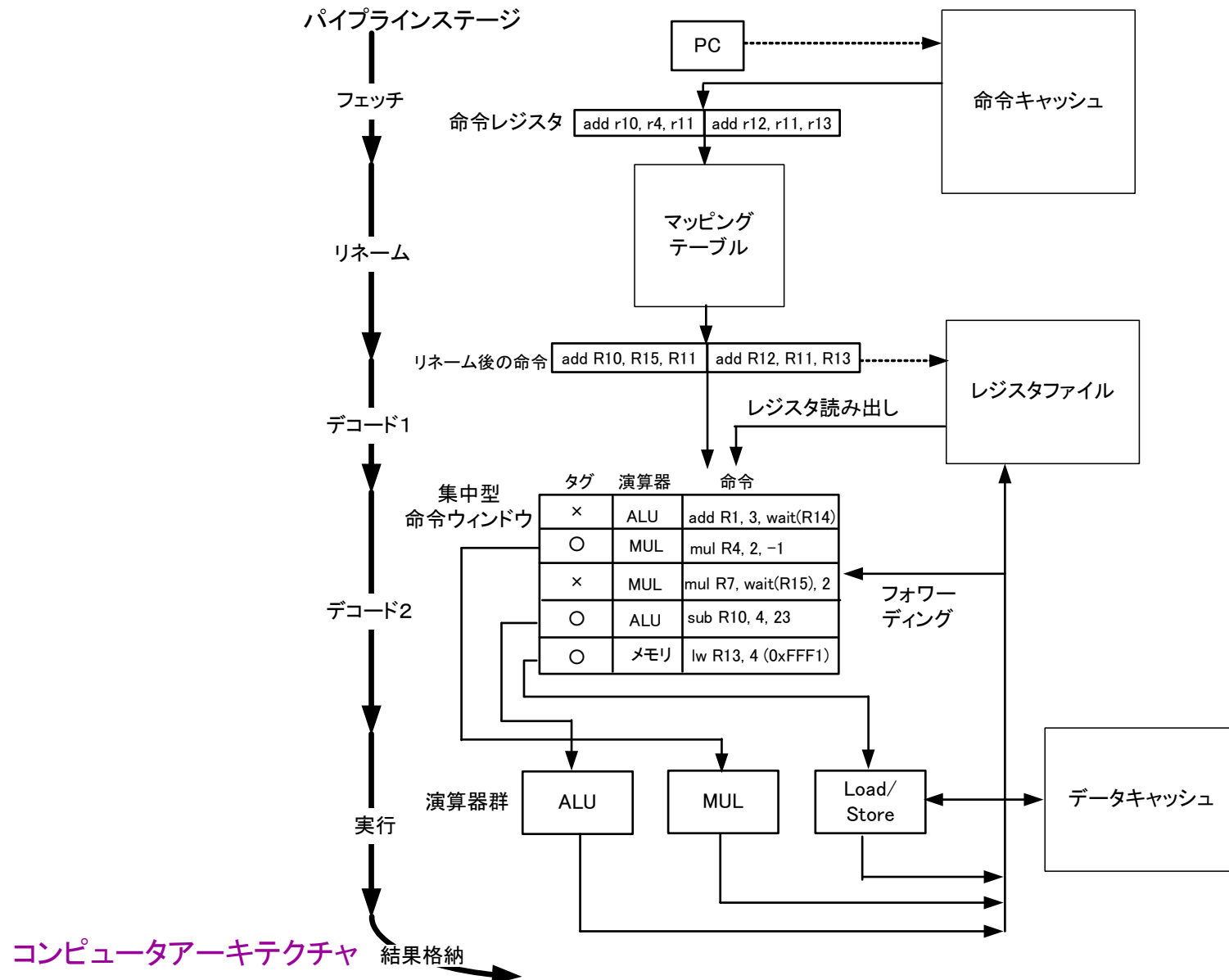
コンピュータアーキテクチャ 命令ウィンドウへ

(b)リオーダーバッファの動作

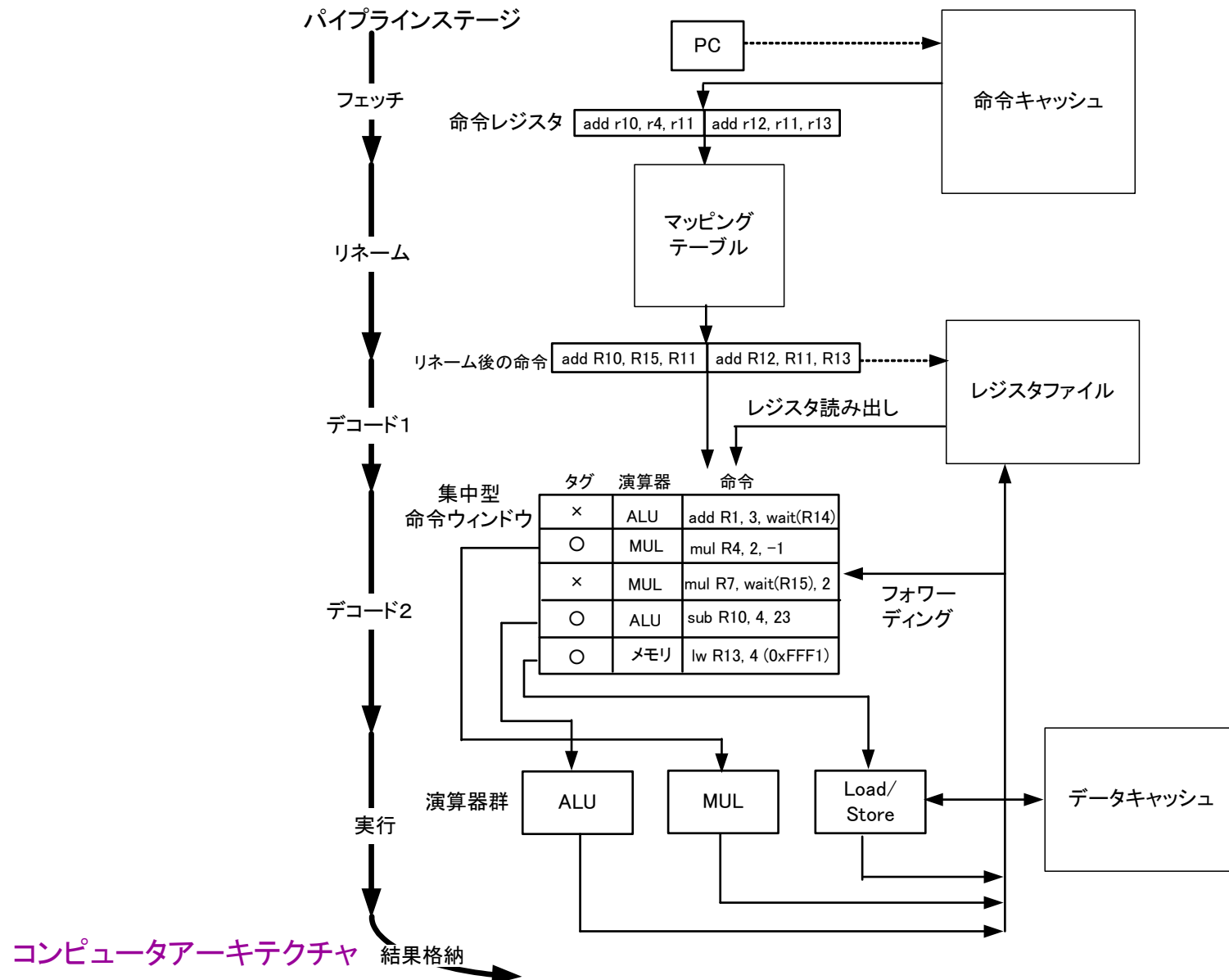
マッピングテーブル vs リオーダーバッファ

	マッピングテーブル	リオーダーバッファ
ハードウェア 機構・動作	単純	複雑
パイプライン長	+ 1 ステージ	増えない

アウトオブオーダー処理を行うプロセッサの構成(1)



アウトオブオーダー処理を行うプロセッサの構成(1)





プロセッサの性能

■ 性能指標(例)

クロックあたりの平均実行命令数 × クロック周波数

－ クロックあたりの平均実行命令数

・ 増やす方法

- － フォワーディング
- － 命令スケジューリング
- － 分岐予測
- － キャッシュ
- － 命令レベル並列処理
- － アウトオブオーダー処理

・ 減る要因

- － 分岐予測の失敗
- － キャッシュミス
- － TLBミス
- － ページフォルト
- － ハザード

－ クロック周波数

- ・ パイプラインの各ステージの複雑さによって決まる