コンピュータアーキテクチャ（4）

坂井　修一
東京大学大学院 情報理工学系研究科 電子情報学専攻
東京大学　工学部　電子情報工学科／電気電子工学科

- はじめに
- パイプライン処理

はじめに

- 本講義の目的
  - コンピュータアーキテクチャの基本を学ぶ
- 時間・場所
  - 火曜日 8:30 – 10:15　工学部2号館241
- ホームページ（ダウンロード可能）
  - url:　http://www.mtl.t.u-tokyo.ac.jp/~sakai/hard/
- 教科書
  - **坂井修一『コンピュータアーキテクチャ』（コロナ社、電子情報レクチャーシリーズC-9）**
  - **坂井修一『実践　コンピュータアーキテクチャ』（コロナ社）**

  教科書通りやります
- 参考書
  - D. Patterson and J. Hennessy, Computer Organization & Design、3rd Ed.（邦訳『コンピュータの構成と設計』(第3版)上下（日系ＢＰ））
  - 馬場敬信『コンピュータアーキテクチャ』（改訂２版）、オーム社
  - 富田眞治『コンピュータアーキテクチャⅠ』、丸善
- 予備知識：　論理回路
  - 坂井修一『論理回路入門』、培風館
- 成績
  - 試験＋レポート＋出席

講義の概要と予定（１／２）

１．コンピュータアーキテクチャ入門

ディジタルな表現、負の数、実数、加算器、ＡＬＵ，フリップフロップ、レジスタ、計算のサイクル

２．　データの流れと制御の流れ

主記憶装置、メモリの構成と分類、レジスタファイル、命令、命令実行の仕組み、実行サイクル、算術論理演算命令、シーケンサ、条件分岐命令

３．命令セットアーキテクチャ

操作とオペランド、命令の表現形式、アセンブリ言語、命令セット、
算術論理演算命令、データ移動命令、分岐命令、アドレシング、
サブルーチン、RISCとCISC

４．パイプライン処理（１）

パイプラインの原理、命令パイプライン、オーバヘッド、構造ハザード、データハザード、制御ハザード

５．パイプライン処理（２）

フォワーディング、遅延分岐、分岐予測、命令スケジューリング

６．キャッシュ

記憶階層と局所性、透過性、キャッシュ、ライトスルーとライトバック、
ダイレクトマップ型、フルアソシアティブ型、セットアソシアティブ型、キャッシュミス

7．仮想記憶

仮想記憶、ページフォールト、TLB、物理アドレスキャッシュ、仮想アドレスキャッシュ、メモリアクセス機構

講義の概要と予定（２／２）

８．基本CPUの設計

ディジタル回路の入力、Verilog HDL、シミュレーションによる動作検証、アセンブラ、基本プロセッサの設計、基本プロセッサのシミュレーションによる検証

９．命令レベル並列処理（１）

並列処理、並列処理パイプライン、VLIW、スーパスカラ、並列処理とハザード

１０．命令レベル並列処理（２）

静的最適化、ループアンローリング、ソフトウェアパイプライニング、トレーススケジューリング

１１．アウトオブオーダ処理

インオーダーとアウトオブオーダー、フロー依存、逆依存、出力依存、
命令ウィンドウ、リザベーションステーション、レジスタリネーミング、
マッピングテーブル、リオーダバッファ、プロセッサの性能

１２．入出力と周辺装置

周辺装置、ディスプレイ、二次記憶装置、ハードウェアインタフェース、割り込みとポーリング、アービタ、DMA、例外処理

試験：　7月

前回のまとめ

- 命令セット
  - コンピュータの全ての命令の集まりである
- 命令の表現形式
  - 命令の2進数表現の形式、フィールドで区切られる。
  - 本システムではR型、I型、A型が存在する。
- アセンブリ言語
  - 機械語を記号で置き換える言語である。（アドレスをラベルにも書き換える）
- 算術論理演算命令
  - 四則演算やシフトなど、レジスタとレジスタまたはレジスタと即値との間で演算が行われ、結果はレジスタに格納される。
- データ移動命令
  - レジスタとメモリとの間でデータのコピーを行う。
- 分岐命令
  - 制御の流れを変更する。無条件分岐命令と条件分岐命令、サブルーチンコール命令が存在する。
- アドレシング
  - メモリアドレスの生成方式。即値アドレシング、ベース相対アドレシング、レジスタアドレシング、PC相対アドレシングが存在する。
- サブルーチン
  - 部分プログラムを再利用可能にしたもので、PCを含むレジスタの待避が必要である。スタックを用いる。

