計算機アーキテクチャ

第18回　ハザードとその解決法

2014年　10月17日

電気情報工学科　　田島　孝治

授業スケジュール（後期）

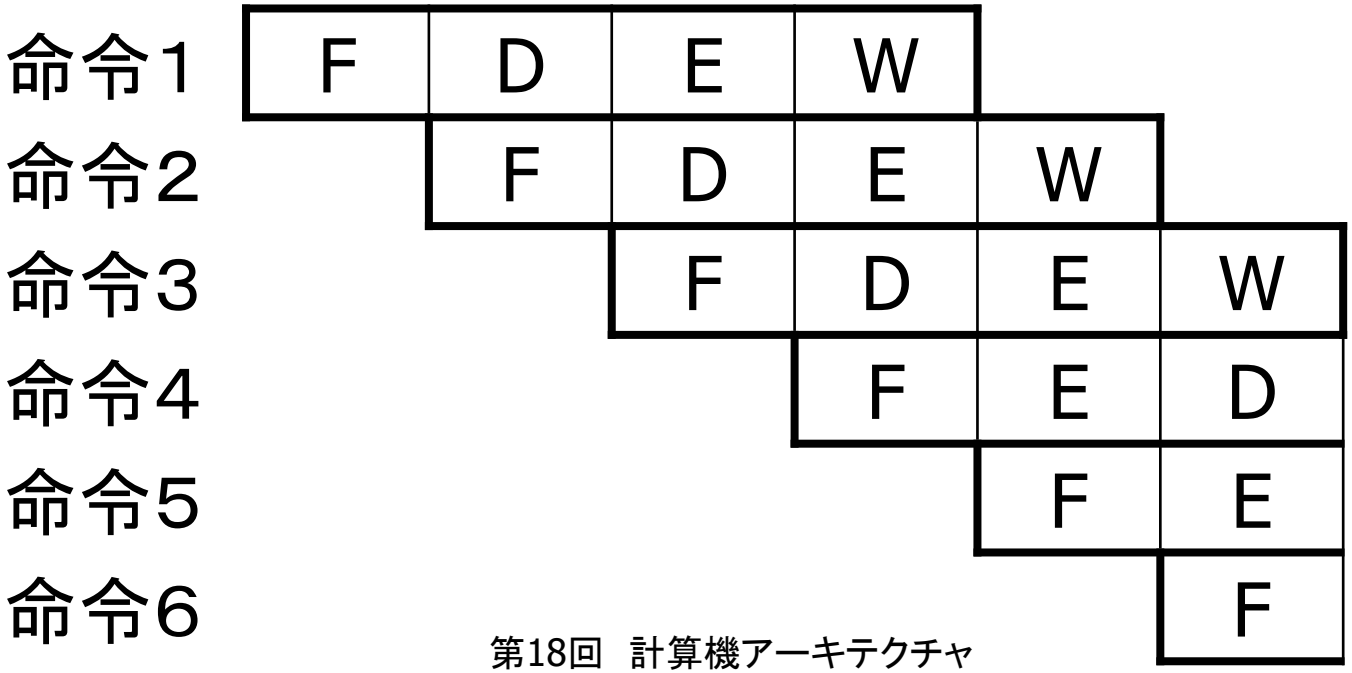
回	日付	タイトル
17	10/7	パイプライン処理
18	10/17	ハザードの解決法
19	10/21	並列処理
20	11/11	マルチプロセッサ
21	11/18	入出力装置の分類と特徴
22	11/25	割り込み
23	12/2	ネットワークアーキテクチャ
24	12/9	中間試験（日程は仮）

回	日付	タイトル
25	12/16	メモリ構成の基本原理
26	12/30	記憶領域の管理手法
27	1/13	仮想メモリ
28	1/20	キャッシュ
29	1/27	システムアーキテクチャ
30	2/3	組み込みコンピューティング
31	2/10	コンピュータの設計と実装
	2/24?	期末試験（日程は仮）
32	3/3?	フォローアップ（日程は仮）

