2010年 後学期

計算機アーキテクチャ 第二 (O)

6. コンピュータの性能

大学院情報理工学研究科 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

計算機アーキテクチャへの要求

- **速度(実行時間), スループット**
- 消費電力, エネルギー
- 熱
- 音
- 価格
- 安定性, など

Which is faster?


Plane	DC to Paris	Speed	Passengers	Throughput (p x mph)
Boeing 747	6.5 hours	610 mph (1130km/h)	470	286,700 (470 x 610)
BAD/Sud Concorde	3 hours	1350 mph (2500km/h)	132	178,200 (132 x 1350)

- **Time to run the task (ExTime)**
  - **Execution time, response time, latency**
- **Tasks per day, hour, week, sec, ns … (Performance)**
  - **Throughput, bandwidth**

MPH (Mile Per Hour)

From the lecture slide of David E Culler

Defining (Speed) Performance

- Normally interested in reducing
  - Response time (execution time) – the time between the start and the completion of a task
    - Important to individual users
  - Thus, to maximize performance, need to minimize execution time

$$\text{performance}_X = 1 / \text{execution\_time}_X$$

If X is n times faster than Y, then

$$\frac{\text{performance}_X}{\text{performance}_Y} = \frac{\text{execution\_time}_Y}{\text{execution\_time}_X} = n$$

  - Throughput – the total amount of work done in a given time
    - Important to data center managers
  - Decreasing response time almost always improves throughput

Performance Factors

- Want to distinguish elapsed time and the time spent on our task
- CPU execution time (CPU time) : time the CPU spends working on a task
  - Does not include time waiting for I/O or running other programs

$$\text{CPU execution time for a program} = \text{\# CPU clock cycles for a program} \times \text{clock cycle time}$$

or

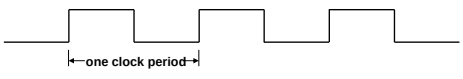
$$\text{CPU execution time for a program} = \frac{\text{\# CPU clock cycles for a program}}{\text{clock rate}}$$

- Can improve performance by reducing either the length of the clock cycle or the number of clock cycles required for a program

Remind: Machine Clock Rate

- Clock rate (MHz, GHz) is inverse of clock cycle time (clock period)

$$\text{Clock rate} = 1 / \text{Clock period}$$



10 nsec clock cycle => 100 MHz clock rate
5 nsec clock cycle => 200 MHz clock rate
2 nsec clock cycle => 500 MHz clock rate
1 nsec clock cycle => 1 GHz clock rate
500 psec clock cycle => 2 GHz clock rate
250 psec clock cycle => 4 GHz clock rate
200 psec clock cycle => 5 GHz clock rate

MIPS (Million Instructions Per Second)

- 1秒当たりに実行された命令の数（単位はMillion）
  - 原始MIPS (native MIPS)
- 注意
  - プロセッサアーキテクチャのMIPSとは関係ない
- MIPSの問題点とは？
  - 命令セットに強く依存する尺度
  - 異なる命令セット，NOP，コンパイラ，性能？

MFLOPS, GFLOPS

- MFLOPS (Million Floating-point Operations Per Second)
- GFLOPS (Giga Floating-point Operations Per Second)
- MIPSとGFLOPSとの相違は？
  - 命令セット，浮動小数点演算

先端マイクロプロセッサ **Cell Broadband Engine**

- ヘテロジニアス チップマルチプロセッサ
  - PowerPC Processor Element (PPE) 1個
  - Synergistic Processor Element (SPE) 8個

PlayStation3 の写真は
PlaySation.com (Japan) から

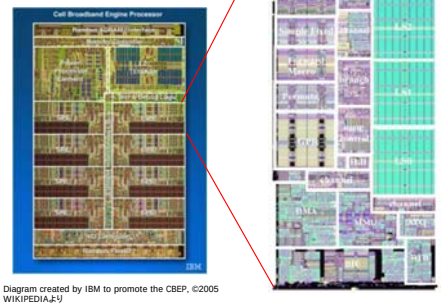


Diagram created by IBM to promote the CBEP, ©2005
WIKIPEDIAより

Cell/B.E. Element Interconnect Bus

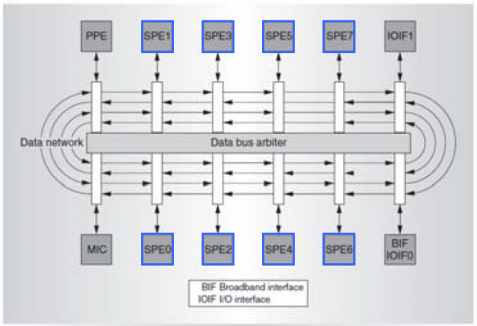


Figure 2. Element interconnect bus (EIB).