５．パイプライン処理

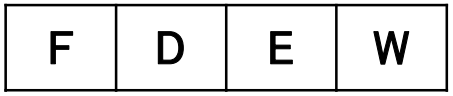
- 内容
  - 命令パイプライン
    - パイプラインの原理
    - 命令パイプラインの基本
    - 基本命令パイプラインの実現
  - 基本命令パイプラインの阻害要因
    - オーバヘッド
    - ハザード
      - 構造ハザード
      - データハザード
      - 制御ハザード
  - データハザードの解決法
    - フォワーディング
  - 制御ハザードの解決法
    - 命令アドレス生成のタイミング
    - 遅延分岐
    - 分岐予測
  - 命令スケジューリング

2週前の講義で

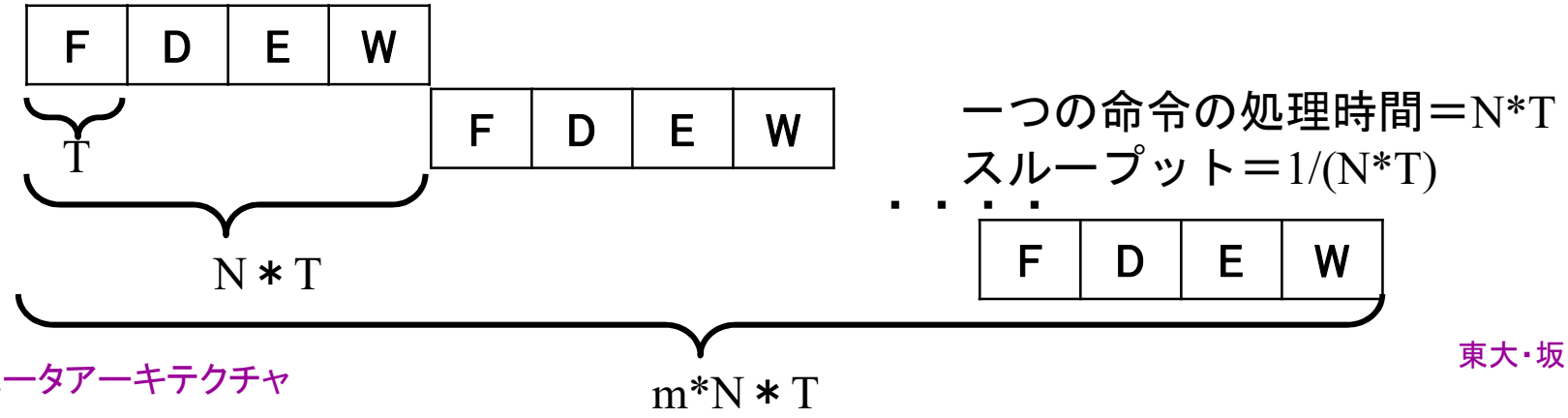
- 計算は通常以下の4ステップで実行されると言った

フェッチ（F)
↓
デコード（D)
（とレジスタ読みだし）
↓
実行（E)
↓
格納（W)

- これをタイミングできには以下のように記述できる。4クロックかかっている。
１クロックの実行時間がTとすると４＊Tの時間が必要



- m命令を実行する場合　m*4Tかかる。一般的には通常１命令がｎステップかかるとき、m*N*Tかかっているといえる。＝＞これをもっと速くすることはできないのか？