※10/14は出張のため振替です。（10/17 9:00～の予定）
※11/4は研修旅行です。試験の日程は未定です。

先週の復習

- パイプライン処理
  - 単位時間当たりの処理量を増加させる流れ作業

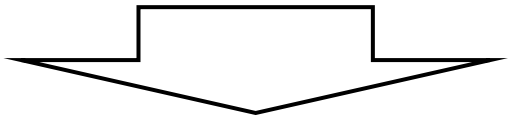


パイプラインの破綻：ハザード

- 構造ハザード
- データハザード
- 制御ハザード

構造ハザード

- 同じハードウェアを同時に使わないといけない場合に起こる



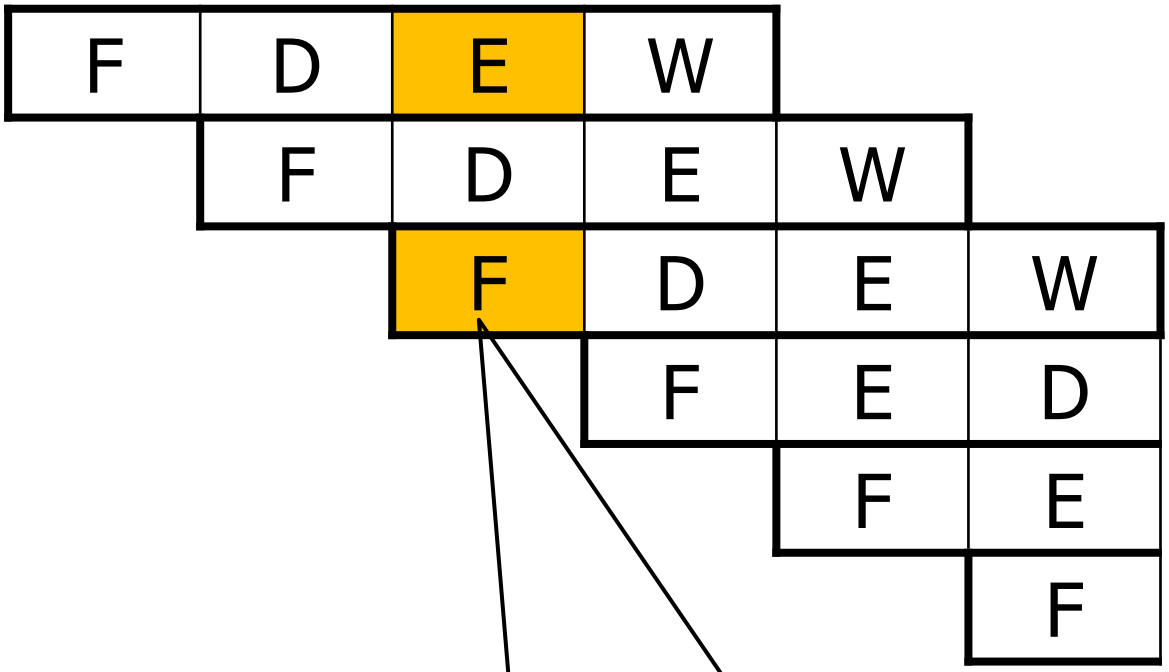
メモリに同時にアクセスする

・命令フェッチ

・データメモリ呼び出し

構造ハザードが起こる場合

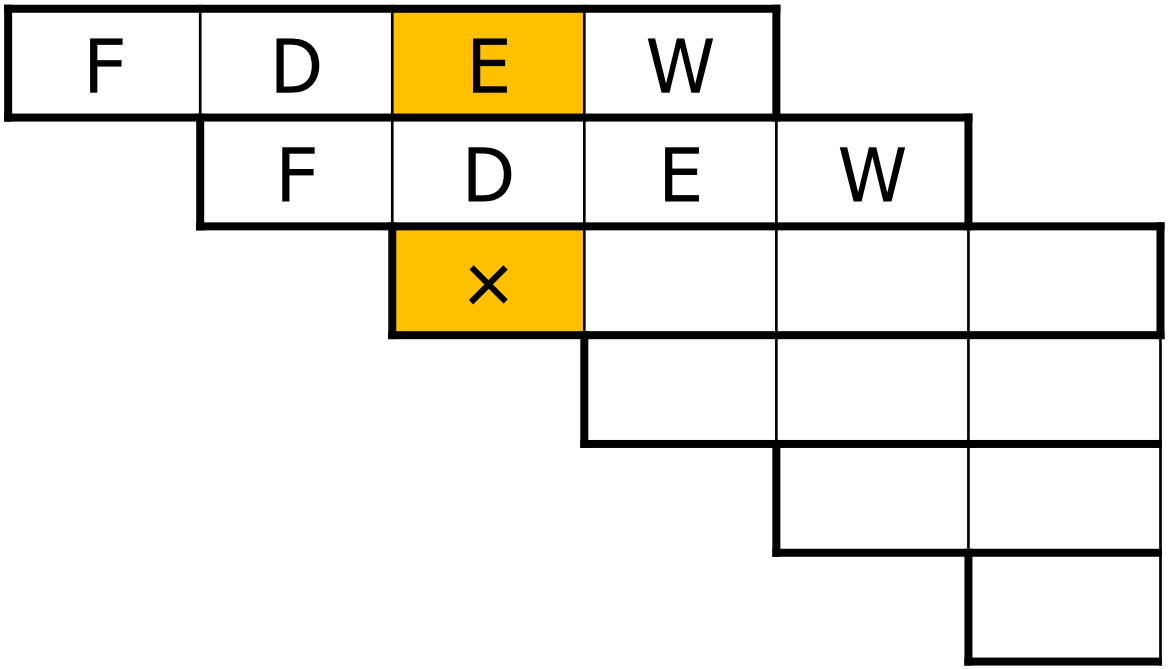
```
lw      $t4, 0($t2)
add     $s1,$t3,$t4
sub     $s2,$s1,$3
li      $t5, 5
li      $t6, 10
bne     $t5,$s2,loop
```



命令メモリを呼び出したい
でも前の命令がメモリを占有中

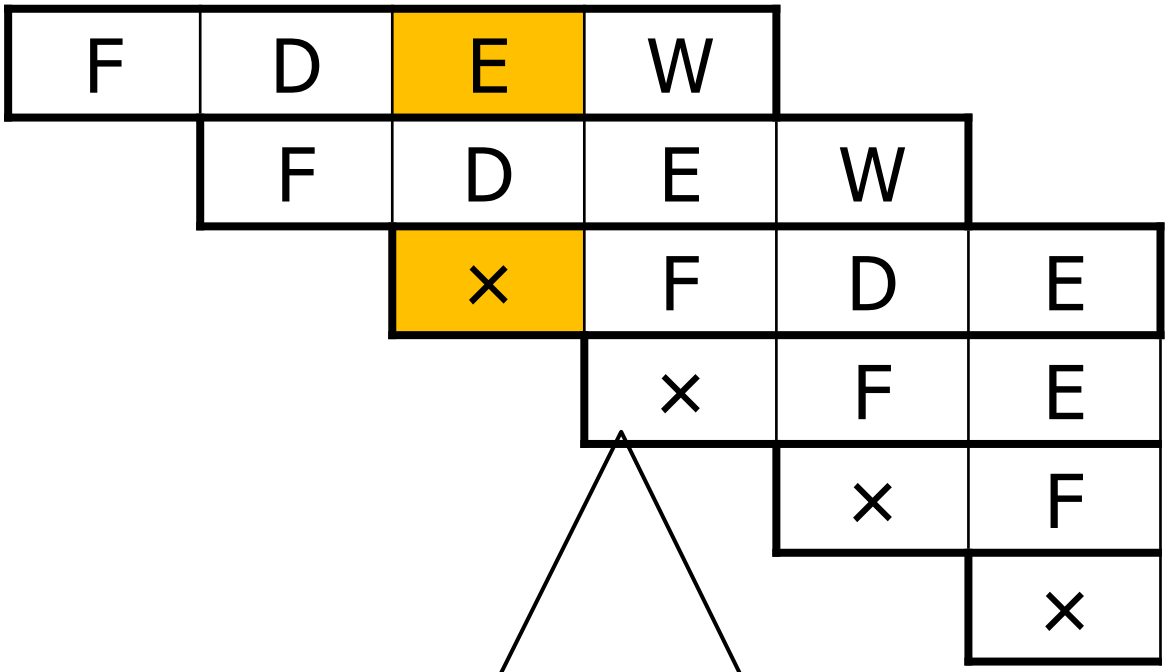
構造ハザードの結果

```
lw      $t4, 0($t2)
add     $s1,$t3,$t4
sub     $s2,$s1,$3
li      $t5, 5
li      $t6, 10
bne     $t5,$s2,loop
```



構造ハザードの結果

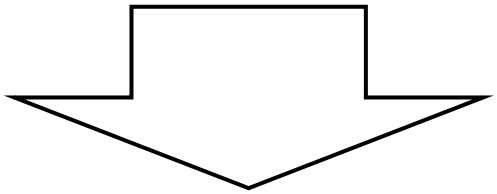
```
lw      $t4, 0($t2)
add     $s1,$t3,$t4
sub     $s2,$s1,$3
li      $t5, 5
li      $t6, 10
bne     $t5,$s2,loop
```