IEEE Micro, Cell Multiprocessor Communication Network: Built for Speed

Cell Broadband Engine (GFLOPSの例)

- ピーク性能
  - 1サイクルで積和演算を1回実行できる演算器
    (**2** FLOP/cycle)
  - 1命令で4つの演算を並列処理(SIMD, Single Instruction Multiple Data構成)で，SPEあたりの並列性 **4**
  - チップ内のSPEの数 **8**
  - 動作周波数 **4**GHz
  - **2** × **4** × **8** × **4** ＝ 256 GFLOPS
  - 積和演算 × SIMD化 × マルチコア × 動作周波数
    - **ペンティアムは 8GFLOPS 程度** (256/8 = 32)
  - 性能を引き出す鍵は
    - DMA転送とローカルストアの使い方，SIMD化，並列化 . . .

相対性能，ベンチマーク

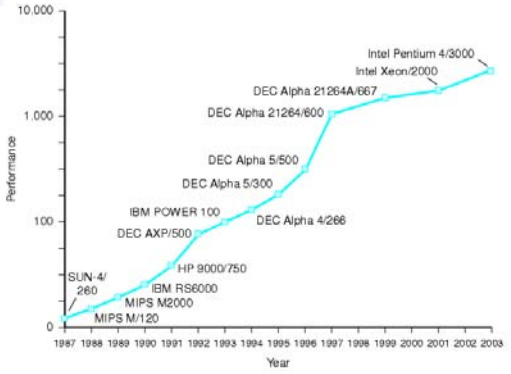
- 合成ベンチマーク (Synthetic Benchmark)
  - Whetstone
  - **Dhrystone**
  - Livermore loops
  - SPEC CPUベンチマーク
    - **SPEC89, SPEC92, SPEC95, SPEC2000, SPEC2006**
  - Intelの新しいベンチマーク
    - Recognition, Mining, and Synthesis
  - 行列積，LU分解
  - 画像処理 (FPS, frame per second)
- 合成ベンチマークの問題点は？
  - 実アプリ，チート，誰がどうやって？

SPEC CINT2000 Summary example

164.gzip	1400	565	248*
175.vpr	1400	655	214*
176.gcc	1100	376	292*
181.mcf	1800	1193	151*
186.crafty	1000	236	424*
197.parser	1800	1023	176*
252.eon	1300	357	365*
253.perlbmk	1800	685	263*
254.gap	1100	525	210*
255.vortex	1900	758	251*
256.bzip2	1500	534	281*
300.twolf	3000	1285	234*
SPECint_base2000			249
SPECint2000			

13

ワークステーションの性能の改善



SPEC整数ベンチマークに基づいて計算

14

落とし穴

- コンピュータのある面を改善することによって，その改善度に等しい性能向上を期待すること．
- Amdahl's Law states that the performance improvement to be gained from using some faster mode of execution is limited by the fraction of the time the faster mode can be used.

15

落とし穴

- 性能の尺度に性能方程式の一部を使用すること．

CPU execution time for a program = # CPU clock cycles for a program x clock cycle time

16

補足:コンピュータの性能

- 性能を議論するためには尺度が必要
  - 尺度の達成が目的化することは危険．
  - テクノロジが進化するとき，尺度も大きく変化する．
  - 時に，適切な尺度を定義することが重要．
  - これにより，飛躍的に進歩することがある．

17

2010年 後学期

計算機アーキテクチャ 第二 (O)

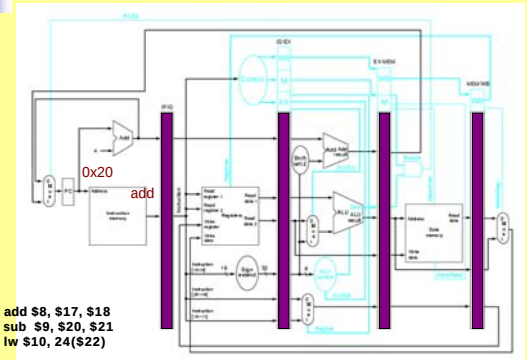
パイプライン処理とパイプライン段数

大学院情報理工学研究科 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

18

プロセッサのデータパス（パイプライン処理）

Clock 1:



