2009年 後学期

計算機アーキテクチャ 第二 (O)

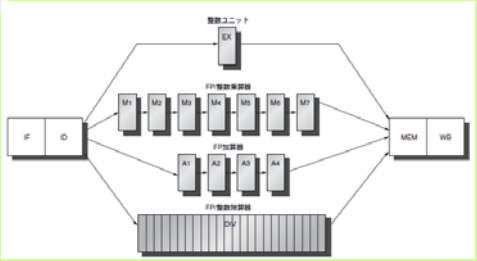
7. スカラプロセッサのパイプライン処理

機能ユニットのレイテンシと発行間隔の例

機能ユニット	レイテンシィ	発行間隔
整数ALU	0	1
データメモリ（整数・FPロード）	1	1
FP加算器	3	1
FP乗算器（整数乗算にも使用）	6	1
FP除算器（整数除算にも使用）	24	25

- レイテンシとは，結果を作る命令とその結果を利用する命令の間のサイクル数
- 発行間隔とは，同じ種類の2命令を発行する間に必用とするサイクル数
- 例えば，FP乗算器は7段にパイプライン化されている．

複数のFP演算をサポートするパイプライン



- FP乗算器，FP加算器をパイプライン化
- 実は，FP除算器はパイプライン化されていない．完了までに24サイクル．
- このパイプラインのタイミングを考える．

ハザード (hazard)

- **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，命令i の後に命令j が実行される場合．
    - RAW (read after write) 命令iが書き込み前に，命令jがそれを読みだそうとする．
    - WAW (write after write) 命令iが書き込む前に，命令jが書き込もうとする．
    - WAR (write after read) 命令iが読む前に，命令jがそこに書こうとする．
- **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

RAWハザードにより生じるストールの例（演習）

命令	クロックサイクル 1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17
L.D F4,0(R2)	IF	ID	EX	MEM	WB												
MUL.D F0,F4,F6																	
ADD.D F2,F0,F8																	
S.D F2,0(R2)																	

- パイプラインは完全にバイパス及びフォワーディングされていると仮定
- IF, ID, MEM, WBステージには1命令しか格納できないことに注意．
- 長い実行パイプラインはストールの頻度が高い．

RAWハザードにより生じるストールの例（演習）

命令	クロックサイクル 1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17
L.D F4,0(R2)	IF	ID	EX	MEM	WB												
MUL.D F0,F4,F6		IF	ID	stall	M1	M2	M3	M4	M5	M6	M7	MEM	WB				
ADD.D F2,F0,F8			IF	stall	ID	stall	stall	stall	stall	stall	stall	A1	A2	A3	A4	MEM	WB
S.D F2,0(R2)					IF	stall	stall	stall	stall	stall	stall	ID	EX	stall	stall	stall	MEM

- パイプラインは完全にバイパス及びフォワーディングされていると仮定
- IF, ID, MEM, WBステージには1命令しか格納できないことに注意．
- 長い実行パイプラインはストールの頻度が高い．

WBステージの競合

命令	クロックサイクル										
	1	2	3	4	5	6	7	8	9	10	11
MUL.D F0,F4,F6	IF	ID	M1	M2	M3	M4	M5	M6	M7	MEM	WB
...		IF	ID	EX	MEM	WB					
...			IF	ID	EX	MEM	WB				
ADD.D F2,F4,F6				IF	ID	A1	A2	A3	A4	MEM	WB
...					IF	ID	EX	MEM	WB		
...						IF	ID	EX	MEM	WB	
L.D F2,0(R2)							IF	ID	EX	MEM	WB

- クロック11において，3つの命令がWBで競合する．
- WBのライトポートを増やすことで解決可能だが，このケースは希なので，構造ハザードとして処理することが一般的．
- IDステージでストールを検出するか，MEMステージに入るときにストールを検出する．
  - MEMステージに入るときには，どの命令をストールさせるかという選択