次以降の命令も実行が遅くなる

データハザード

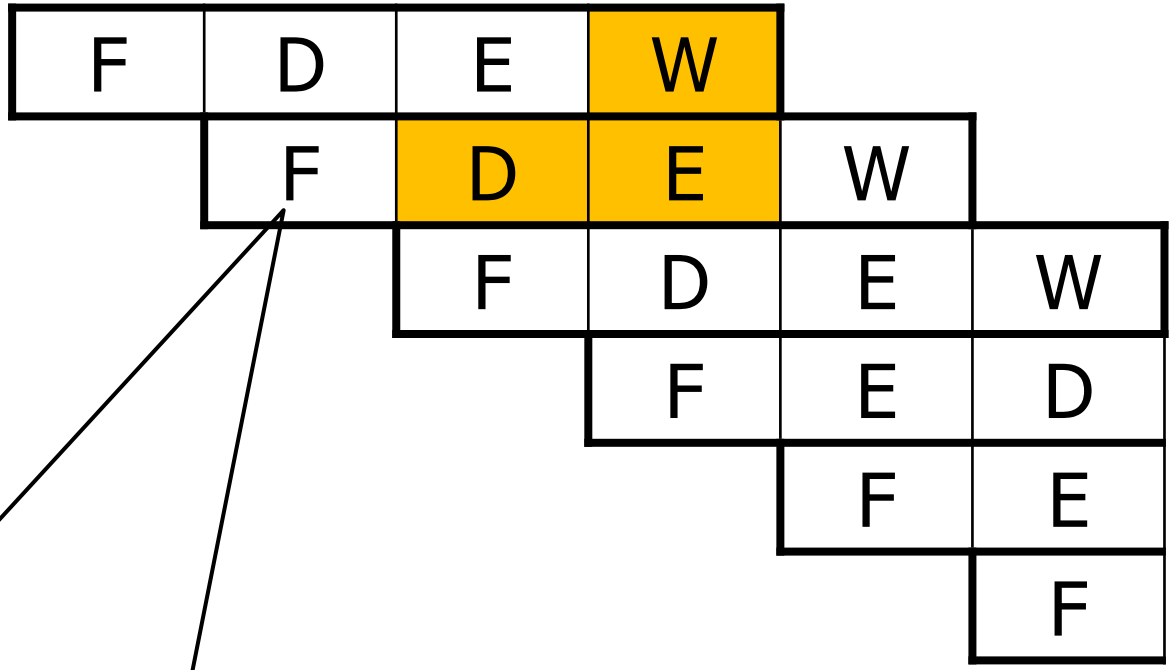
- 前の処理の結果（データ）を使わないと、次の処理が実行できない場合に起こる
- 前の命令の変数を共有している



前の計算結果を次の命令で利用する

データハザードが起こる場合

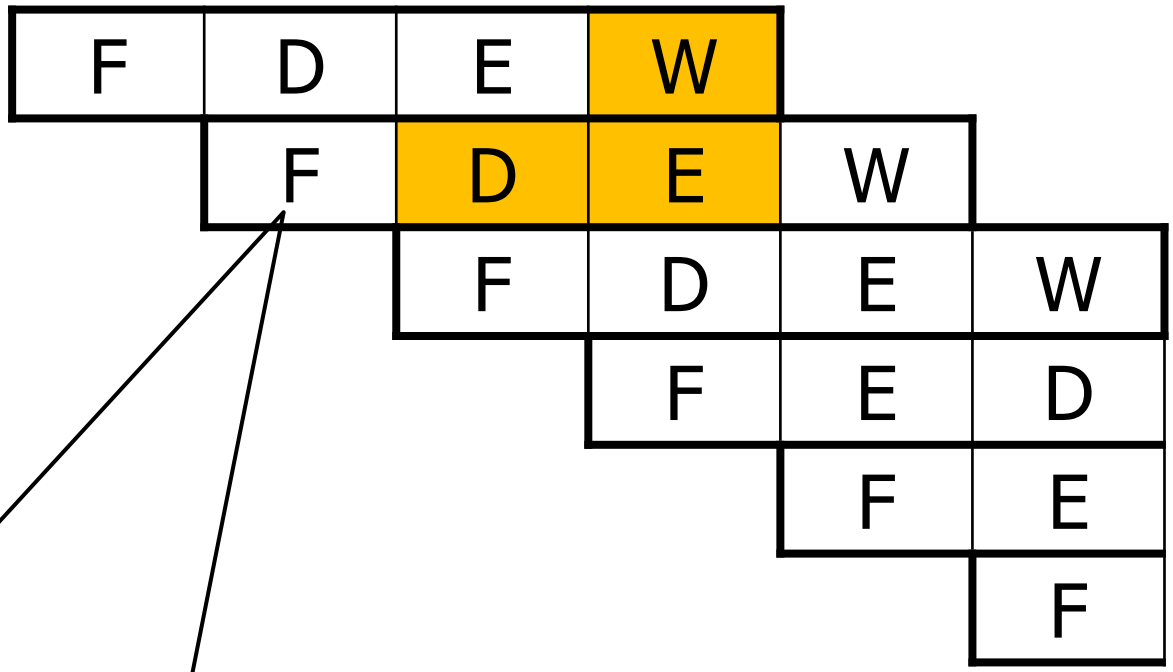
```
lw    $t4, 0($t2)
add   $s1,$t3,$t4
sub   $s2,$s1,$3
li    $t5, 5
li    $t6, 10
bne   $t5,$s2,loop
```



t4が決まらないとデコード
できない（ALUにデータが送れない）

データハザードが起こる場合

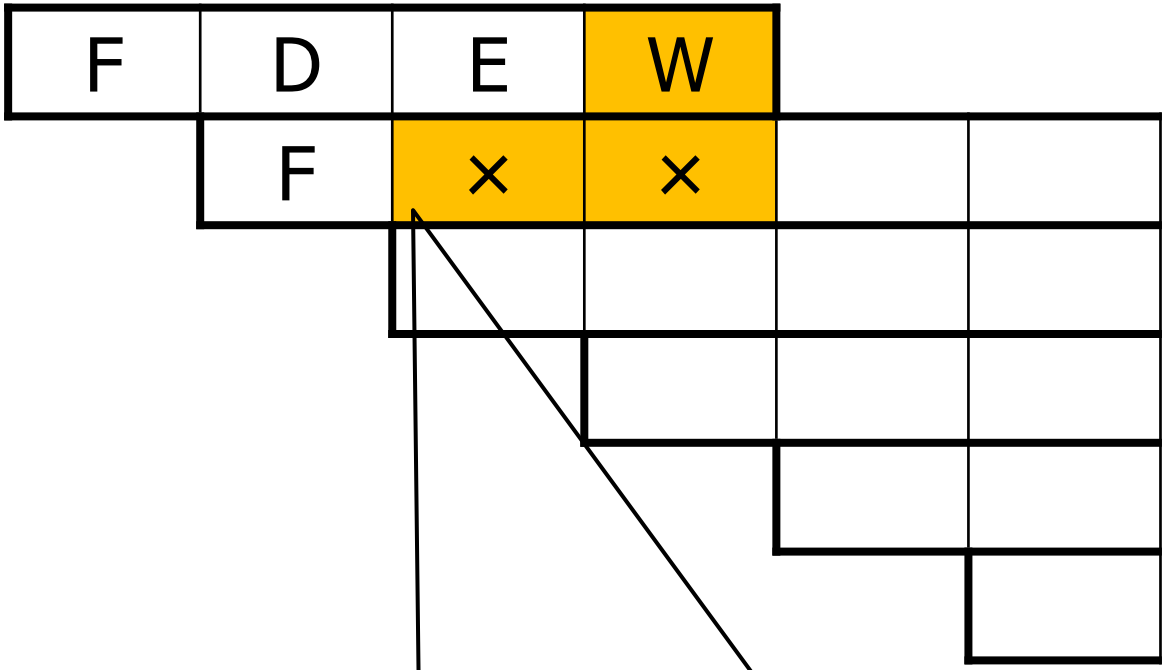
```
lw    $t4, 0($t2)
add   $s1,$t3,$t4
sub   $s2,$s1,$3
li    $t5, 5
li    $t6, 10
bne   $t5,$s2,loop
```