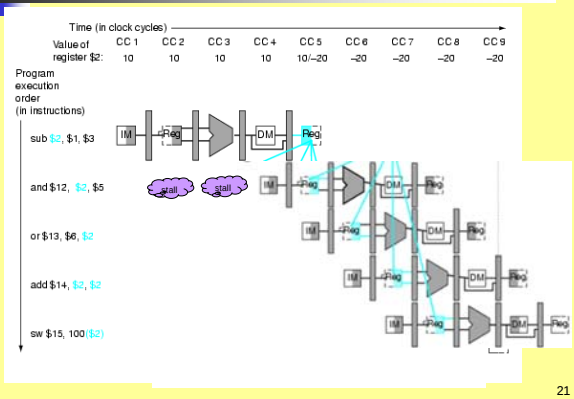
(1) 0x20: add $8, $17, $18
(2) 0x24: sub $9, $20, $21
(3) 0x28: lw $10, 24($22)

ハザード (hazard)

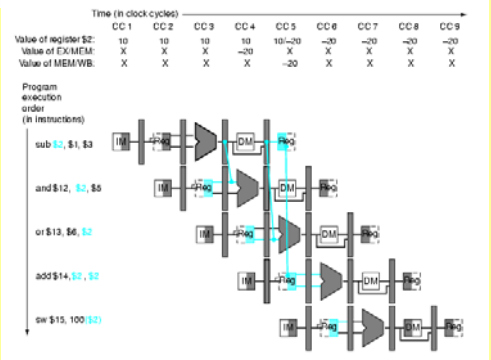
命令を適切なサイクルで実行できないような状況が存在する．これをハザードと呼ぶ．

- 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

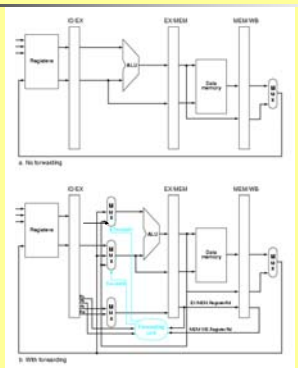
データハザード (ストール)



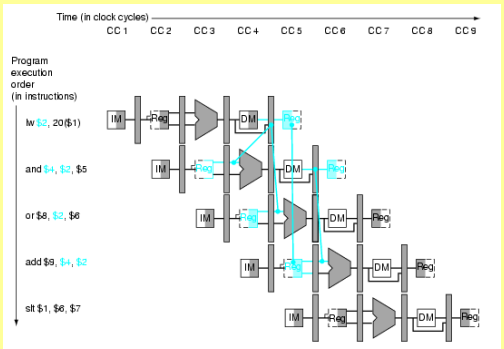
フォワーディングによるデータハザードの回避



フォワーディングのための変更点



データハザードによる生じるストール



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

戦略4：遅延分岐 (delayed branch)

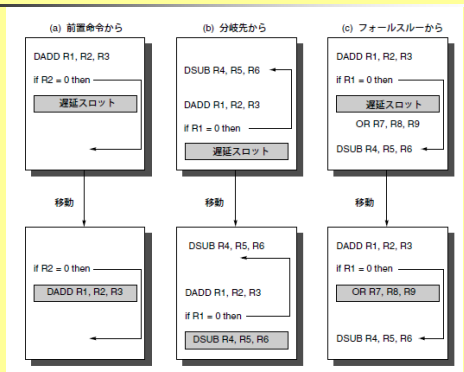
- 分岐命令の後続の幾つかの命令を実行した後に，分岐する．
  1サイクルの遅延を持つ命令実行順は次の通り．
  - 分岐命令を実行
  - 分岐命令の次アドレスの命令を実行
  - 分岐成立では，飛び先アドレスの命令を実行（不成立では，分岐命令の次の次のアドレスの命令を実行）

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	WB		
命令 *i* + 2			IF	ID	EX	MEM	WB	
命令 *i* + 3				IF	ID	EX	MEM	WB
命令 *i* + 4					IF	ID	EX	MEM WB
Taken 分岐命令	IF	ID	EX	MEM	WB			
分岐遅延命令 (*i* + 1)		IF	ID	EX	MEM	WB		
分岐先			IF	ID	EX	MEM	WB	
分岐先 + 1				IF	ID	EX	MEM	WB
分岐先 + 2					IF	ID	EX	MEM WB

遅延分岐スロットのスケジューリング



Nop命令

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

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（プログラマ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

