

計算機アーキテクチャ特論 (Advanced Computer Architectures)

9. アウトオブオーダープロセッサ ステートと例外回復

1

動的スケジューリング(アウトオブオーダー実行)

- (1) DIV.D F0, F2, F4
- (2) ADD.D F10, F0, F8
- (3) SUB.D F12, F8, F14

- DIV.DとADD.Dの依存がパイプラインをストールさせ、SUB.D命令の実行を阻害
- SUB.Dはパイプラインのどの命令にもデータ依存しない
- プログラム順序に従って命令を実行するという制約を取り除くことで、この制限を解消

2

Adapted from Computer Organization and Design, Patterson & Hennessy, © 2005

例外 (exception), 割り込み (interrupt)

- (希に)発生する好ましくないイベント
 - 分岐予測ミス
 - 頻度
 - 算術演算オーバーフロー/アンダーフロー
 - ページ・フォールト(HDDへのアクセス)
 - メモリ保護違反
 - 未定義命令の使用
 - 入出力(I/O)からのリクエスト
 - ブレークポイント(プログラマが設定する割り込み)
 - ...

3

Adapted from Superscalar Microprocessor Design, Mike Johnson

投機的実行 (speculative execution)

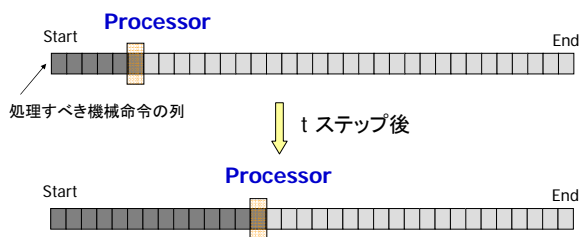
- 投機的実行
 - 割り込みが発生しないという**仮定**のもとで、命令をフェッチおよび実行することで高い性能を得る。
- 回復 (recovery, repair)
 - 誤った**仮定**に基づいて実行された命令の影響を取り消す処理。
- 再開 (restart)
 - 回復の後に、正しい命令列を作り直す処理。
- 正確な割り込み
 - 割り込みを発生させた命令より前の命令列は完了
 - 割り込みを発生させた命令以降を回復、再開

4

Adapted from Superscalar Microprocessor Design, Mike Johnson

古典的なプロセッサの実行モデル

- 1サイクルに1命令を処理する古典的なプロセッサの実行モデル
- 正確な割り込み
 - 割り込みを発生させた命令より前の命令列は完了
 - 割り込みを発生させた命令以降を回復、再開

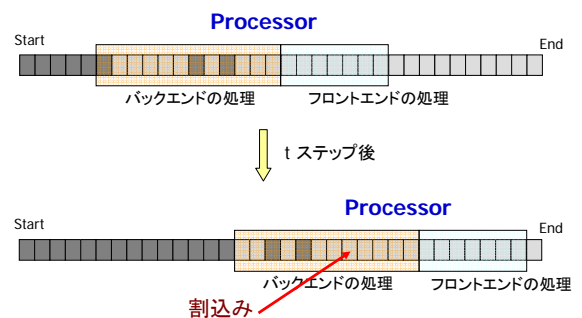


5

Adapted from Superscalar Microprocessor Design, Mike Johnson

高性能プロセッサの実行モデル

- 多数の命令を並列処理する高性能プロセッサの実行モデル



6

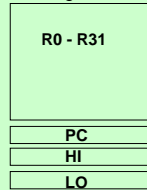
Adapted from Superscalar Microprocessor Design, Mike Johnson

MIPS R3000 Instruction Set Architecture (ISA)

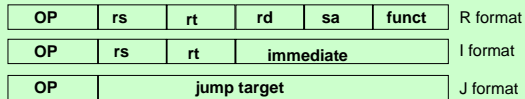
Instruction Categories

- Computational
- Load/Store
- Jump and Branch
- Floating Point
 - coprocessor
- Memory Management
- Special