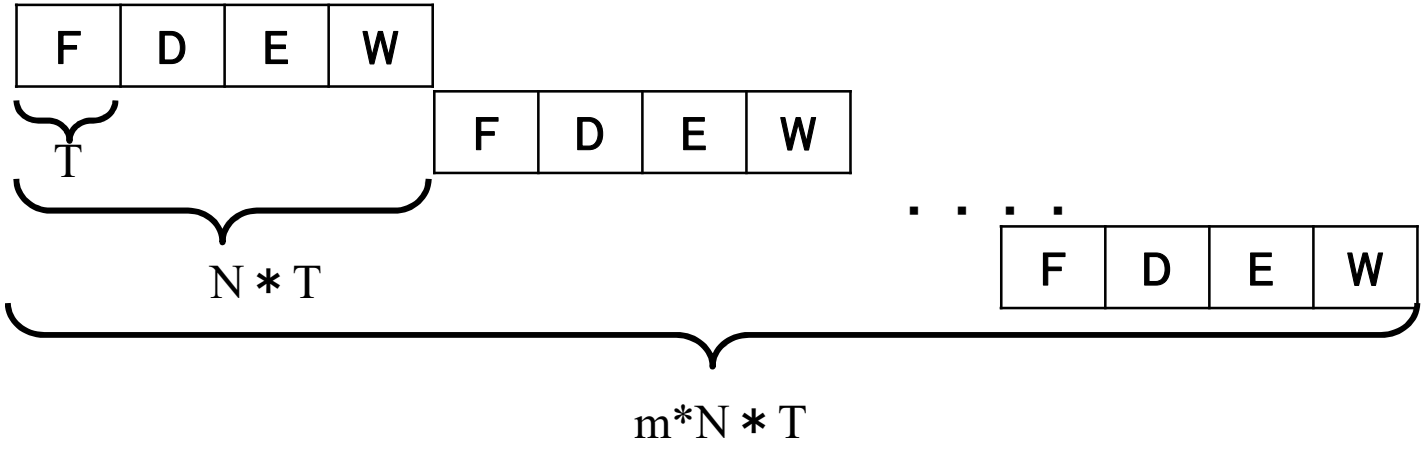


世の中で行われている事

- 工学部2号館のサンドウィッチ屋さんで毎日行っていること

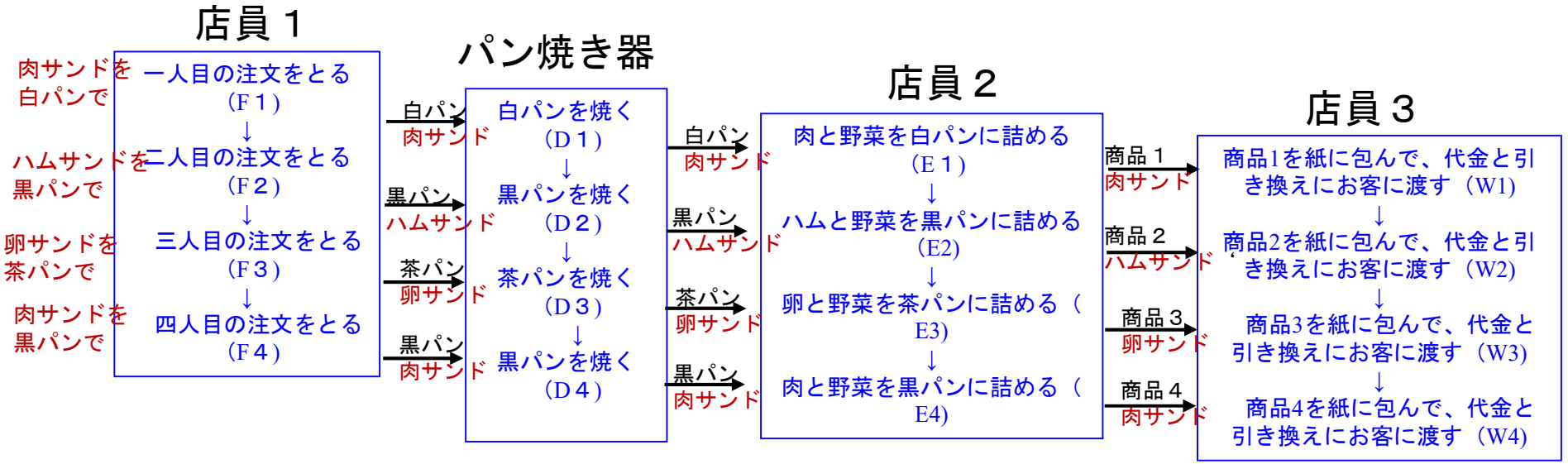
注文をとる（F)
↓
パンを焼く（D)
↓
肉と野菜をパンに詰める（E)
↓
紙に包んで、代金と引き換えにお客に渡す（W)

