コンピュータアーキテクチャII チートシート

1. パイプラインとハザード

パイプライン処理は、命令の実行を複数のステージ（IF, ID, EX, MEM, WB）に分割し、並列実行することでスループットを向上させる技術。

重要公式

- **パイプラインのクロックサイクル時間**:
  パイプライン処理のクロックサイクル時間は、**最も処理時間が長いステージ**によって決定される。
  クロック周波数 $= \frac{1}{\text{最長ステージの処理時間}}$
- **パイプラインによるCPI (Cycles Per Instruction)**
  - n個の命令を5ステージパイプラインで実行する場合の理想的なCPIは、$CPI = \frac{n+4}{n} = 1 + \frac{4}{n}$ となる。命令数が十分に大きい場合 ($n \to \infty$)、CPIは1に近づく。
- **パイプラインによる性能向上率**:
  向上率 $= \frac{\text{パイプラインなしの実行時間}}{\text{パイプラインありの実行時間}}$
- **性能指標 MIPS (Million Instructions Per Second)**
  - $MIPS = \frac{\text{クロック周波数 (MHz)}}{CPI}$
  - $MIPS = \text{クロック周波数 (MHz)} \times IPC$ (※IPCはCPIの逆数)

基本用語

- **パイプラインステージ (Pipeline Stages)**
  - **IF (Instruction Fetch):** 命令フェッチ
  - **ID (Instruction Decode):** 命令デコードとレジスタ読み出し
  - **EX (Execute):** 演算実行
  - **MEM (Memory Access):** メモリアクセス
  - **WB (Write Back):** 結果のレジスタ書き込み
- **ハザード (Hazard):** パイプラインの流れを乱し、性能を低下させる要因。
  - **構造ハザード (Structural Hazard):** ハードウェア資源の競合。例えば、命令フェッチ（IF）とデータアクセス（MEM）が同時に単一ポートのメモリにアクセスしようとする場合。
    - **解決策:** 命令キャッシュとデータキャッシュを分離する。
  - **データハザード (Data Hazard):** 後続の命令が、まだパイプライン内にある先行命令の実行結果を必要とする場合に発生する。
    - **解決策: フォワーディング（バイパシング）** で演算結果を後続ステージに直接渡す、または**ストール（バブル）** を挿入して待機する。
  - **制御ハザード (Control Hazard):** 分岐命令の実行結果（分岐するか否か）が確定するまで、次にどの命令をフェッチすればよいか分からない状態。
    - **解決策: ストール**、**分岐予測**、**遅延分岐**。

- **分岐予測 (Branch Prediction):** 制御ハザードによるペナルティを削減する技術。
    - **分岐ペナルティ:** 分岐予測が外れた場合に発生するストール（バブル）による遅延。分岐先が確定するステージ（例: EXステージ）までにフェッチしてしまった不正な命令の数だけペナルティが発生する。
    - **静的分岐予測:** コンパイル時に分岐方向を固定的に予測する。
        - **常にNot Taken (分岐しない):** ループの分岐で有効。ループの最後の1回以外は予測が成功する。
        - **常にTaken (分岐する):** ループの分岐で有効。ループの最初の1回以外は予測が成功する。
    - **動的分岐予測:** 実行時の履歴に基づいて予測する。**分岐予測バッファ（または分岐履歴テーブル BHT）**に履歴を保持する。
        - **1ビット予測:** 直前の分岐結果をそのまま次の予測とする。ループの最初と最後で2回予測を外す。
        - **2ビット飽和カウンタ:** 4つの状態（Strongly/Weakly Taken/Not Taken）で履歴を管理。2回連続で予測が外れない限り予測を変更しないため、ループの最後など一時的な分岐パターンの変化に強い。
        - **分岐予測ヒット率:** (予測が当たった回数)/(分岐命令の総実行回数)

---

2. 記憶階層（キャッシュと仮想記憶）

プロセッサと主記憶の速度差（フォン・ノイマン・ボトルネック）を埋めるため、高速・小容量のキャッシュメモリを階層的に用いる。

重要公式

- **キャッシュミス発生時の実効CPI**
    - 基本CPIが1の場合、キャッシュミスによるペナルティを考慮したCPIは以下のように計算できる。
    - $CPI_{\text{実効}} = 1 + \text{ミス率} \times \text{ミスペナルティ}$
        - ミスペナルティは、主記憶へのアクセス時間（クロックサイクル数）。
- **キャッシュミスを考慮した実効CPI（詳細）**
  $CPI_{\text{実効}} = CPI_{\text{理想}} + \frac{\text{メモリアクセスによるストールサイクル数}}{\text{総命令数}}$
  メモリストールサイクル数 $= (\text{命令フェッチのミス回数} + \text{データアクセスのミス回数}) \times$ ミスペナルティ
- **平均メモリアクセス時間 (AMAT: Average Memory Access Time):**
  $AMAT = \text{ヒット時間} + \text{ミス率} \times \text{ミスペナルティ}$

主要アルゴリズム

- **LRU (Least Recently Used) ブロック置換アルゴリズム:** キャッシュがいっぱいの時に新しいブロックを入れる際、最も長い間アクセスされていないブロックを追い出す方式。時間的局所性の原理に基づく。