Registers



3 Instruction Formats: all 32 bits wide



Adapted from Superscalar Microprocessor Design, Mike Johnson

7

効率のよい回復とステート情報

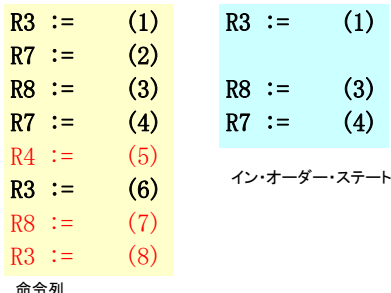
- 論理レジスタのみの状態(ステート)を考える.
 - 主記憶は書き込み頻度が低いので異なる技術を利用
- 効率よい回復のため, 少なくとも2組のステート情報が必要
 - 計算に必要なステートの集合
 - 例外発生時に利用するステートの集合

Adapted from Superscalar Microprocessor Design, Mike Johnson

8

イン・オーダー・ステート (in-order state)

- (5, 7, 8) の命令は処理中, その他は完了している.
- 完了命令だけから成るもっとも長い連続命令列による最新の代入操作によって構成される状態



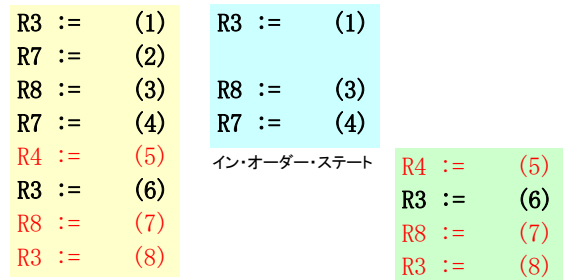
命令列

Adapted from Superscalar Microprocessor Design, Mike Johnson

9

先見ステート (lookahead state)

- 最初の未完了命令から命令列の末尾に至るまでの代入操作のすべてから構成されるステート



命令列

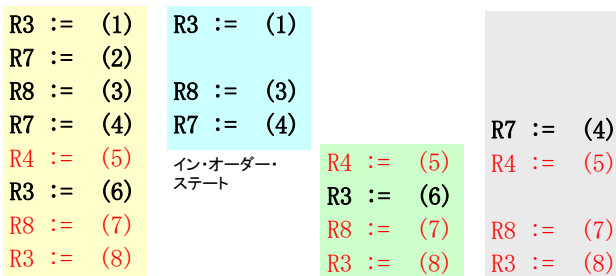
Adapted from Superscalar Microprocessor Design, Mike Johnson

先見ステート

10

アーキテクチャ・ステート (architectural state)

- 現在実行中の命令列の末尾からみて各レジスタに対する最新の代入操作から構成されるステート



命令列

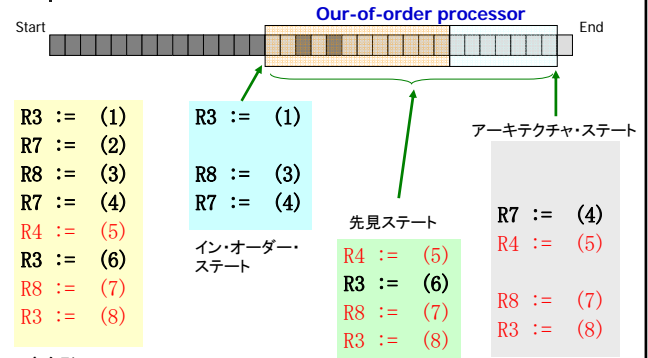
先見ステート

アーキテクチャ・ステート

Adapted from Superscalar Microprocessor Design, Mike Johnson

11

3つのステートの関係



命令列

Adapted from Superscalar Microprocessor Design, Mike Johnson

12

Exercise

氏名, 学籍番号, 年月日 Advance
学籍番号マーク欄(右詰で)

The image shows a sample of a Japanese university entrance exam form. It is a white sheet of paper with a header section and a main body. The header section contains fields for 'Name' (氏名), 'Student ID' (学籍番号), 'Date' (年月日), and 'Advance' (Advance). The main body is a large table with multiple rows and columns. A red box highlights the 'Student ID' field in the header, and another red box highlights the 'Student ID mark' (学籍番号マーク欄) field, which is located on the right side of the form. Arrows point from the text labels to the corresponding fields on the form.