- これを一人でやっていたらm人のお客さんの注文をさばくには時間がかかりすぎて昼食時はお客をさばけない。

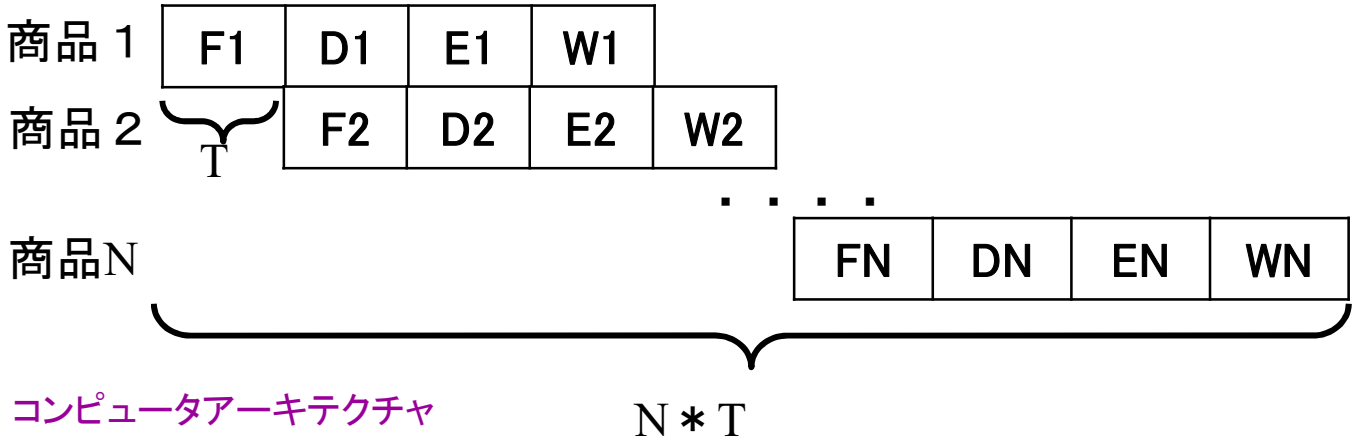


世の中で行われている事(2)

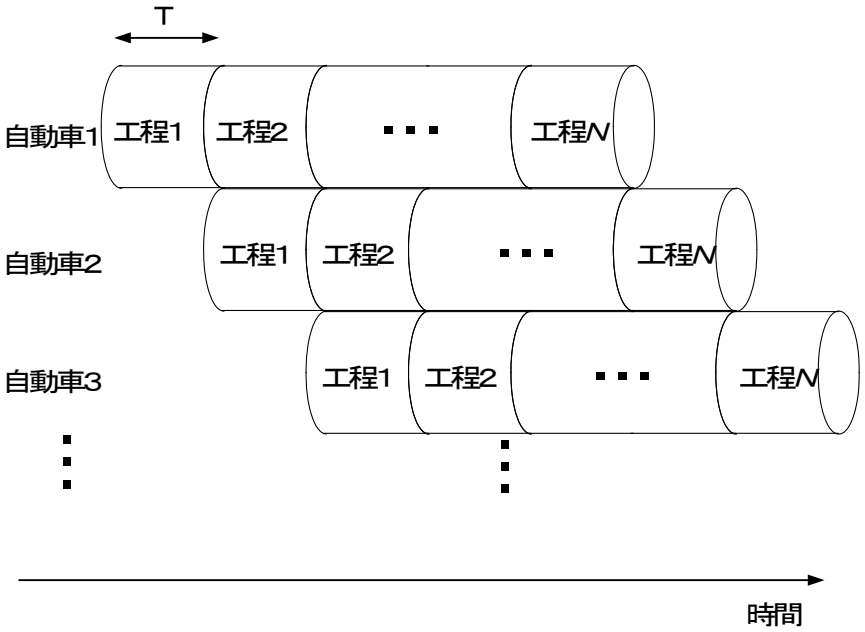
- そこで3人の店員とパン焼き器をつかって、流れ作業的に、パンと注文を引き渡しながら仕事をしていく



