11 高速制御方式

- ♣ ベクトル処理, ベクトル型スーパコンピュータ, SIMD 演算
- ♣ スーパースケーラ, VLIW
- ♣ マルチプロセッサ, マルチコア

11.1 ベクトル処理

- スカラー (scalar) 命令 … 1 つの命令で 1 つの演算を行う (普通の命令/演算)

```
add $3, $1, $2
```

	\$1	11
+)	\$2	21
	\$3	32

- ベクトル (vector) 命令
SIMD (single instruction multiple data) 命令

  1 つの命令で 複数 のデータに対して 同じ 演算を行う

```
vad 3, 1, 2
```

	VR1	11	53	42	31	23	41	73	14
+)	VR2	21	15	44	15	71	32	15	53
	VR3	32	68	86	46	94	73	88	67

- ベクトル/SIMD 命令の高速実行
  - 並列 処理 (複数の演算器で実行)
  - パイプライン 処理 (流れ作業で処理)
- ベクトル型スーパコンピュータ
  - 特に 浮動小数点 数のベクトル演算を超高速に実行
    - ∗ 気象予報, 流体計算, 構造計算, etc.
- PC やゲーム機

  マルチメディア (画像や音声) データの処理を高速化

  例) CELL (PS3 に搭載)
    - ∗ 128 bit レジスタを 128 本搭載
    - ∗ 単精度 (32 bit) 浮動小数点演算× 4 , 倍精度 (64 bit) 浮動小数点× 2 を 1 命令で実行
    - ∗ 16bit 整数演算× 8 , 32bit 整数演算× 4 を 1 命令で実行

  例) x86 の SSE (streaming SIMD extension)
    - ∗ x86 に 128 bit レジスタ× 8 本を追加
    - ∗ 単精度浮動小数点演算 (SSE)
    - ∗ 倍精度浮動小数点演算, 整数演算 (SSE2 以降)

アムダールの法則 (Amdahl's Law)

プログラム中の並列処理可能な部分の割合が α ならば, 並列化による高速化の上限は $\frac{1}{1-\alpha}$ 倍である, という法則. 例えば, プログラムのうち並列処理可能な部分が 1/2 しかなければ, それを並列処理によって無限倍に高速化できるコンピュータがあっても, プログラム全体としては [2] 倍しか早くならない, というちょっと考えれば当たり前の話. 計算を速くするためには, 並列処理の高速化だけでなく, 1) 並列処理可能な部分を [増やす] こと, 2) 並列処理 [できない] 部分を高速化すること, を忘れてはならない, という文脈でよく引用される.

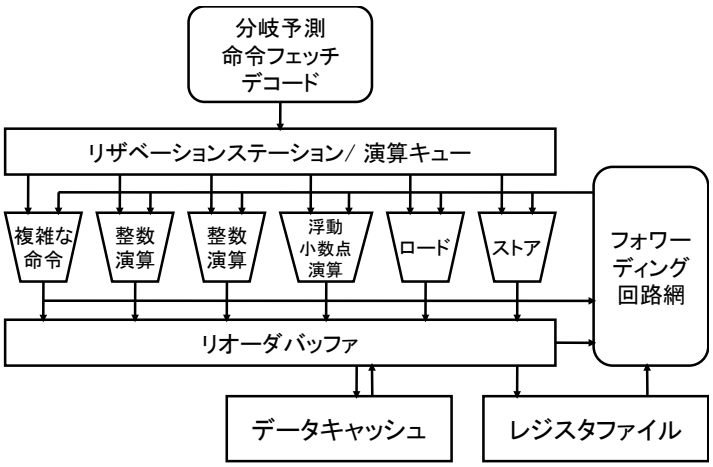
11.2 命令レベル並列処理

11.2.1 スーパースケーラ

- スーパースケーラ ([superscalar]) アーキテクチャ
  - ☆ ベクトル処理により高速化できない部分 (scalar 処理) を高速化するので superscalar と言う
  - – 複数の演算器を持ち, 複数の (異なる種類の) 命令を [並列] に実行
    - ☆ 1 命令ずつ逐次的に実行した場合と同じ結果を [保証]
    - ☆ 命令の並列化 ([スケジューリング]) は [ハードウェア] が動的に行う
      - ∗ 一度に [複数] 命令をフェッチ
      - ∗ フェッチした命令間のデータ依存関係を [解析] (ハードウェアが行う)
      - ∗ データ依存関係がなく [演算器] が空いていれば命令の実行を開始
  - – スーパースケーラプロセッサの CPI と [IPC]
    - ∗ スーパースケーラでも (当然) 命令のパイプライン処理を行う
      → 1 クロックで複数の命令を実行できる → 平均 CPI は 1 [未満]
    - ∗ そこで, CPI の逆数 IPC を用いることが多い
      IPC = [instruction per clock] ; 1 クロックで実行できる平均命令数
    - ☆ 一般的なスーパースケーラの並列度は [2〜8] 程度
- 命令のインオーダー実行とアウトオブオーダ実行
  - – インオーダー ([in-order]) 実行
    - ∗ プログラムに書かれた順序 [どおり] に命令を実行する
      → 実行時間が長い命令の後の命令が待たされる
  - – アウトオブオーダー ([out-of-order]) 実行
    - ∗ 命令の実行の [追い越し] を許す
      (しかし, 計算結果は in-onder 実行と異ならないようにしなければならない)