Adapted from *Computer Organization and Design*, Patterson & Hennessy, © 2005

13

イン・オーダー・ステート, 先見ステート, アーキテクチャ・ステートを求めよ

R3 := (1)
R7 := (2)
R8 := (3)
R7 := (4)
R4 := (5)
R3 := (6)
R2 := (7)
R2 := (8)
R7 := (9)

命令列

赤色の命令はまだ完了していない

Adapted from *Superscalar Microprocessor Design*, Mike Johnson

14

Adapted from *Superscalar Microprocessor Design*, Mike Johnson

14

Register	Value
R3	(1)
R7	(2)
R8	(3)

Adapted from *Superscalar Microprocessor Design*, Mike Johnson

15

分岐予測ミスからの回復の例

Adapted from *Superscalar Microprocessor Design*, Mike Johnson

16 |

例外回復のためのバッファリング手法

- チェックポイント回復 (checkpoint repair)
- ヒストリ・バッファ (history buffer)
- リオーダー・バッファ (reorder buffer)
- フューチャ・ファイル (future file)

Adapted from *Superscalar Microprocessor Design*, Mike Johnson

17

Adapted from *Superscalar Microprocessor Design*, Mike Johnson

17

チェックポイント回復 (checkpoint repair)

- 分岐命令のたびにアーキテクチャ・ステートのバックアップを生成
- 命令が完了するたびにすべての空間を更新、バックアップのすべてのレジスタが更新されるとインオーダー・ステートとなる。
- 例外が発生した場合には、バックアップの内容をコピーして再開

The diagram illustrates the checkpoint repair process. At the top, a flowchart shows the sequence of events: '命令実行結果' (Command execution result) leads to 'アーキテクチャ・ステート (現論理空間)' (Architecture state (current logical space)), which then leads to 'バックアップ0' (Backup 0), 'バックアップ1' (Backup 1), and finally 'バックアップ2' (Backup 2). Arrows indicate that each backup is a copy of the state at that point. Below this, a horizontal timeline labeled 'Start' and 'End' shows a sequence of operations represented by colored blocks (grey, blue, orange, brown). A vertical line marks a specific point in time, corresponding to the state captured in the backups.

命令実行結果

アーキテクチャ・ステート (現論理空間)

バックアップ0

バックアップ1

バックアップ2

バックアップ・スタック

Start

End

Out-of-order processor

Adapted from Superscalar Microprocessor Design, Mike Johnson

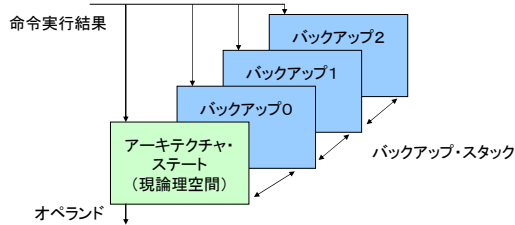
18

Adapted from *Superscalar Microprocessor Design*, Mike Johnson

18

チェックポイント回復 (checkpoint repair)

- 短所
 - ステートのコピー(バックアップの作成)に長い時間を要する。
 - 記憶量が多い。
 - 先見能力に比例してバックアップの数が増大する。

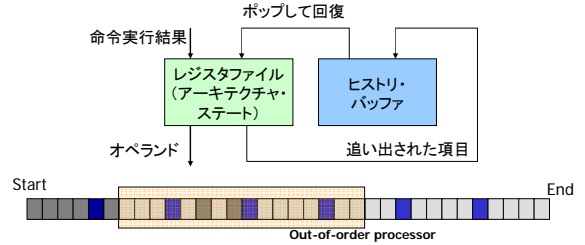


Adapted from Superscalar Microprocessor Design, Mike Johnson

19

ヒストリ・バッファ (history buffer)

- レジスタファイルにはアーキテクチャ・ステートを保存
- スタックの構造を持つヒストリ・バッファ (先見ステートを保持)
 - 命令がデコードされるとプッシュされる。
 - 例外時に、例外命令までポップして回復
 - インオーダー・ステートに含まれる命令はバッファから削除