基本用語

- **参照の局所性:**

    - **時間的局所性:** 一度アクセスされたデータは、近い将来再びアクセスされる可能性が高い。（例: ループ内の変数）
    - **空間的局所性:** あるデータがアクセスされた場合、その近傍のデータもアクセスされる可能性が高い。（例: 配列のシーケンシャルアクセス）

- **キャッシュマッピング方式 (Cache Mapping)**
    - **ダイレクトマップ (Direct Mapped):** 主記憶の各ブロックがキャッシュ内の特定の1つのラインにのみ配置される方式。回路は単純だが、コンフリクトミス（スラッシング）が起きやすい。
    - **フルアソシアティブ (Fully Associative):** 主記憶のどのブロックもキャッシュ内のどのラインにでも配置できる方式。ヒット率は高いが、全てのタグを比較する必要があり回路規模が大きく複雑になる。
    - **nウェイ・セットアソシアティブ (n-way Set-Associative):** 上記2つの中間。キャッシュを複数のセットに分割し、各セット内ではn個のブロックを自由に配置できる。

- **書き込み方針 (Write Policy)**
    - **ライトスルー (Write-Through):**
        - **動作:** CPUが書き込みを行う際、キャッシュと主記憶の両方を**同時に**更新する。
        - **特徴:** 構造が単純で、主記憶は常に最新の状態に保たれる。しかし、書き込みのたびに低速な主記憶へのアクセスが発生するため、書き込み性能が低い（ライトストールが発生しやすい）。
    - **ライトバック (Write-Back):**
        - **動作:** 書き込みはまずキャッシュに対してのみ行い、そのブロックがキャッシュから追い出される時に初めて主記憶に書き戻す。ダーティビットで変更の有無を管理する。
        - **特徴:** 主記憶への書き込み回数が減るため高速。ただし、制御が複雑になり、キャッシュと主記憶の内容が一時的に不一致になる。

- **仮想記憶 (Virtual Memory):** 二次記憶装置を利用して、主記憶の物理容量よりも大きなアドレス空間をプロセスに提供する仕組み。

- **ページング (Paging):** 仮想アドレス空間と物理アドレス空間を**ページ**という固定長のブロックに分割して管理する方式。

- **ページテーブル (Page Table):** 仮想ページ番号と物理ページフレーム番号の対応関係を格納するテーブル。

- **TLB (Translation Lookaside Buffer):** ページテーブルのエントリをキャッシュするための高速な連想メモリ。アドレス変換を高速化する。

- **セグメンテーション (Segmentation):** プログラムをコード、データ、スタックといった論理的な**セグメント**に分割して管理する方式。

---

3. 命令レベル並列性 (ILP)

パイプラインを複数用意し、依存関係のない命令を同時に実行することで性能を向上させる技術 。性能指標としてCPI (< 1) の代わりに**IPC (Instructions Per Cycle)**が用いられる 。

基本用語

- **VLIW (Very Long Instruction Word):** コンパイラが静的に並列実行可能な命令群を1つの長い命令にまとめる。
- **スーパースカラ (Superscalar):** ハードウェアが動的に命令間の依存関係を解析し、複数の命令を並列に実行する。**Out-of-Order実行**や**レジスタリネーミング**といった技術が用いられる。
    - **Out-of-Order実行:** 依存関係がなければ、プログラム上の命令の順序とは異なる順序で命令を実行する。
- **データ依存の種類**
    - **フロー依存 (真の依存)**: Read-After-Write (RAW)。先行命令の書き込み結果を後続命令が読み出す。この順序は変更不可 。
    - **逆依存 (偽の依存)**: Write-After-Read (WAR)。先行命令が読み出したレジスタを後続命令が上書きする 。
    - **出力依存 (偽の依存)**: Write-After-Write (WAW)。先行命令と後続命令が同じレジスタに書き込む 。
- **レジスタリネーミング (Register Renaming)**: 偽の依存関係（逆依存、出力依存）を、物理的に多数あるレジスタを割り当てることで動的に解消し、並列性を高める技術 。

---

4. データ並列とマルチプロセッサ

基本用語

- **データ並列性 (Data Parallelism):** 複数のデータに対して同じ処理を並列に実行できる性質。
- **SIMD (Single Instruction, Multiple Data):** 1つの命令で複数のデータを同時に処理するアーキテクチャまたは命令セット（例: MMX, SSE, AVX）。
- **共有メモリ型マルチプロセッサ:** 複数のプロセッサが単一のアドレス空間を共有するシステム。
    - **UMA (Uniform Memory Access):** どのプロセッサからも全てのメモリ領域に同じ時間でアクセスできる。
    - **NUMA (Non-Uniform Memory Access):** プロセッサとメモリの物理的な位置関係によってメモリアクセス時間が異なる。
- **キャッシュコヒーレンシ (Cache Coherency):** マルチプロセッサシステムにおいて、各プロセッサが持つキャッシュの内容と主記憶の内容との整合性を保つこと。
- **スヌーピング (Snooping):** 各キャッシュが共有バスを監視（スヌープ）し、他のプロセッサのメモリアクセスを検知して自身のキャッシュの状態を更新（または無効化）する手法。
- **ディレクトリ方式 (Directory-based Protocol):** 大規模な分散共有メモリシステムで用いられる。各メモリブロックの共有状態（どのプロセッサがコピーを持っているか）をディレクトリで一元管理する。