t4が決まらないとデコード
できない（ALUにデータが送れない）

データハザードが起こる場合

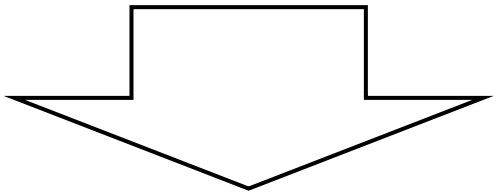
```
lw    $t4, 0($t2)
add   $s1,$t3,$t4
sub   $s2,$s1,$3
li    $t5, 5
li    $t6, 10
bne   $t5,$s2,loop
```



t4が決まらないとデコード
できない（ALUにデータが送れない）

制御ハザード

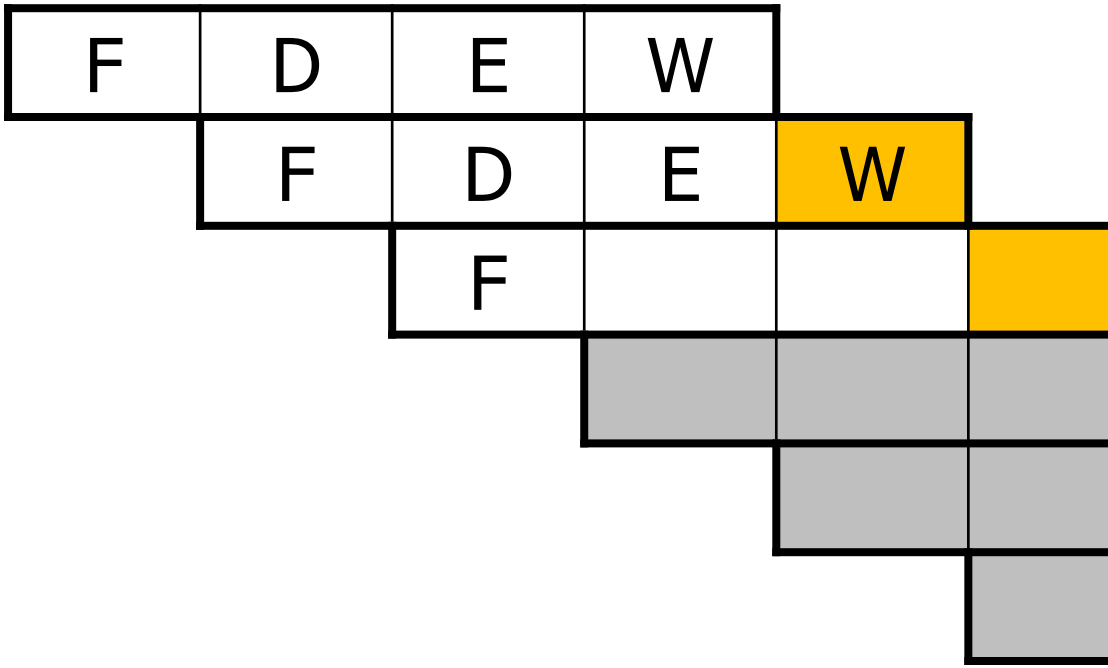
- 分岐命令がある場合に起こる



先読みした命令が実行されない

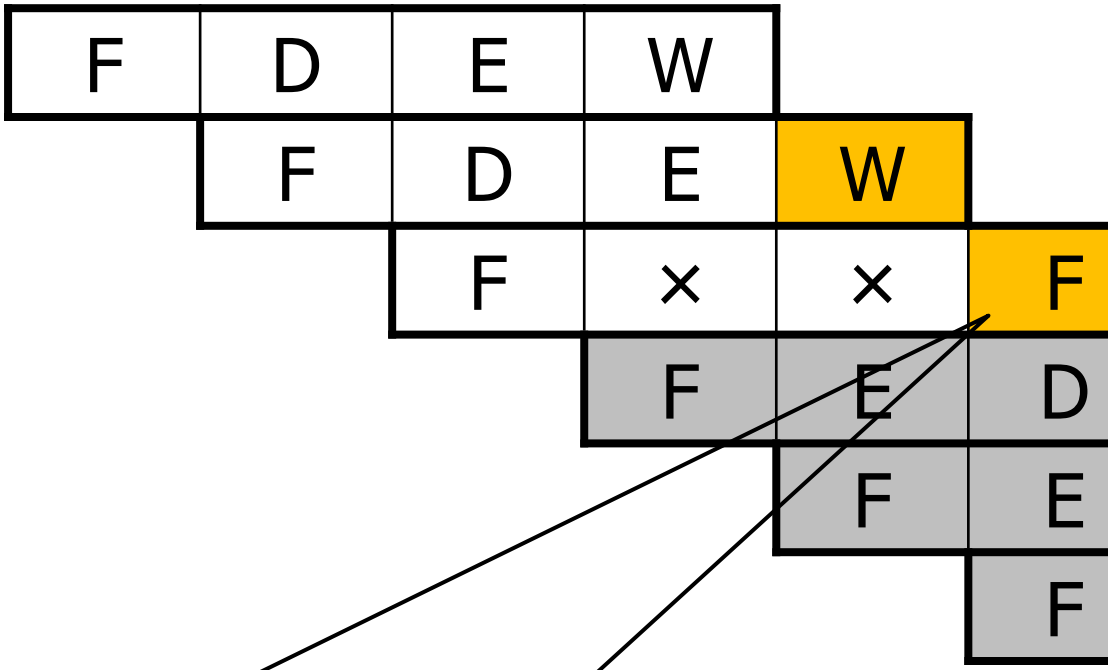
制御ハザードが起こる場合

```
loop:   $t4, 0($t2)
bne     $t5,$s2,end
add     $s2,$s1,$3
sub     $t5,$s2,$s1
j loop      $t6, 10
end: add $t5,$zero,$zero
```



制御ハザードが起こる場合

```
loop:   $t4, 0($t2)
bne     $t5,$s2,end
add     $s2,$s1,$3
sub     $t5,$s2,$s1
j loop      $t6, 10
end: add $t5,$zero,$zero
```