WAWハザードの可能性

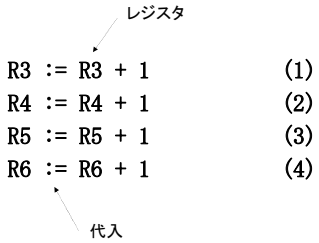
命令	クロックサイクル										
	1	2	3	4	5	6	7	8	9	10	11
MUL.D F0,F4,F6	IF	ID	M1	M2	M3	M4	M5	M6	M7	MEM	WB
...		IF	ID	EX	MEM	WB					
...			IF	ID	EX	MEM	WB				
ADD.D F2,F4,F6				IF	ID	A1	A2	A3	A4	MEM	WB
...					IF	ID	EX	MEM	WB		
...						IF	ID	EX	MEM	WB	
L.D F2,0(R2)							IF	ID	EX	MEM	WB

- もし，L.D命令が1サイクル早く発行されると
- ADD.D より早く F2 に書き込むので WAW ハザードが生じる．
  - ADD.D の結果が利用されることなく上書きされる場合に発生する．
  - ADD.D と L.D の間で利用されると，RAWハザードのためストールする．
- (1) L.D の発行を遅らせる．
- (2) ADD.D の結果を書き込まない．

ハザード (hazard)

- **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，命令i の後に命令j が実行される場合．
    - RAW (read after write) 命令iが書き込み前に，命令jがそれを読みだそうとする．
    - WAW (write after write) 命令iが書き込む前に，命令jが書き込もうとする．
    - WAR (write after read) 命令iが読む前に，命令jがそこに書こうとする．
- **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

データの受け渡しの制約（データ依存）のない命令列



データ依存関係のある命令列

```
R3 := R3 x R5        (1)
R4 := R3 + 1         (2)
R3 := R5 + 1         (3)
R7 := R3 x R4        (4)

If R3=20, R5=3
60 := 20 x 3         (1)
61 := 60 + 1         (2)
4  := 3  + 1         (3)
244:= 4  x 61        (4)
```


真のデータ依存 (true data dependence)

```
R3 := R3 x R5        (1)
R4 := R3 + 1         (2)
R3 := R5 + 1         (3)
R7 := R3 x R4        (4)

If R3=20, R5=3
60 := 20 x 3         (1)
61 := 60 + 1         (2)
4  := 3  + 1         (3)
244:= 60 x 61        (4)
```

3番目の命令が完了する前に，4番目の命令を実行してはいけない．
RAW (read after write)

出力依存 (output dependence)

```
R3 := R3 x R5        (1)
R4 := R3 + 1         (2)
R3 := R5 + 1         (3)
R7 := R3 x R4        (4)

If R3=20, R5=3
60 := 20 x 3         (1)
61 := 60 + 1         (2)
4  := 3  + 1         (3)
XXX:= 60 x 61        (4)
```

1番目の命令の代入が3番目の命令の代入より後に完了してはいけない.
WAW (write after write)

逆依存 (antidependence)

```
R3 := R3 x R5        (1)
R4 := R3 + 1         (2)
R3 := R5 + 1         (3)
R7 := R3 x R4        (4)

If R3=20, R5=3
60 := 20 x 3         (1)
XX := 4  + 1         (2)
4  := 3  + 1         (3)
XXX:= 4  x XX        (4)
```

2番目の命令が実行を始める前に, 3番目の命令を完了してはいけない.
WAR (write after read)

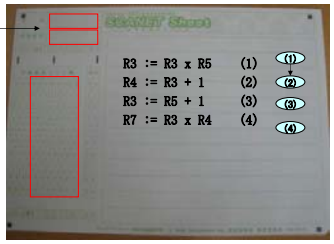
データ依存関係の例（演習）

```
R3 := R3 x R5        (1)
R4 := R3 + 1         (2)
R3 := R5 + 1         (3)
R7 := R3 x R4        (4)
```

