

計算機アーキテクチャ 第二 (O)

5. パイプライン処理

大学院情報理工学専攻 計算工学専攻
吉瀬謙二 kise_at_cs.titech.ac.jp
S321講義室 月曜日 5, 6時限 13:20-14:50

1

ハザード (hazard)

命令を適切なサイクルで実行できないような状況が存在する。これをハザードと呼ぶ。

- **構造ハザード (structural hazard)**
 - オーバラップ実行する命令の組み合わせをハードウェアがサポートしていない場合。
 - 資源不足により生じる。
- **データハザード (data hazard)**
 - データの受け渡しの制約によって生じるハザード
- **制御ハザード (control hazard)**
 - 分岐命令、ジャンプ命令によって生じるハザード

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

2

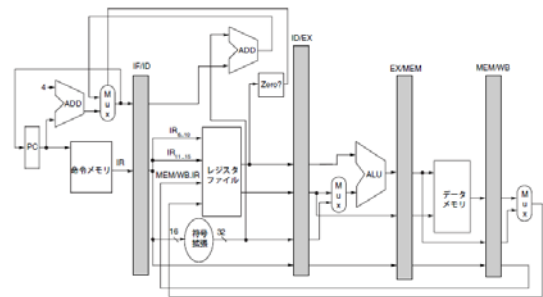
MIPSの基本的な5つのステップ(ステージ)

- **IFステージ**
メモリから命令をフェッチする。
- **IDステージ**
命令をデコードしながら、レジスタを読み出す。分岐命令である可能性を考慮し、読み出されたレジスタの間で一致比較を行う。必要であれば、命令のオフセットフィールドを符号拡張し、インクリメントされたPCに符号拡張されたオフセットを足し合わせて分岐先のアドレスを計算する。条件が成立した場合には分岐先アドレスをPCにセットして、このステージで分岐命令は完了する。

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

3

プロセッサのデータパス(パイプライン処理)



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

4

静的に採用できる制御ハザードの対処 (演習)

- **戦略1**
 - 分岐方向が判明するまで分岐命令の後続命令を止める。
 - IDステージで分岐命令が完了することに注意。

| 分岐命令 | IF | ID | EX | MEM | WB |
|-----------|----|----|----|-----|----|
| 分岐先命令 | | | | | |
| 分岐先命令 + 1 | | | | | |
| 分岐先命令 + 2 | | | | | |

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

5

静的に採用できる制御ハザードの対処

- **戦略1**
 - 分岐方向が判明するまで分岐命令の後続命令を止める。
 - IDステージで分岐命令が完了することに注意。
 - 分岐命令の出現毎に1サイクルのストールが発生する。

| 分岐命令 | IF | ID | EX | MEM | WB | | |
|-----------|----|----|----|-----|----|--|--|
| 分岐先命令 | | | | | | | |
| 分岐先命令 + 1 | | | | | | | |
| 分岐先命令 + 2 | | | | | | | |

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

6

戦略2: predicted-not-taken方式 (Exercise)

- すべての分岐命令を not taken (不成立)として処理を進める。

| Untaken分岐命令 | IF | ID | EX | MEM | WB |
|-------------|----|----|----|-----|----|
| 命令 $i+1$ | | | | | |
| 命令 $i+2$ | | | | | |
| 命令 $i+3$ | | | | | |
| 命令 $i+4$ | | | | | |
| Taken分岐命令 | IF | ID | EX | MEM | WB |
| 命令 $i+1$ | | | | | |
| 分岐先 | | | | | |
| 分岐先 + 1 | | | | | |
| 分岐先 + 2 | | | | | |

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

7

戦略2: predicted-not-taken方式

- すべての分岐命令を not taken (不成立)として処理を進める。
 - 分岐結果が不成立であれば、ペナルティは生じない。
 - 分岐結果が成立であれば、1サイクルのペナルティ

| Untaken分岐命令 | IF | ID | EX | MEM | WB |
|-------------|----|----|------|------|------|
| 命令 $i+1$ | | IF | ID | EX | MEM |
| 命令 $i+2$ | | ID | EX | MEM | WB |
| 命令 $i+3$ | | IF | ID | EX | MEM |
| 命令 $i+4$ | | IF | ID | EX | MEM |
| Taken分岐命令 | IF | ID | EX | MEM | WB |
| 命令 $i+1$ | | IF | idle | idle | idle |
| 分岐先 | | IF | ID | EX | MEM |
| 分岐先 + 1 | | IF | ID | EX | MEM |
| 分岐先 + 2 | | IF | ID | EX | MEM |

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

8

戦略3: predicted-taken方式

- すべての分岐命令を taken (成立)として処理を進める。
- IDステージが終了して、分岐と判定するとすぐに分岐成立として処理を継続。
- 今考えているパイプライン構成では、この方式の利点はない。

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

9

戦略4: 遅延分岐 (delayed branch)