Wまで終わってPCが更新される
＝次に実行される行が決まる

ハザードが起こった場合の手順

- 次のプログラムを実行する際のパイプラインを示せ
  - プログラムは次ページに明記する
  - ハザードは指定した条件を満たした場合のみ起こるものとする
- この際のスループットも求めよ
  - 全てのステージの実行時間は1クロックとする

実行するプログラム

```
la      $s0, values
addi    $s3,$s2,4
sw      $s1, 0($s0)
lw      $t1, 4($s0)
add     $s4,$t3,$t1
sw      $s4, 8($s0)
```

ハザードの条件

- lw命令の実行(E)はフェッチ(F)と同時にできない
- sw命令の実行(E)は結果格納(W)と同時にできない
- 前の命令の実行結果を使う命令のデコード(D)は、前の命令の結果格納(W)が終わった後にしかできない
- 制御ハザードはなし
（分岐命令がないですね）

ハザードのまとめ

- 何らかの原因により命令をパイプライン動作させられない状態を【　　】いう
- 【　　】ハザード:
  - 同じハードウェアの同時利用が原因
- 【　　】ハザード:
  - 変数の共有等が原因
- 【　　】ハザード:
  - 分岐命令が原因

ハザードのまとめ

- 何らかの原因により命令をパイプライン動作させられない状態を【ハザード】いう
- 【　　】ハザード:
  - 同じハードウェアの同時利用が原因
- 【　　】ハザード:
  - 変数の共有等が原因
- 【　　】ハザード:
  - 分岐命令が原因

ハザードのまとめ

- 何らかの原因により命令をパイプライン動作させられない状態を【ハザード】いう
- 【構造】ハザード:
  - 同じハードウェアの同時利用が原因
- 【　　】ハザード:
  - 変数の共有等が原因
- 【　　】ハザード:
  - 分岐命令が原因

ハザードのまとめ

- 何らかの原因により命令をパイプライン動作させられない状態を【ハザード】いう
- 【構造】ハザード:
  - 同じハードウェアの同時利用が原因
- 【データ】ハザード:
  - 変数の共有等が原因
- 【　　】ハザード:
  - 分岐命令が原因

ハザードのまとめ

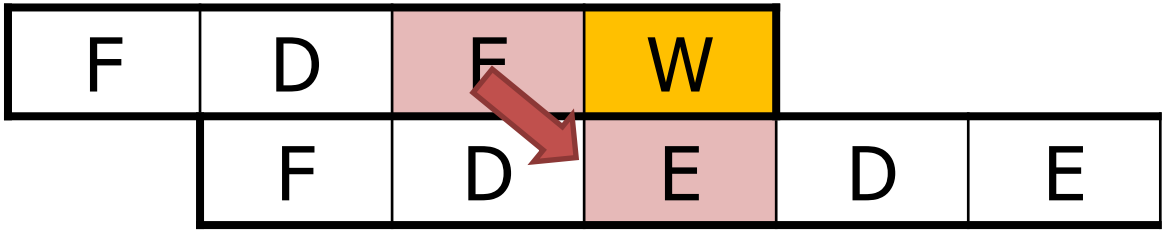
- 何らかの原因により命令をパイプライン動作させられない状態を【ハザード】いう
- 【構造】ハザード:
  - 同じハードウェアの同時利用が原因
- 【データ】ハザード:
  - 変数の共有等が原因
- 【制御】ハザード:
  - 分岐命令が原因

ハザードの解消

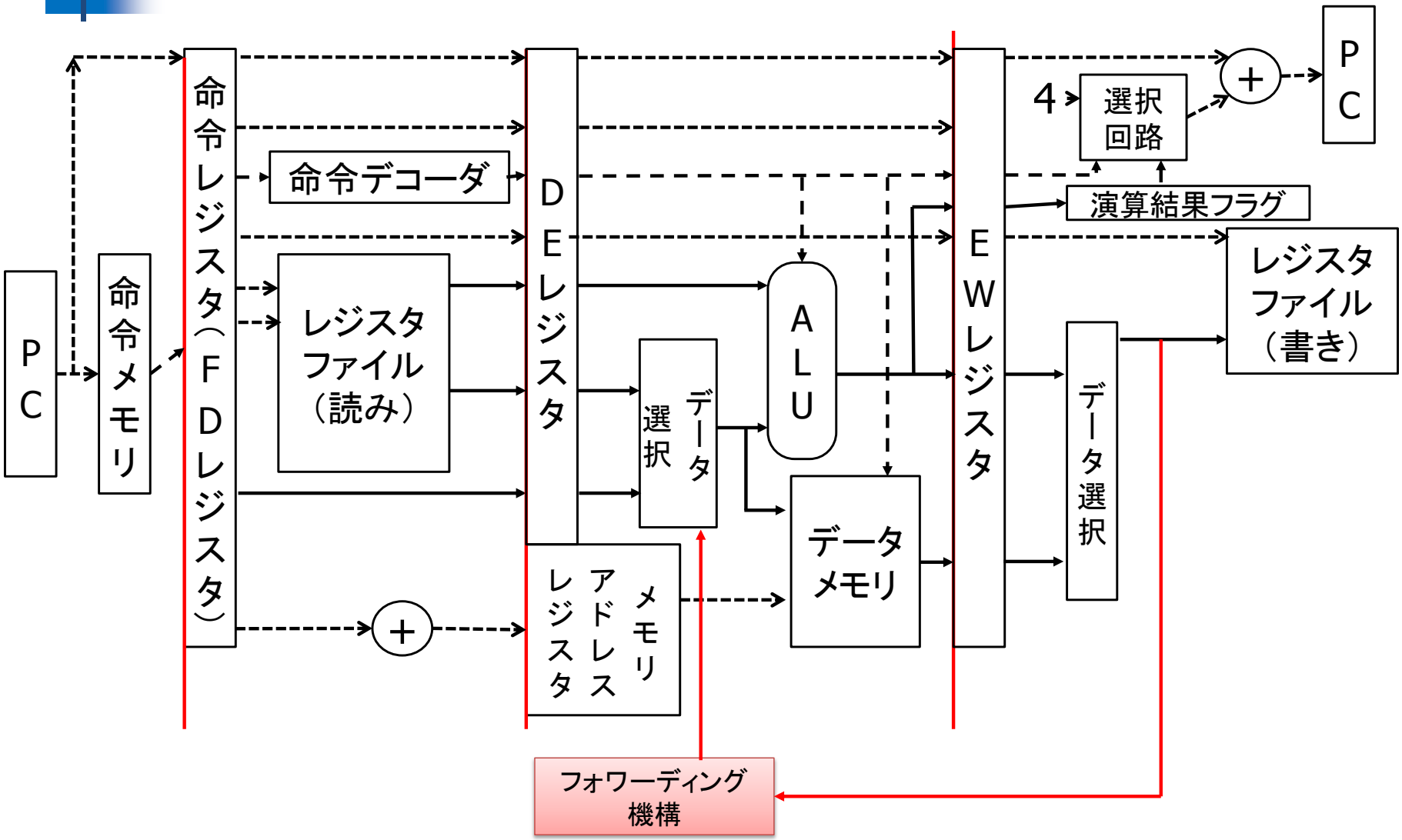
データハザードの対策

- 計算結果を次の命令の実行ステージに書き戻す
  - レジスタから読み込まなくても、前の命令の実行結果を使えるようにハードウェアを改良

```
lw    $t4, 0($t2)
add   $s1,$t3,$t4
```