- Nが十分大きければN*Tで処理がさばける（商品ごとの時系列で図示）



パイプラインの原理



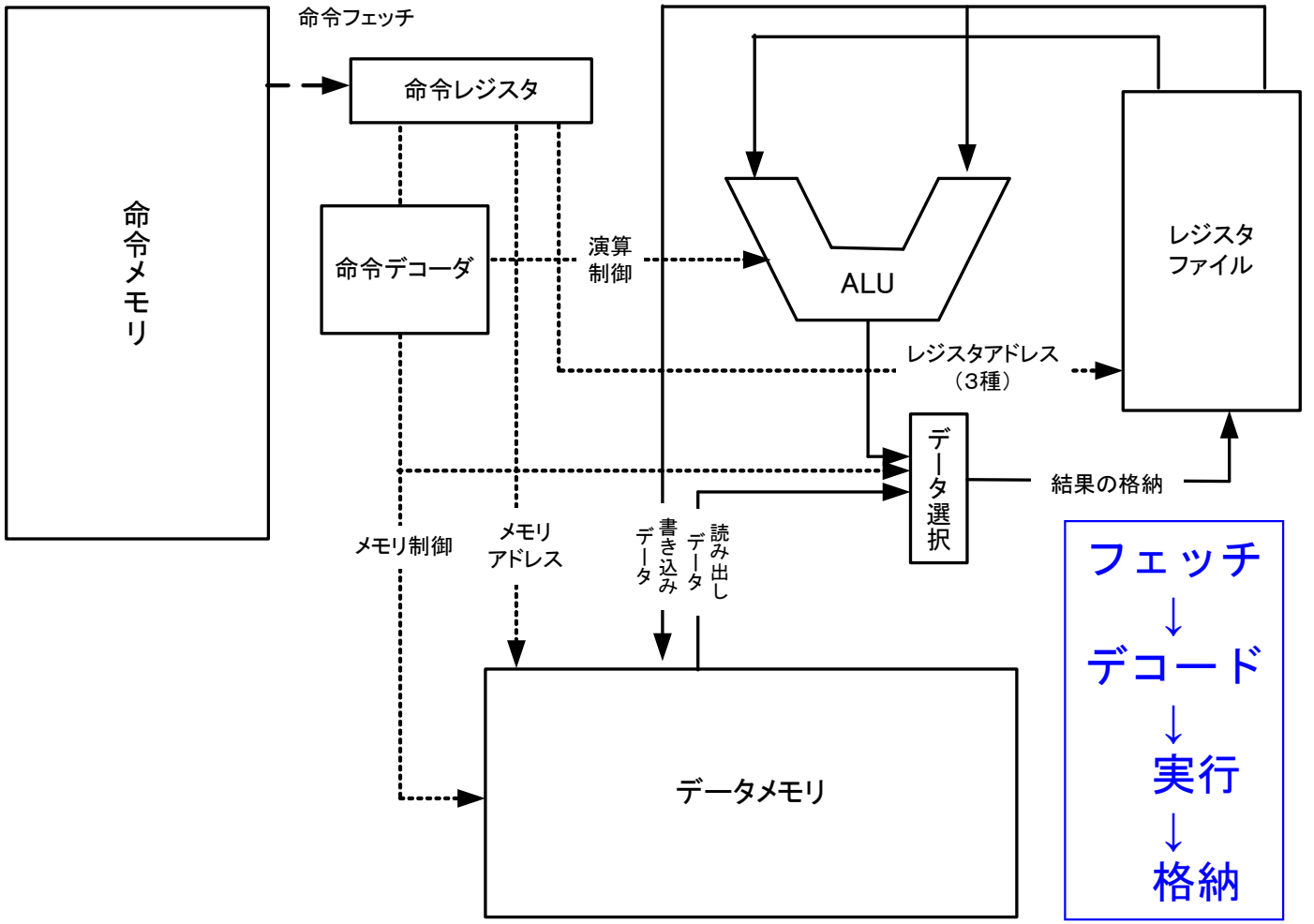
日産の自動車製造ライン

図 4.1 パイプラインの基本

- パイプライン
  - 全体の作業を多数の工程に分割し、各工程を並列に処理することで、単位時間あたりの処理量を飛躍的に向上させる流れ作業のこと

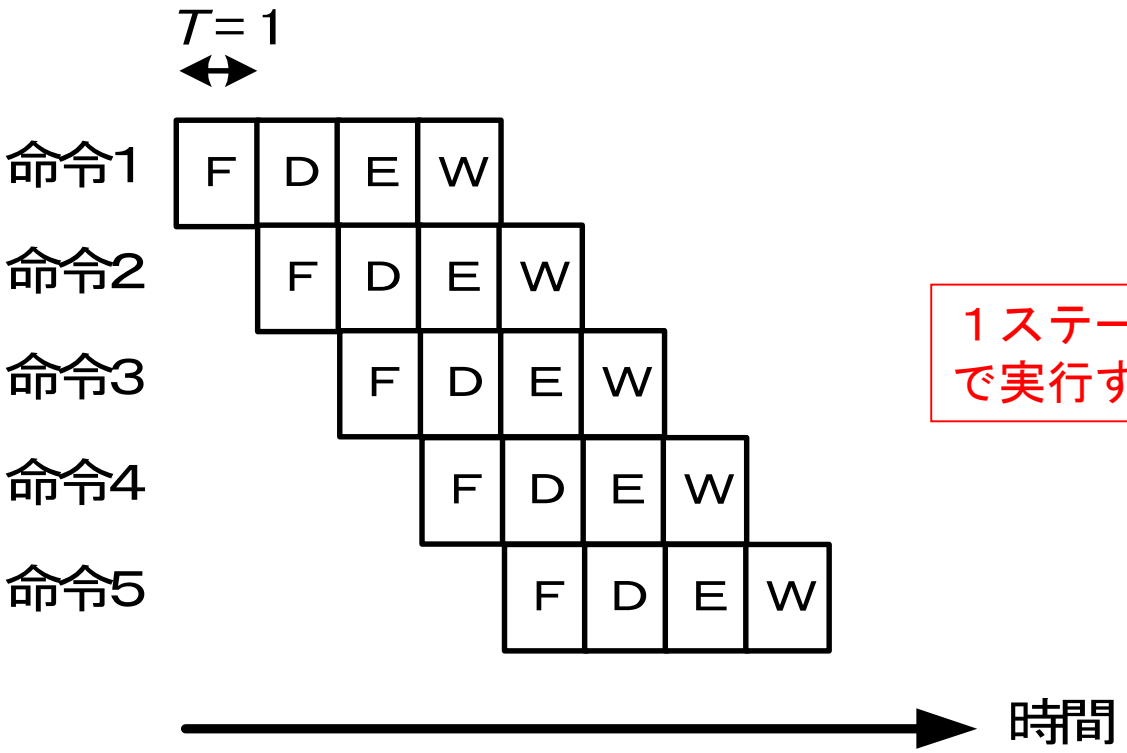
  - 実行時間： ひとつの作業の開始から終了までの時間 （N×T）
  - スループット： 単位時間当たりに完了する作業量 （1／T）
  - ステージ： 工程のこと