- 分岐命令の後続の幾つかの命令を実行した後に、分岐する。
1サイクルの遅延を持つ命令実行順は次の通り。
 - 分岐命令を実行
 - 分岐命令の次アドレスの命令を実行
 - 分岐成立では、飛び先アドレスの命令を実行(不成立では、分岐命令の次の次のアドレスの命令を実行)

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

10

戦略4: 遅延分岐 (delayed branch)

- 分岐命令の後続の幾つかの命令を実行した後に、分岐する。

| Untaken分岐命令 | IF | ID | EX | MEM | WB |
|--------------|----|----|----|-----|----|
| 分岐遅延命令 $i+1$ | | | | | |
| 命令 $i+2$ | | | | | |
| 命令 $i+3$ | | | | | |
| 命令 $i+4$ | | | | | |
| Taken分岐命令 | IF | ID | EX | MEM | WB |
| 分岐遅延命令 $i+1$ | | | | | |
| 分岐先 | | | | | |
| 分岐先 + 1 | | | | | |
| 分岐先 + 2 | | | | | |

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

11

戦略4: 遅延分岐 (delayed branch)

- 分岐命令の後続の幾つかの命令を実行した後に、分岐する。分岐命令によるストールは生じない。
- 初期のRISCプロセッサにて利用された。

| Untaken分岐命令 | IF | ID | EX | MEM | WB |
|--------------|----|----|----|-----|-----|
| 分岐遅延命令 $i+1$ | | IF | ID | EX | MEM |
| 命令 $i+2$ | | ID | EX | MEM | WB |
| 命令 $i+3$ | | ID | EX | MEM | WB |
| 命令 $i+4$ | | IF | ID | EX | MEM |
| Taken分岐命令 | IF | ID | EX | MEM | WB |
| 分岐遅延命令 $i+1$ | | IF | ID | EX | MEM |
| 分岐先 | | IF | ID | EX | MEM |
| 分岐先 + 1 | | IF | ID | EX | MEM |
| 分岐先 + 2 | | IF | ID | EX | MEM |

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

12

遅延分岐スロットのスケジューリング



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

13

パイプラインの実行の困難さ

- 例外への対処
 - I/O デバイスからの要求
 - ユーザプログラムからのOSサービスの呼び出し
 - 命令実行のトレース生成
 - ブレークポイント(プログラムの要求による割り込み)
 - 整数演算命令のオーバーフロー
 - FP演算命令の不規則さ
 - ページフォールト(メインメモリ内に無い場合)
 - 整列されていないメモリアクセス(整列が必要な場合)
 - メモリ保護違反
 - 未定義あるいは未実装命令の使用
 - ハードウェア異常故障
 - 電源異常
- 命令セットの複雑さ
- 複数サイクル処理の扱い

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

14

パイプラインの実行の困難さ: 例外への対処

| パイプラインステージ 起こり得る問題となる例外 | | | | | | |
|-------------------------|-------------------------------------|--|--|--|--|--|
| IF | 命令フェッチ時ページフォールト、不整列メモリアクセス、メモリ保護違反 | | | | | |
| ID | 未定義・不法オペコード | | | | | |
| EX | 演算例外 | | | | | |
| MEM | データフェッチ時ページフォールト、不整列メモリアクセス、メモリ保護違反 | | | | | |
| WB | 無し | | | | | |

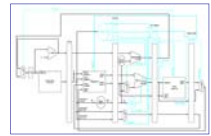
| | | | | | | |
|------|----|----|----|-----|-----|----|
| LD | IF | ID | EX | MEM | WB | |
| DADD | | IF | ID | EX | MEM | WB |

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

15

パイプラインの実行の困難さ: 例外への対処

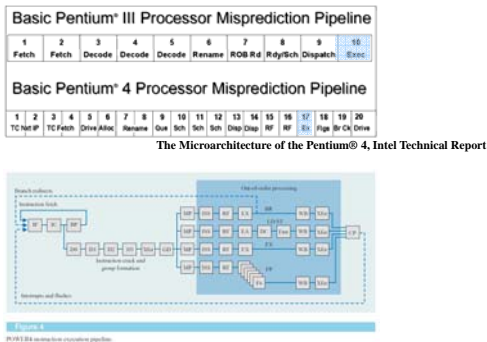
- 次の命令フェッチ時に、トラップ命令をパイプラインに挿入
- トラップ命令が実行されるまで、フォールトした命令とパイプライン中でそれに後続している命令による書き込みをすべて取りやめる。例外を生じた命令からトラップ命令直前のパイプライン中の命令に対して、パイプラインラッチにゼロを書き込むことで実現する。
- OSの例外ハンドラのルーチンが制御を獲得したあとで、そのルーチンはフォールトした命令のPCを直ちに保存する。この値は、後ほど例外から戻る時に使用。