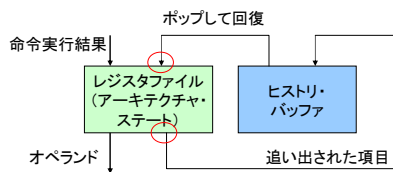


Adapted from Superscalar Microprocessor Design, Mike Johnson

20

ヒストリ・バッファ (history buffer)

- 短所
 - レジスタファイルのポート数の増加(性能向上に寄与しない)
 - 回復するための時間が長い、先見能力の向上に伴ってサイクル数が増加する。

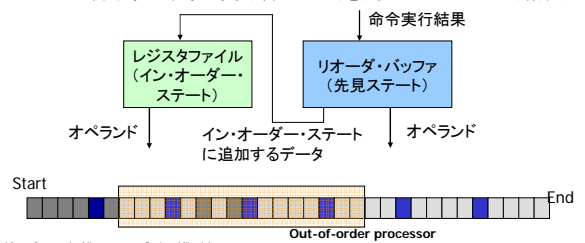


Adapted from Superscalar Microprocessor Design, Mike Johnson

21

リオーダー・バッファ (reorder buffer, ROB)

- イン・オーダー・ステートと先見ステートを結合して、アーキテクチャ・ステートを得る。
- FIFOキュー構造をもつリオーダー・バッファ
 - 命令がデコードされるとエンキューする。
 - インオーダー・ステートに含まれる命令をデキューする。
 - 例外時には、その命令以降のエントリをリオーダー・バッファから破棄する。

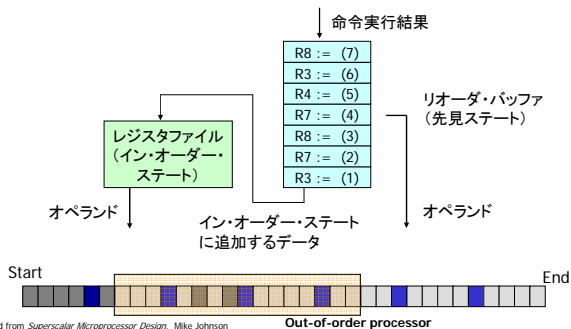


Adapted from Superscalar Microprocessor Design, Mike Johnson

22

リオーダー・バッファ (reorder buffer, ROB)

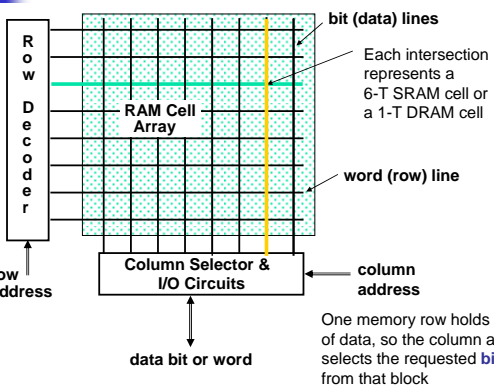
- リオーダー・バッファには、優先度付きの連想検索の機能が必要



Adapted from Superscalar Microprocessor Design, Mike Johnson

23

Classical RAM (random access memory) Organization

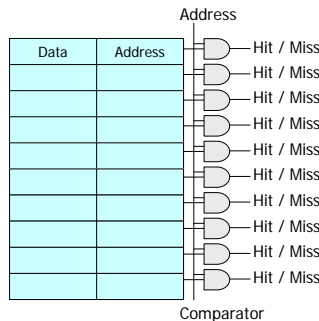


Adapted from Superscalar Microprocessor Design, Mike Johnson

24

連想メモリ, CAM (content addressable memory)

- アドレスを入力として、連想メモリは全内容からそのアドレスと一致するもの検索して、対応するデータを出力する。

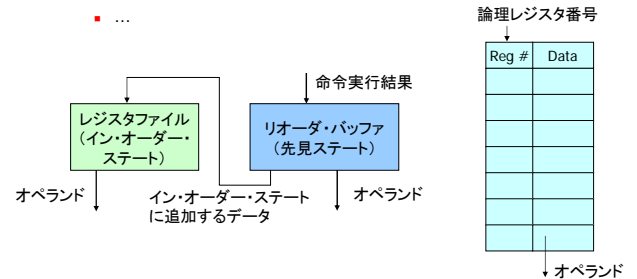


Adapted from Superscalar Microprocessor Design, Mike Johnson

25

リオーダー・バッファ (reorder buffer, ROB)

- 短所
 - 連想検索の機能が必要となる。エントリ数の増大が困難。
- 長所
 - ...