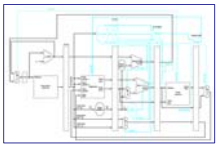
パイプラインの実行の困難さ:例外への対処

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

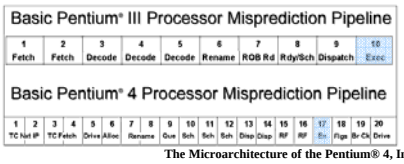
LD	IF	ID	EX	MEM	WB	
DADD		IF	ID	EX	MEM	WB

パイプラインの実行の困難さ:例外への対処

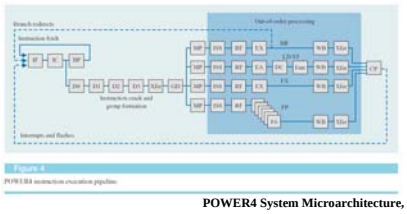
1. 次の命令フェッチ時に，トラップ命令をパイプラインに挿入
2. トラップ命令が実行されるまで，フォールトした命令とパイプライン中でそれに後続している命令による書き込みをすべて取りやめる．例外を生じた命令からトラップ命令直前のパイプライン中の命令に対して，パイプラインラッチにゼロを書き込むことで実現する．
3. OSの例外ハンドラのルーチンが制御を獲得したあとで，そのルーチンはフォールトした命令のPCを直ちに保存する．
   この値は，後ほど例外から戻る時に使用．



プロセッサの命令パイプラインの例



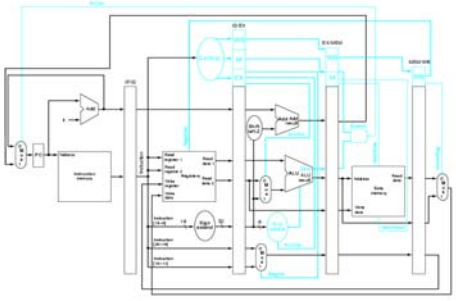
The Microarchitecture of the Pentium® 4, Intel Technical Report



POWER4 System Microarchitecture, IBM Journal

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？



Session 1 – Processor Pipelines

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

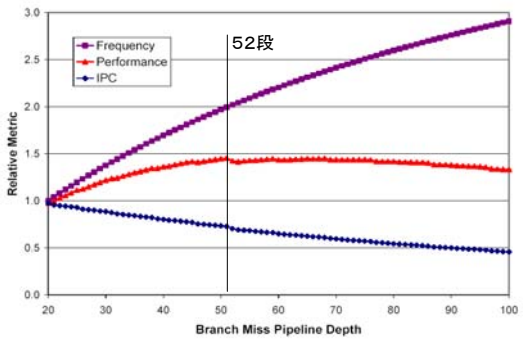
Suite	Number of Benchmarks	Description
SPECint95	8	spec.org
Multimedia	22	speech recognition, mpeg, photoshop, ray tracing, rsa
Productivity	13	sysmark2k internet/business/ productivity, Premiere
SPECfp2k	10	spec.org
SPECint2k	12	spec.org
Workstation	14	CAD, rendering
Internet	12	webmark2k, specjbb

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

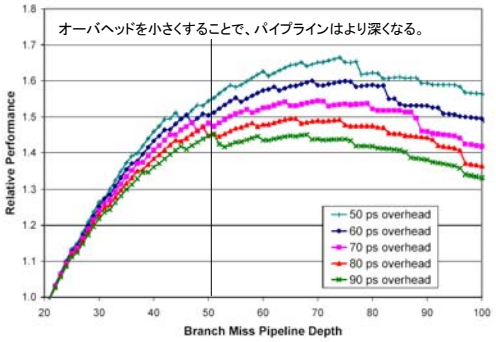
パイプライン段数を変化させた時の動作周波数、IPC、性能の評価結果

- 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



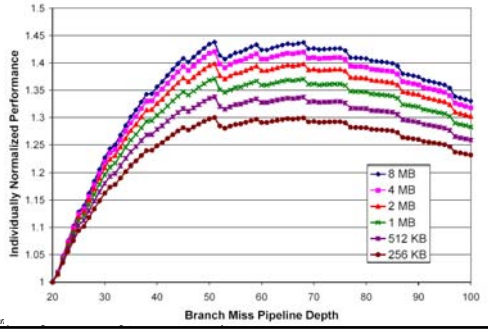
パイプライン段数とオーバヘッドを変化させた時の性能の評価結果

- 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ヘッドを想定
- 2GHz のパイプラインピッチ 500ps、Pentium 4のオーバヘッドは 90ps



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。



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