フォワーディング機構


PC
命令メモリ
命令レジスタ（FDレジスタ）
命令デコーダ
レジスタファイル（読み）
DEレジスタ
データ選択
ALU
メモリアドレスレジスタ
データメモリ
EWレジスタ
4
選択回路
演算結果フラグ
PC
レジスタファイル（書き）
データ選択
フォワーディング機構

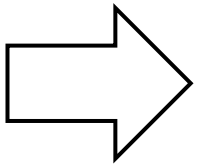
フォワーディング機構

- フォワーディング（forwarding）
  - データハザードを解決するハードウェア機構
  - バイパシング、ショートカットとも呼ばれる
- 計算命令で使う変数の共有を考える必要がなくなった

制御ハザードの解消

- 条件分岐によるハザードの解消
  - 命令アドレスがいつ決まるかが重要

- 無条件分岐
  （j命令、jr命令）



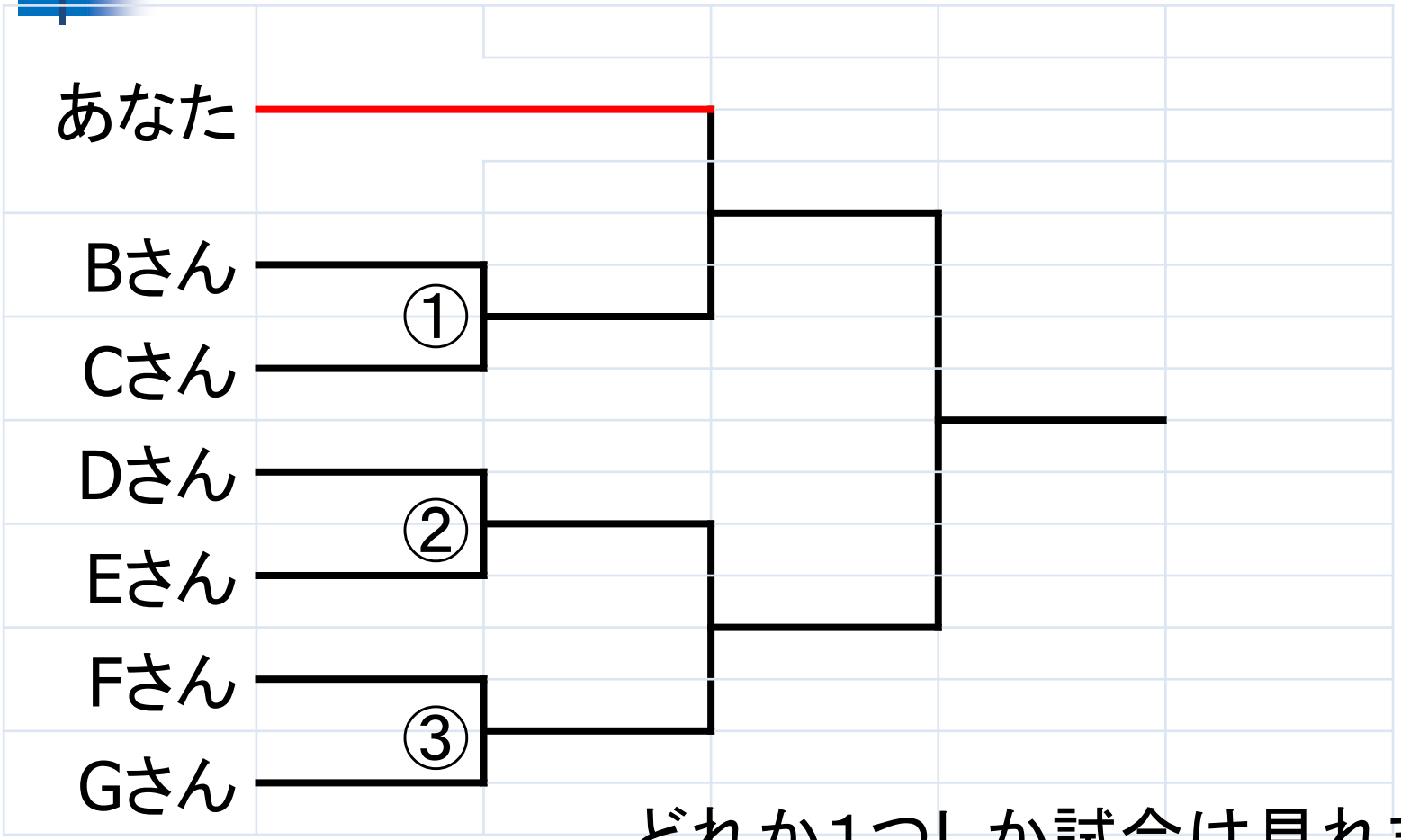
Dステージ直後
に決定

直後にPCを更新

条件がある場合

- Wステージが終わるまではどこに分岐するかわからない
- どうやって無駄な時間をすごすか
- 質問です
  - あなたはゲームのトーナメント戦に出場しています
  - １試合目に勝ちました、２試合目までの時間をどう過ごしますか？

1試合目終了時点でのトーナメント表



どれか1つしか試合は見れません

①～③のどれを見に行きますか？

遅延分岐

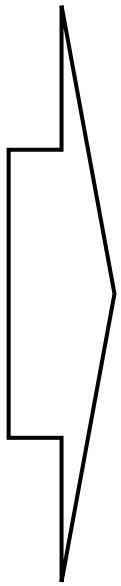
- 分岐をしてもしなくても直後に同じ命令を実行する場合に有効

```
        beq  $t1,t2,sub2
sub1:   la   $t0, a
        lw   $s0, 0($t0)
        sub  $s2,$s0,$s1
sub2:   lw   $s0, 0($t0)
        la   $t0, a
        add  $s2,$s0,$s1
```