Exercise

データ依存関係の例（演習）
(A) 真のデータ依存 (true data dependence)
(B) 出力依存 (output dependence)
(C) 逆依存 (antidependence)

氏名, 学籍番号,
学籍番号マーク欄(**右詰で**)

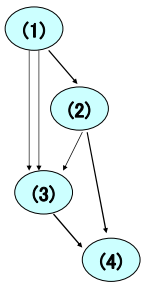


データ依存関係の例

```
R3 := R3 x R5        (1)
R4 := R3 + 1         (2)
R3 := R5 + 1         (3)
R7 := R3 x R4        (4)
```

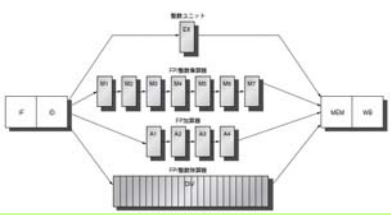

データ依存関係の例

```
R3 := R3 x R5        (1)
R4 := R3 + 1         (2)
R3 := R5 + 1         (3)
R7 := R3 x R4        (4)
```



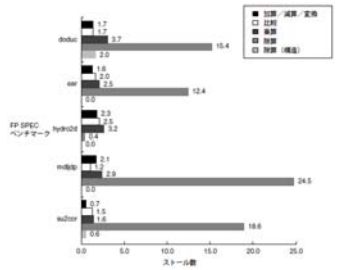
命令レベル並列性は？

複数のFP演算をサポートするパイプラインの性能



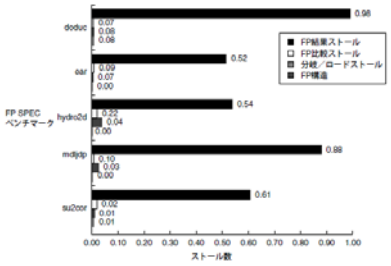
機能ユニット	レイテンシ	発行間隔
整数ALU	0	1
データメモリ（整数・FPロード）	1	1
FP加算器	3	1
FP乗算器（整数乗算にも使用）	6	1
FP除算器（整数除算にも使用）	24	25

各浮動小数点演算のストールサイクル数



- 最初のバーは，FP加算／減算／変換の平均ストール数
- これらは，レイテンシと結果が利用できるまでのサイクル数等に依存

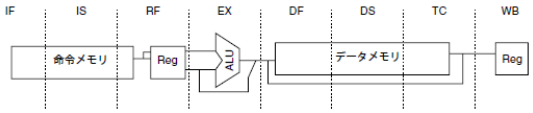
MIPS FPパイプラインの性能



- FP結果ストールが支配的
- 命令当たり，平均で 0.71 サイクルのストール

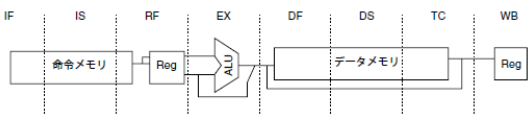
補足：MIPS R4000 (1991) パイプライン

- 整数パイプラインを8段とすることで，クロック周波数を向上させる．
- キャッシュアクセスの時間が厳しいため，ここに追加のパイプラインを割り当てる．
- 深いパイプラインは，**スーパーパイプライン**とよばれることがある．