命令実行の基本形



これをそのままパイプラインにする

基本命令パイプライン

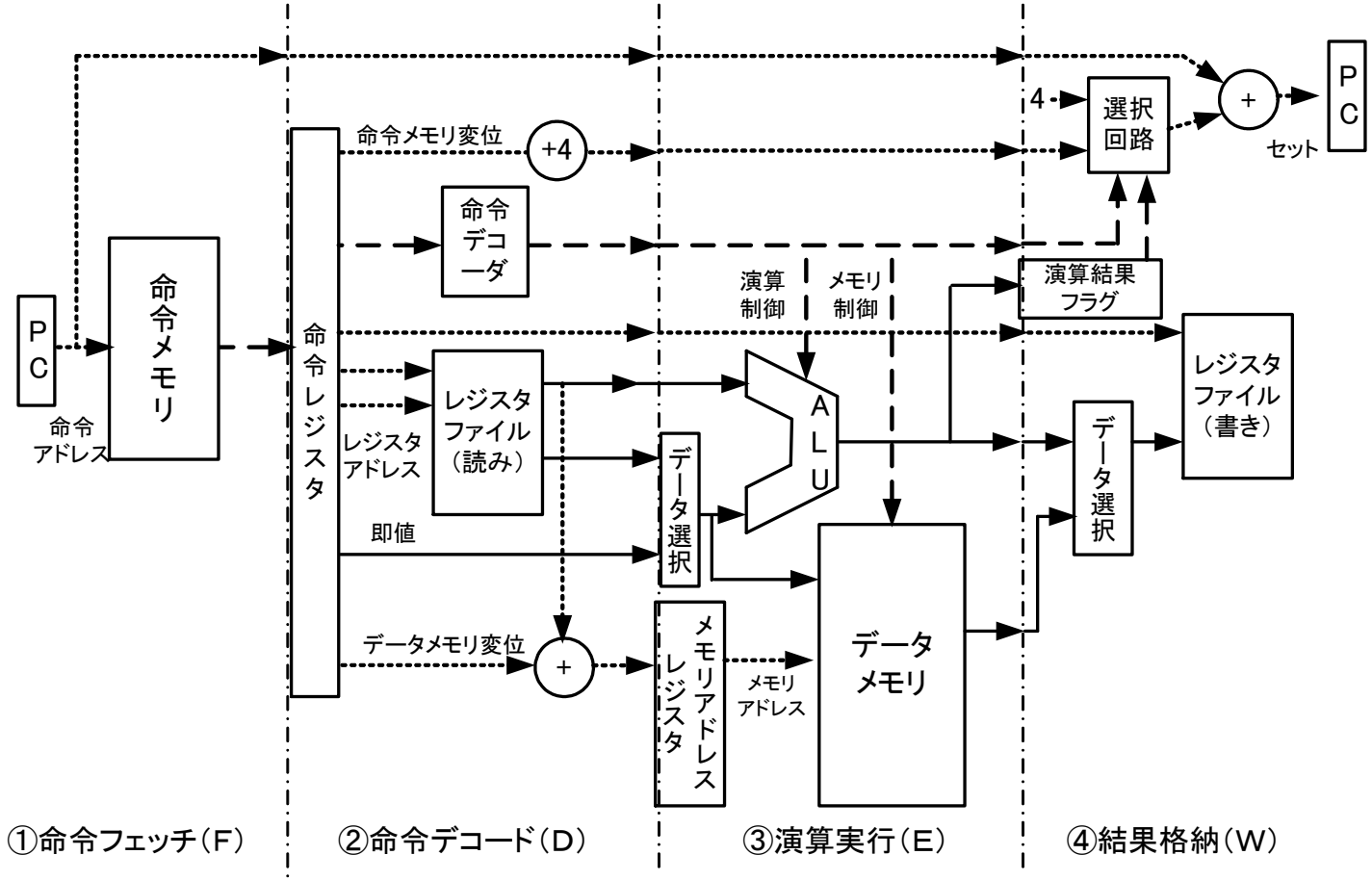


１ステージは１クロックで実行するものとする

F: 命令フェッチ、 D:命令デコード、 E:演算実行、 W:結果格納

図 4.2　基本命令パイプラインの理想的な動作

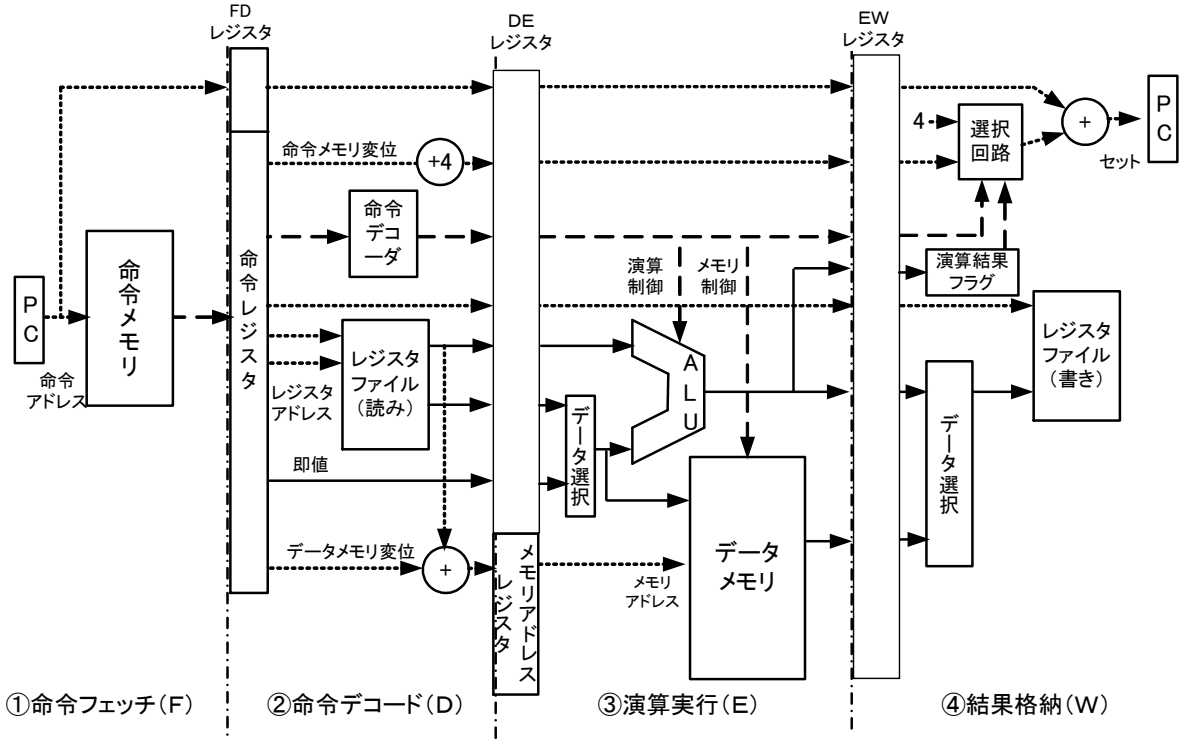