同じ命令を実行

遅延分岐の結果

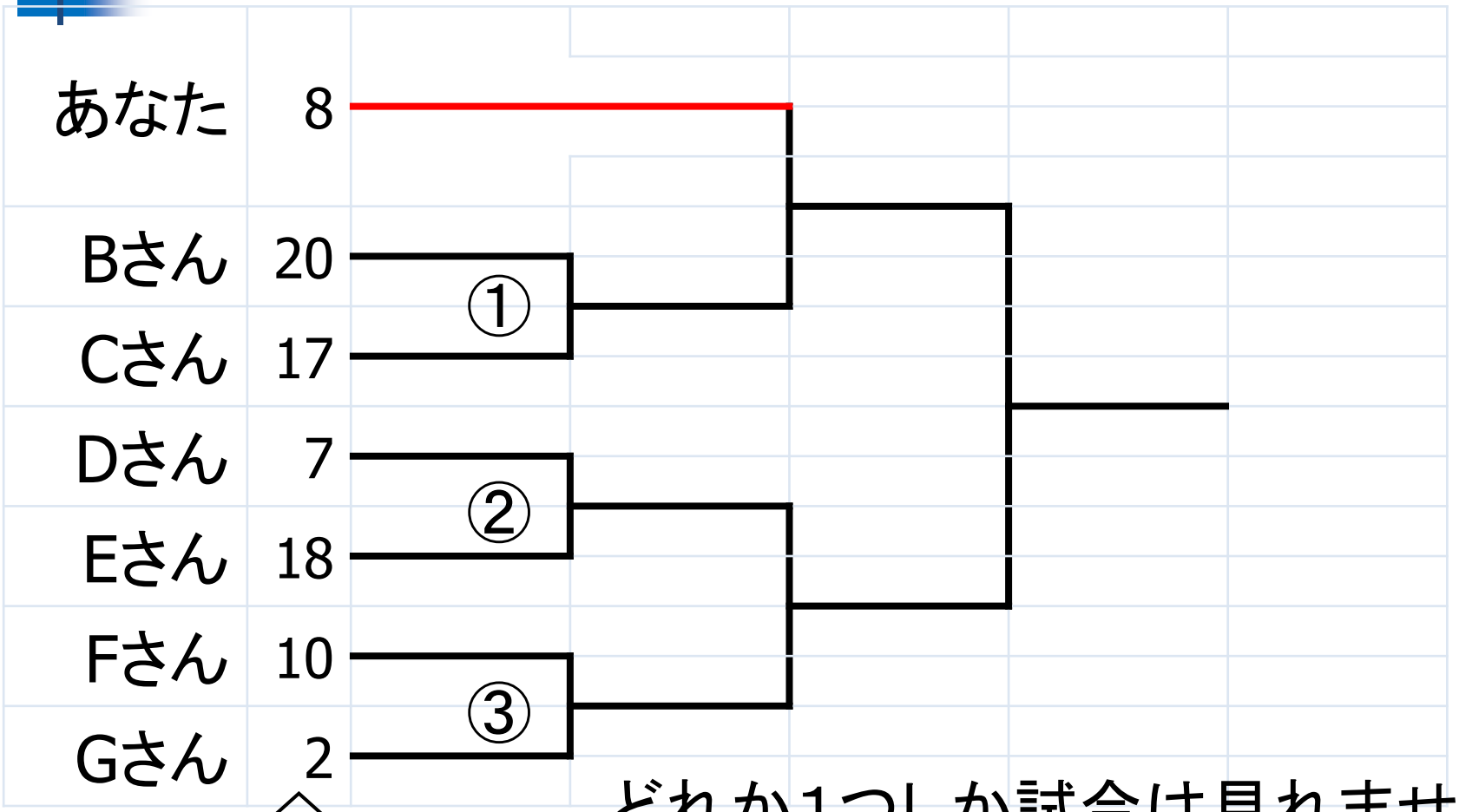
```
beq  $t1,t2,sub2
sub1:
la   $t0, a
lw   $s0, 0($t0)
sub  $s2,$s0,$s1
sub2:
lw   $s0, 0($t0)
la   $t0, a
add  $s2,$s0,$s1
```



```
beq  $t1,t2,sub2
la   $t0, a
lw   $s0, 0($t0)
sub1:
sub  $s2,$s0,$s1
sub2:
add  $s2,$s0,$s1
```

命令を精査することで解決

1試合目終了時点でのトーナメント表



現在の
世界ランク

どれか1つしか試合は見れません

①～③のどれを見に行きますか？

分岐予測

- 分岐が起こるかどうかを予測して処理を進める
  - 事前に起こりそうな方を進めておく
  - 予測がはずれたら、事前に進めた分は破棄

ある意味で、当たり前な方法ですね

分岐予測

- 分岐が起こるかどうかを予測して処理を進める
  - 事前に起こりそうな方を進めておく
  - 予測がはずれたら、事前に進めた分は破棄

でもどうやって予想するの？

どうやって次を予測するのか？

- あなたは、バス（または電車）に乗って通学しています
- できれば椅子に座りたいと考えていますが、あいにく満席でした
- どこ（どんな人の横、または場所）で席が空くのを待ちますか？
  - 理由も考えてください