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

16

プロセッサの命令パイプラインの例

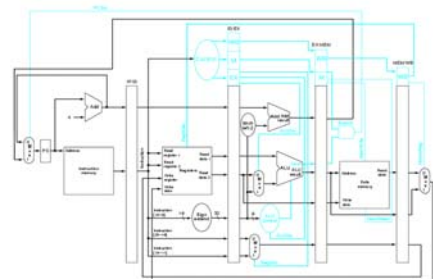


Adapted from Computer Organization and Design, Patterson & Hennessy, © 2005

17

パイプラインの段数

- パイプラインの段数はどこまで増やすことができる？



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

18

Increasing Processor Performance by Implementing Deeper Pipelines

Eric Sprangle, Doug Carmean
Pentium Processor Architecture Group,
Intel Corporation
ISCA-2002 pp.25-34

背景と目的

- プロセッサの動作周波数の決定はプロセッサ設計者の直面する本質的な課題となっている。
- パイプラインが深くなると、設計の複雑さと工程は劇的に増加する。
- パイプラインの深さとキャッシュサイズの関数として、プロセッサ性能を予測するモデルを構築し、シミュレーションにより性能を評価する。
- Pentium 4プロセッサをベースラインとして、深いパイプラインが性能向上につながることを示す。

Simulated 2GHz Pentium 4 like processor config.

Core
3-wide fetch/retire
2 ALUs (running at 2x frequency)
1 load and store / cycle
In-order allocation/de-allocation of buffers
512 rob entries, load buffers and store buffers
Memory System
64 kB/8-way I-cache
8 kB/4-way L1 D-cache, 2 cycle latency
256 kB/8-way unified L2 cache, 12 cycle latency
3.2 GB/sec memory system, 165ns average latency
Perfect memory disambiguation
16 kB Gshare branch predictor
Streaming based hardware prefetcher

Skeleton という実行駆動のシミュレータを用いて評価する。

Simulated Benchmark Suites

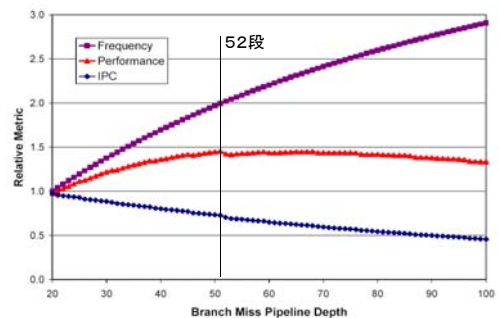
| Suite | Number of Benchmarks | Description |
|--------------|----------------------|---|
| SPECint95 | 8 | spec.org |
| Multimedia | 22 | speech recognition, mpeg, photoshop, ray tracing, rsa |
| Productivity | 13 | sysmark2k internet/business/ productivity, Premiere |
| SPECfp2k | 10 | spec.org |
| SPECint2k | 12 | spec.org |
| Workstation | 14 | CAD, rendering |
| Internet | 12 | webmark2k, specjbb |

パイプラインのオーバーヘッド

- Conservative ASIC design
 - Clock skew + jitter = 51ps
 - Standard 0.18um process, flop overhead is 3 FO4 = 75ps
 - Pipeline overhead = 51ps + 75ps = 125ps
- Custom design
 - Most of clock skew and jitter overhead can be hidden.
 - Pipeline overhead = 75ps
- Extreme custom design
 - Sub-50ps at the cost of a much larger design cost
- Pentium 4 overhead
 - Pipeline overhead = 90ps
 - Use 90ps as a baseline overhead time

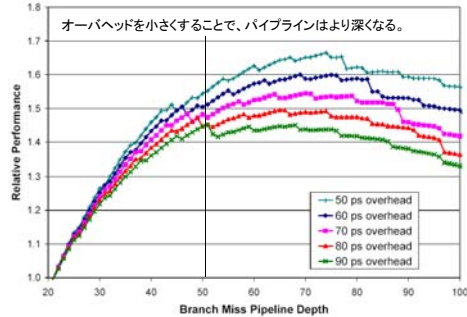
パイプライン段数を変化させた時の動作周波数、IPC、性能の評価結果

- パイプラインが52段で、動作周波数が2倍になるまで性能が向上



パイプライン段数とオーバーヘッドを変化させた時の性能の評価結果

- 評価には、パイプライン段数に影響を受けず一定のオーバーヘッドを想定
- 2GHz のパイプラインピッチ 500ps、Pentium 4のオーバーヘッドは 90ps

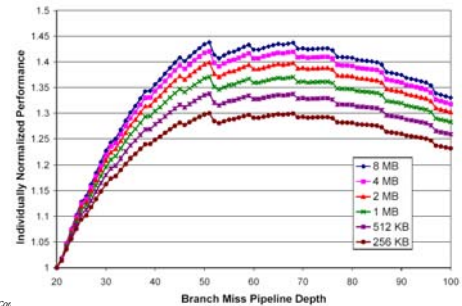


Adapted fr

25

パイプライン段数とキャッシュサイズを変化させた時の性能の評価結果

- キャッシュサイズを変化させることで相対性能は変化するが、最適なパイプライン段数はほとんど変化しない。



Adapted from Cor

26

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

27