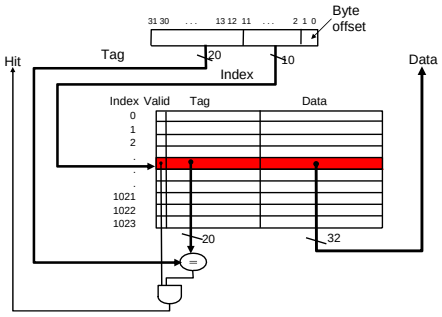


R4000 (1991) のステージ構成

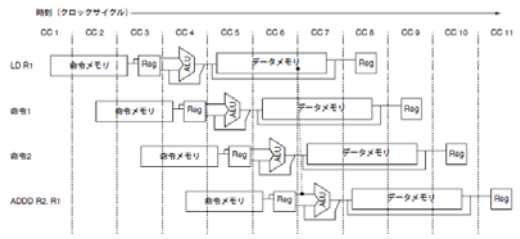
- IF：命令フェッチの前半，PC の選択、命令キャッシュ(8KB)のアクセス開始
- IS：命令フェッチの後半，命令キャッシュアクセスを完了
- RF：命令デコードとレジスタフェッチ、ハザードチェック、そして命令キャッシュのヒット検出
- EX: 実行
- DF：データフェッチ、データキャッシュ(8KB) アクセスの前半
- DS：データフェッチの後半、データキャッシュアクセスの完了
- TC：タグのチェック、データキャッシュアクセスヒットの決定
- WB：ロード演算とレジスタ-レジスタ間演算の結果格納



MIPS Direct Mapped Cache Example



MIPS R4000: 2サイクルのロード遅延



- データ値はDSステージで利用可能
- TCステージでのタグ比較によりミスが判明すると，1サイクル後退する.

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

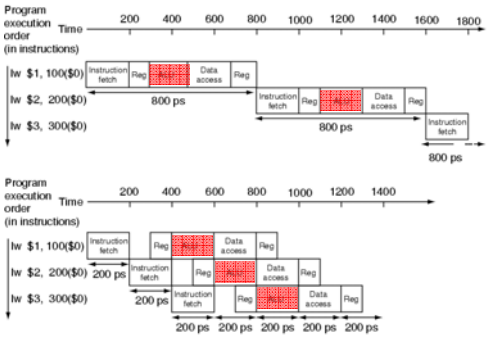
ハザード (hazard)

- **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，命令i の後に命令j が実行される場合.
    - RAW (read after write) 命令iが書き込み前に，命令jがそれを読みだそうとする.
    - WAW (write after write) 命令iが書き込む前に，命令jが書き込もうとする.
    - WAR (write after read) 命令iが読む前に，命令jがそこに書こうとする.
- **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

計算機アーキテクチャ 第二 (O)

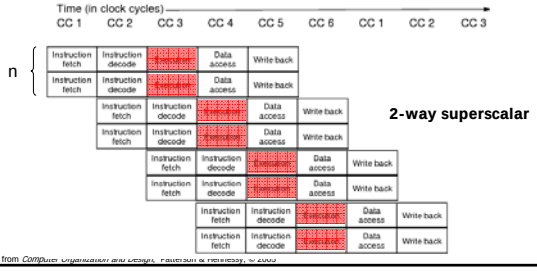
8. アウトオブオーダ実行プロセッサ

パイプライン処理 (pipelining)とスカラプロセッサ



スーパースカラプロセッサと命令レベル並列性

- 複数のパイプライを利用して IPC (instructions per cycle) を 1以上に引き上げる
  - n-way スーパースカラ
- ハザードの積極的な解消とストールの隠蔽が重要

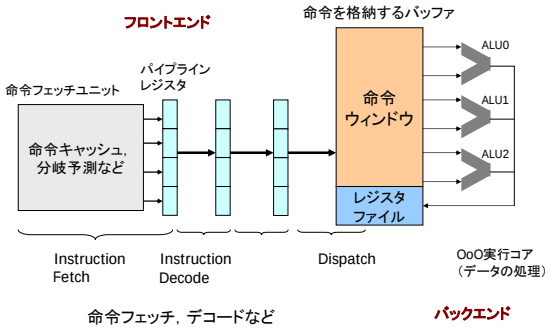


スーパースカラプロセッサにおける動的スケジューリング（アウトオブオーダ実行）

(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

アウトオブオーダ実行プロセッサの概要



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
- 講義用の計算機
  - 131.112.16.56 (情報工学科の演習室からは入れません)
  - ssh archo@131.112.16.56
    - mkdir myname
    - cd myname