方法1：常に分岐しないと予想

- 常に分岐すると予想しても同義

- 当たるか外れるかは運しだい
- ただ待っているだけより良い
- 予想が当たれば無駄が少ない

方法2：アドレスが小さい方を予測

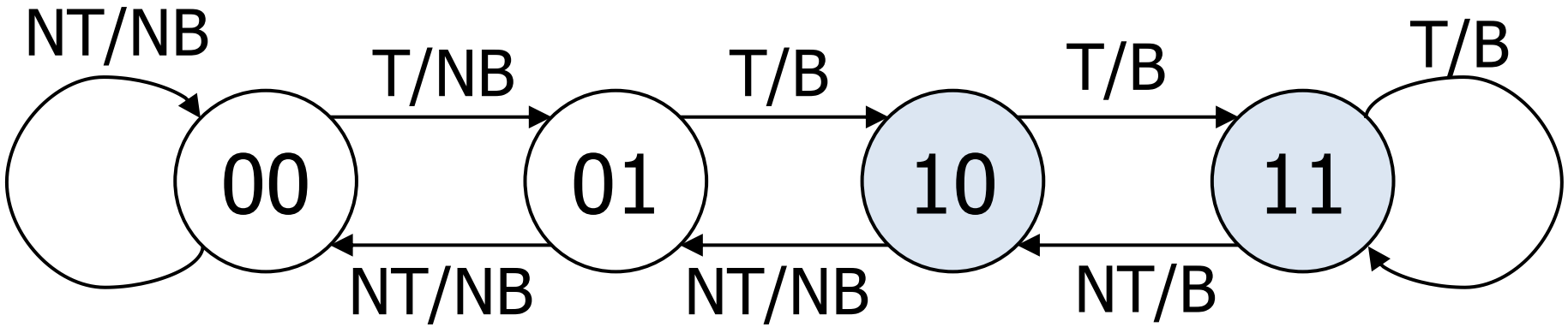
- もっともよく使われる条件分岐

ループ（繰り返し）処理

  - １回で終わることは少ない
  - 繰り返す場合は、前のアドレス（少ない方）に戻る

方法3：分岐予測テーブルを使う

- 2ビット予測器
  - 予想が2回続けて外れたら予測を変える

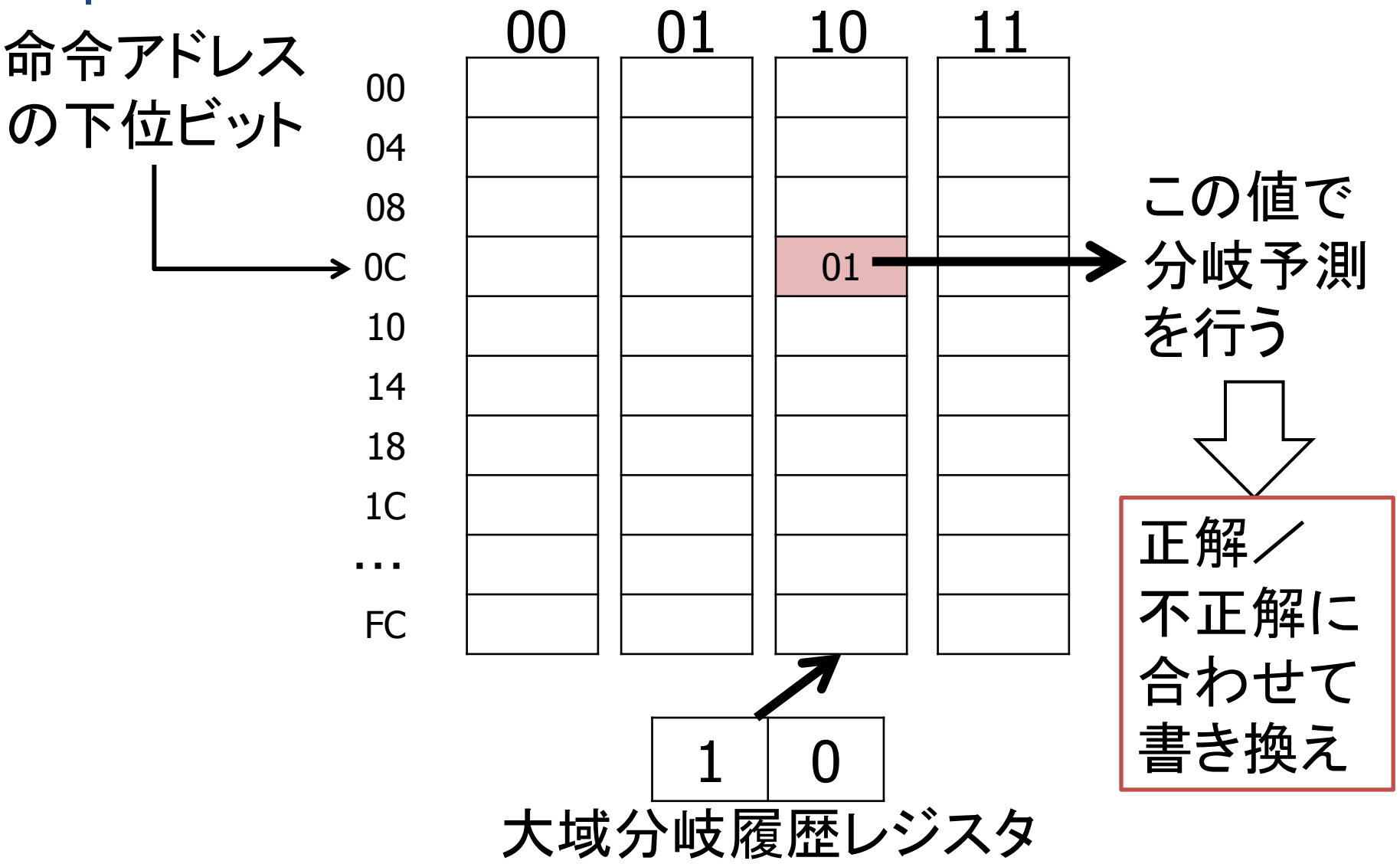


入力		出力	
T	分岐した	B	分岐させる
NT	分岐しなかった	NB	分岐させない

分岐予測テーブル

- 2ビット予測器とテーブルを組み合わせて予測率を高める方式
- 各分岐命令ごとの履歴とすべての分岐の履歴を組み合わせるアイデア
- 平均的な成功率は90%を超える

分岐予測テーブル


命令アドレス
の下位ビット
00
01
10
11
00
04
08
0C
10
14
18
1C
…
FC
01
この値で
分岐予測
を行う
正解／
不正解に
合わせて
書き換え
1
0
大域分岐履歴レジスタ

ハザードの解消のまとめ

- データハザードには【　　】が有効
- 構造ハザードと、データハザードは、【　　】の改良で対処する
- 制御ハザードの対策は、【　　】をいつ決めるかが重要
- 先に共通して命令を実行する【　　】と、次にどの命令が実行されるか予測する【　　】がある

ハザードの解消のまとめ

- データハザードには【フォワーディング】が有効
- 構造ハザードと、データハザードは、【　　】の改良で対処する
- 制御ハザードの対策は、【　　】をいつ決めるかが重要
- 先に共通して命令を実行する【　　】と、次にどの命令が実行されるか予測する【　　】がある

ハザードの解消のまとめ

- データハザードには【フォワーディング】が有効
- 構造ハザードと、データハザードは、【ハードウェア】の改良で対処する
- 制御ハザードの対策は、【　　】をいつ決めるかが重要
- 先に共通して命令を実行する【　　】と、次にどの命令が実行されるか予測する【　　】がある

ハザードの解消のまとめ

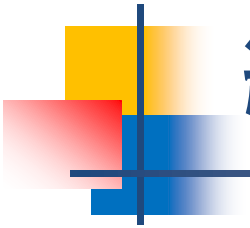
- データハザードには【フォワーディング】が有効
- 構造ハザードと、データハザードは、【ハードウェア】の改良で対処する
- 制御ハザードの対策は、【命令アドレス】をいつ決めるかが重要
- 先に共通して命令を実行する【　　】と、次にどの命令が実行されるか予測する【　　】がある

ハザードの解消のまとめ

- データハザードには【フォワーディング】が有効
- 構造ハザードと、データハザードは、【ハードウェア】の改良で対処する
- 制御ハザードの対策は、【命令アドレス】をいつ決めるかが重要
- 先に共通して命令を実行する【遅延分岐】と、次にどの命令が実行されるか予測する【　　】がある

ハザードの解消のまとめ

- データハザードには【フォワーディング】が有効
- 構造ハザードと、データハザードは、【ハードウェア】の改良で対処する
- 制御ハザードの対策は、【命令アドレス】をいつ決めるかが重要
- 先に共通して命令を実行する【遅延分岐】と、次にどの命令が実行されるか予測する【分岐予測】がある



練習問題

- 次のプログラムの分岐命令（3か所）に関して、分岐予測の正解率を求めよ
  - 次の4つの場合を考える事
    - 常に分岐しないと考える
    - 常に小さいアドレスに分岐すると考える
    - 全体で1つの2ビット予測器を使う
    - 分岐命令単位で個別の2ビット予測器を使う


FIN