- 大まかな処理の流れ
  - フェッチした命令は解読され, 各演算器の前にあるバッファで待つ

    バッファは, リザベーションステーション (reservation station), あるいは演算キュー (queue) と呼ばれる
  - データ依存 関係のある先行命令が全て実行されてオペランドがそろい, 演算器が 空け ば, 命令は実行される
  - アウトオブオーダ実行でレジスタへの書き込み順に矛盾が起こらないよう, 実行結果は一旦 リオーダーバッファ に書き込まれる
  - 演算器やリオーダバッファからは フォワーディング (バイパス) 回路網が張り巡らされる
  - 命令の実行を終了しても良いことが 確定 すると, リオーダバッファからレジスタファイルやデータキャッシュへの書き込みが行われる (commit)



- 長短
  - ハードウェアが動的スケジューリングを行う (実行時 の状況に応じて 並列 に実行する命令を決める) ので非常に高速 (非ベクトル型のアーキテクチャとしては現時点で最強)
  - ハードウェアが非常に 複雑 で, 消費電力 が大きい
  - ☆ PC , EWS , サーバー 用のプロセッサで採用される

11.2.2 VLIW

VLIW (Very Long Instruction Word) アーキテクチャ

- 1 つの 長い 命令語の中に 複数 の演算/命令を収容し, これらを 並列 に実行

  (例) TMS32C6000 (Texus Instruments) では 1 命令が 256 bit

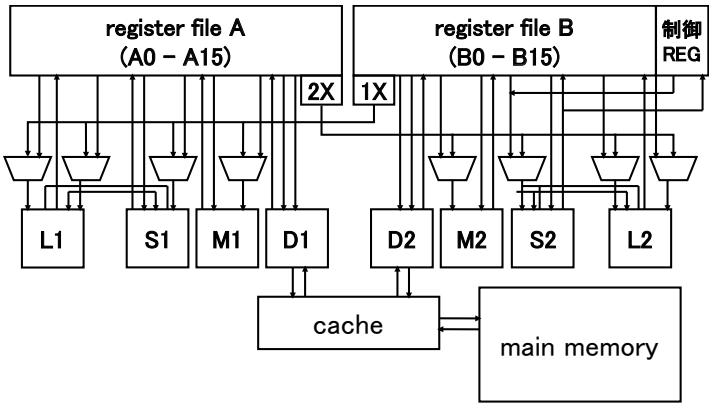
256bit

ADD A6,A7,A7	SUB A1,1,A1	MPY A3,A4,A3	LDW *A2++,A5	ADD B2,B3,B3	NOP	MPY B1,B4,B6	LDW *B2++,B5
L1 slot	S1 slot	M1 slot	D1 slot	L2 slot	S2 slot	M2 slot	D2 slot
32bit	32bit	32bit	32bit	32bit	32bit	32bit	32bit

∗ 256bit = 32 bit× 8 命令を並列に実行

∗ 並列に実行できる演算の 組合せ には制限がある

例) 命令の最初の位置 (L1 slot) にある演算は, 演算器 L1 で実行できるものに限られる



– 命令の並列化は コンパイラ か 人手 で行う (静的 スケジューリング)

- 長短
  - スーパースケーラよりはハードウェアが 単純 で, 消費電力が 小さい
  - 命令の並列化は, プログラムの情報だけから行うものに 限定 される
  - ☆ 携帯 機器に搭載する信号処理プロセッサ (DSP : digital signal processor) 等に搭載される

11.3 マルチプロセッサとマルチコア

- マルチプロセッサ (multi-processor) システム
  - 複数の CPU で構成されるシステムの総称
  - プロセッサ 数 や 結合 方式に多くのバリエーションがある
- マルチコア (multi-core)
  - 1チップ に複数のプロセッサを集積したもの
- 分類

(1) プロセッサの種類による分類

– ホモジニアス (homogeneous) … 同種類 のプロセッサからなる

– ヘテロジニアス (heterogeneous) … 異種類 のプロセッサからなる

(2) プロセッサが処理するタスクによる分類

– SMP (symmetric multi-processing)

∗ 全てのプロセッサに 均一 的に処理が割り付けられる

∗ 1 つの OS が全プロセッサを管理する

– AMP (asymmetric multi-processing)

＊ 特定のプロセッサに 特定 の処理が (あらかじめ) 割り付けられている

- マルチプロセッサを用いる目的
  (1) 高速 化
  - ただし, コア数/プロセッサ数に比例して処理が速くなる とは限らない
  - プログラミング が大変

  (2) リアルタイム応答性 の確保
  - あるプロセッサが外部との インタフェース を担当し,
    他のプロセッサ (群) が バックグラウンド の処理を担当

  (3) 高信頼 化
  - 複数のプロセッサで 同一 の計算をする
    ＊ 誤り の 検出 や 訂正 を行う
    ＊ 2 重系 (dual system), 3 重系 (triple system)
  - 予備 プロセッサとして待機
    ＊ 障害時に 切り替える
    ＊ 2 重系 (duplex system)

  (4) 低消費電力 化

  ☆ 予備知識
    ＊ LSI は動作電圧が高いほど 高速 (動作周波数を 高 くできる)
    ＊ LSI の消費電力は 動作電圧の２乗 に比例
  - 高速なプロセッサの代わりに, 低速なプロセッサ複数で処理
    → 低動作電圧 → 低消費電力

- マルチコアプロセッサの例
  - Intel Core i7 ··· ホモ ジニアス SMP 4 コア
  - Texus Instruments OMAP (携帯電話等に搭載) ··· ヘテロ ジニアス 2 コア (ARM+ DSP)
  - NVIDIA GeForce GTX 1080 (GPU: グラフィックプロセッサ) ··· 2560 コア

Nagisa ISHIURA