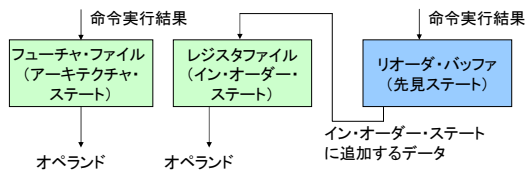


Adapted from Superscalar Microprocessor Design, Mike Johnson

26

フューチャ・ファイル (future file)

- リオーダーバッファはオペランドを供給しない。
- レジスタファイルと同様の構成のフューチャ・ファイルを追加する。
 - フューチャ・ファイルにはアーキテクチャ・ステートを保存
- リオーダーバッファの連想検索を排除できる。
- 例外の場合には、その時点までのインオーダー・ステートを構築して、フューチャ・ファイルの内容を無効化する。



Adapted from Superscalar Microprocessor Design, Mike Johnson

27

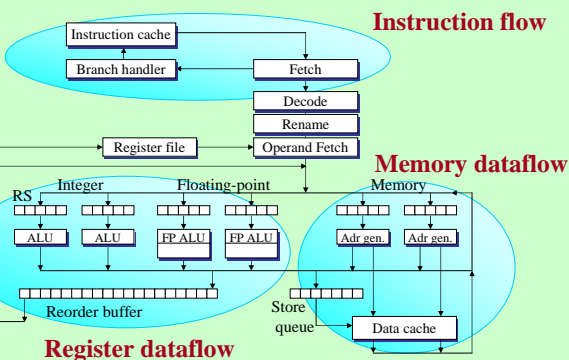
例外回復のためのバッファリング方法 まとめ

- チェックポイント回復
 - アーキテクチャ・ステートをバックアップ
- ヒストリ・バッファ
 - ヒストリバッファに先見ステートの履歴を格納
 - レジスタファイルにアーキテクチャ・ステートを格納
- リオーダー・バッファ
 - レジスタファイルにイン・オーダー・ステートを格納
 - リオーダー・バッファに先見ステートを格納
- フューチャ・ファイル
 - レジスタファイルにイン・オーダー・ステートを格納
 - リオーダー・バッファに先見ステートを格納
 - フューチャ・ファイルにアーキテクチャ・ステートを格納

Adapted from Superscalar Microprocessor Design, Mike Johnson

28

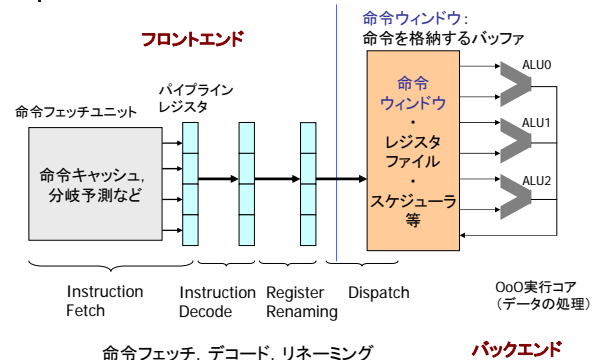
Out-of-order スーパー スカラ プロセッサ



Adapted from Computer Organization and Design, Patterson & Hennessy, © 2005

29

アウトオブオーダー実行プロセッサの構成

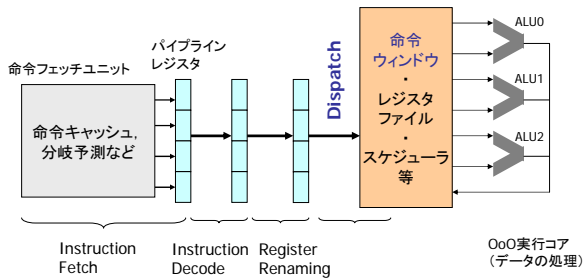


Adapted from Computer Organization and Design, Patterson & Hennessy, © 2005

30

アウトオブオーダー実行プロセッサの構成

- ディスパッチ (dispatch) : 命令ウィンドウに命令を格納する動作

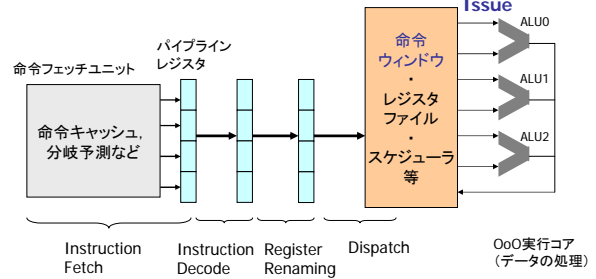


Adapted from Computer Organization and Design, Patterson & Hennessy, © 2005

31

アウトオブオーダー実行プロセッサの構成

- 発行 (issue, fire) : 命令ウィンドウから、データ依存が解消された命令を機能ユニットに送り出す動作
 - この時に、レジスタファイルの値を読み出すことがある。

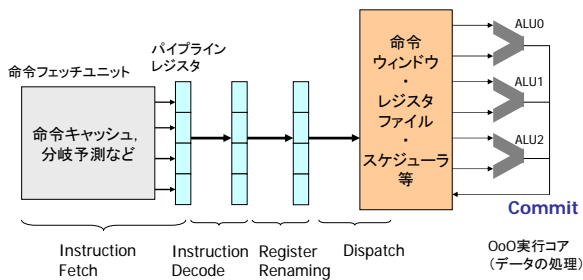


Adapted from Computer Organization and Design, Patterson & Hennessy, © 2005

32

アウトオブオーダー実行プロセッサの構成

- コミット (commit) : 命令の実行結果によってアーキテクチャ・ステートを更新する動作 (リオーダーバッファに値を書き込む)

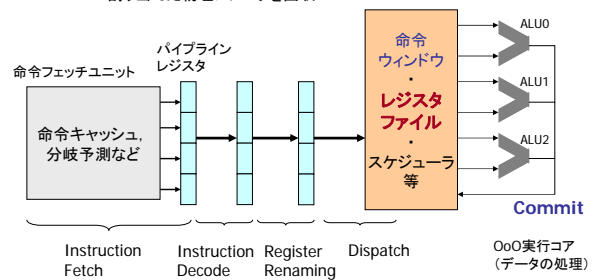


Adapted from Computer Organization and Design, Patterson & Hennessy, © 2005

33

アウトオブオーダー実行プロセッサの構成

- リタイア (retire) : 命令の実行結果によってインオーダー・ステートを更新する動作
 - 割り当てた物理レジスタを回収



Adapted from Computer Organization and Design, Patterson & Hennessy, © 2005

34

レポート(1) : 分岐予測の実装と評価

詳細は12月9日のスライド

- gshare分岐予測を実装し、その予測ミス率を測定せよ。また、bimodal分岐予測との予測精度 (20本のベンチマークのミス率の算術平均) の比較を示せ。
 - ハードウェア量を 2KB, 4KB, 8KB, 16KB, 32KB, 64KBとしてグラフを描け。
- gshare分岐予測に工夫を施し (あるいは、異なる方式の予測を実装し)、予測ミス率を測定せよ。
 - ハードウェア量を 2KB, 4KB, 8KB, 16KB, 32KB, 64KBとしてグラフを描け。
 - 予測ミス率が低い (性能が高い) と高得点。
- 1月6日の講義の開始時にレポートを提出
 - コードの説明 (コードは少ないほどベター)、工夫した点
 - ハードウェア量の計算方法を明示
 - ミス率のグラフ (表ではないので注意)
 - 考察と感想

Adapted from Computer Organization and Design, Patterson & Hennessy, © 2005

35

アナウンス

- 講義スライド、講義スケジュール
 - www.arch.cs.titech.ac.jp

Adapted from Computer Organization and Design, Patterson & Hennessy, © 2005

36