基本命令パイプラインの信号の流れ

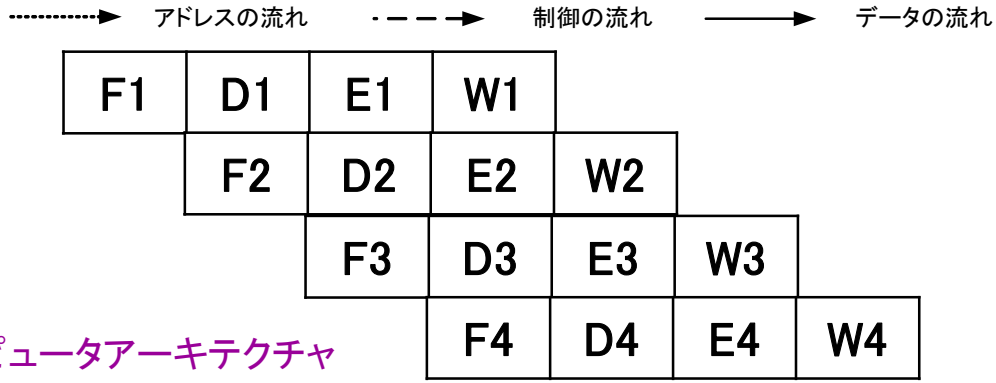


アドレスの流れ　制御の流れ　データの流れ

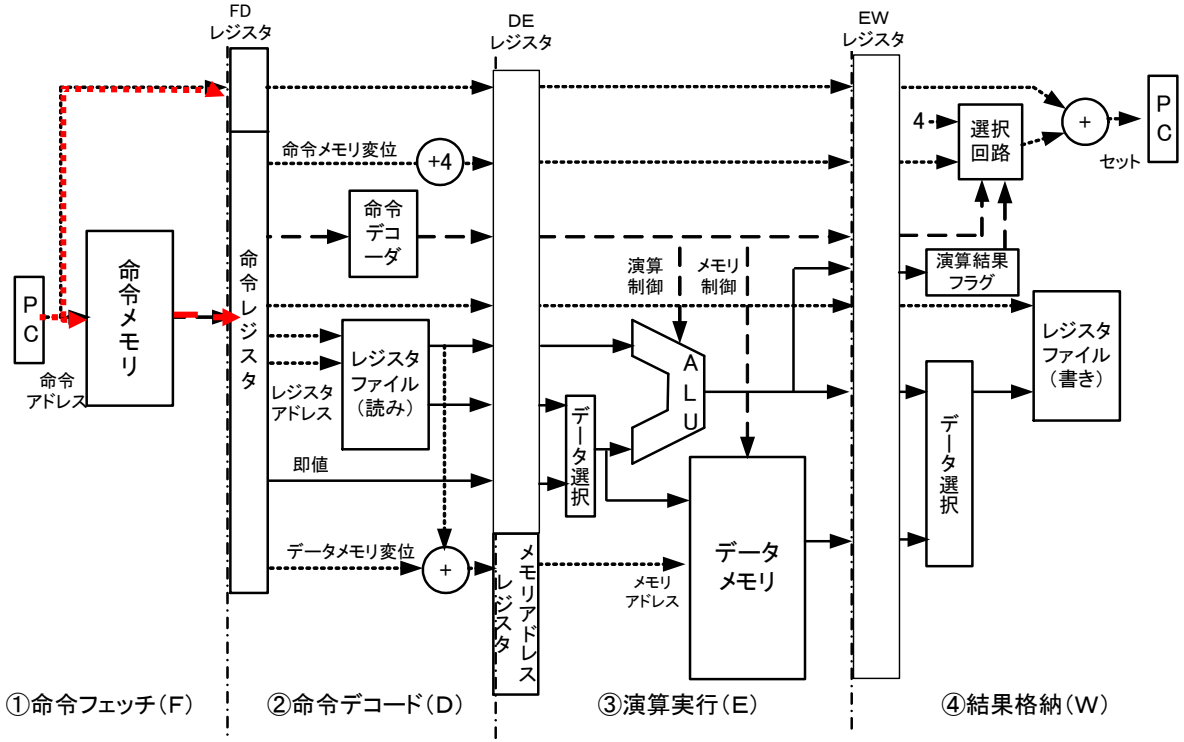
問題：　全ステージでデータが素通し

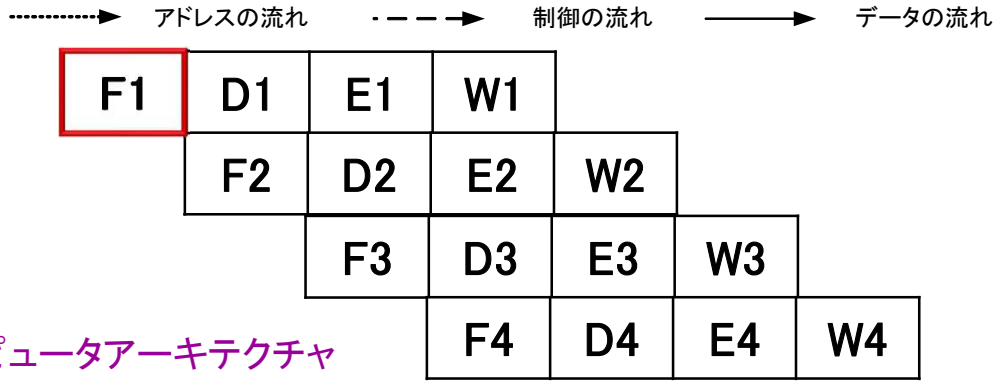
基本命令パイプライン


FD
レジスタ
DE
レジスタ
EW
レジスタ
PC
命令
アドレス
命令メモリ
命令レジスタ
命令メモリ変位
+4
命令
デコーダ
レジスタ
ファイル
（読み）
レジスタ
アドレス
即値
データメモリ変位
+
データ選択
ALU
演算
制御
メモリ
制御
メモリアドレスレジスタ
メモリ
アドレス
データ
メモリ
4
選択
回路
+
PC
セット
演算結果
フラグ
レジスタ
ファイル
（書き）
データ選択
①命令フェッチ（F）
②命令デコード（D）
③演算実行（E）
④結果格納（W）

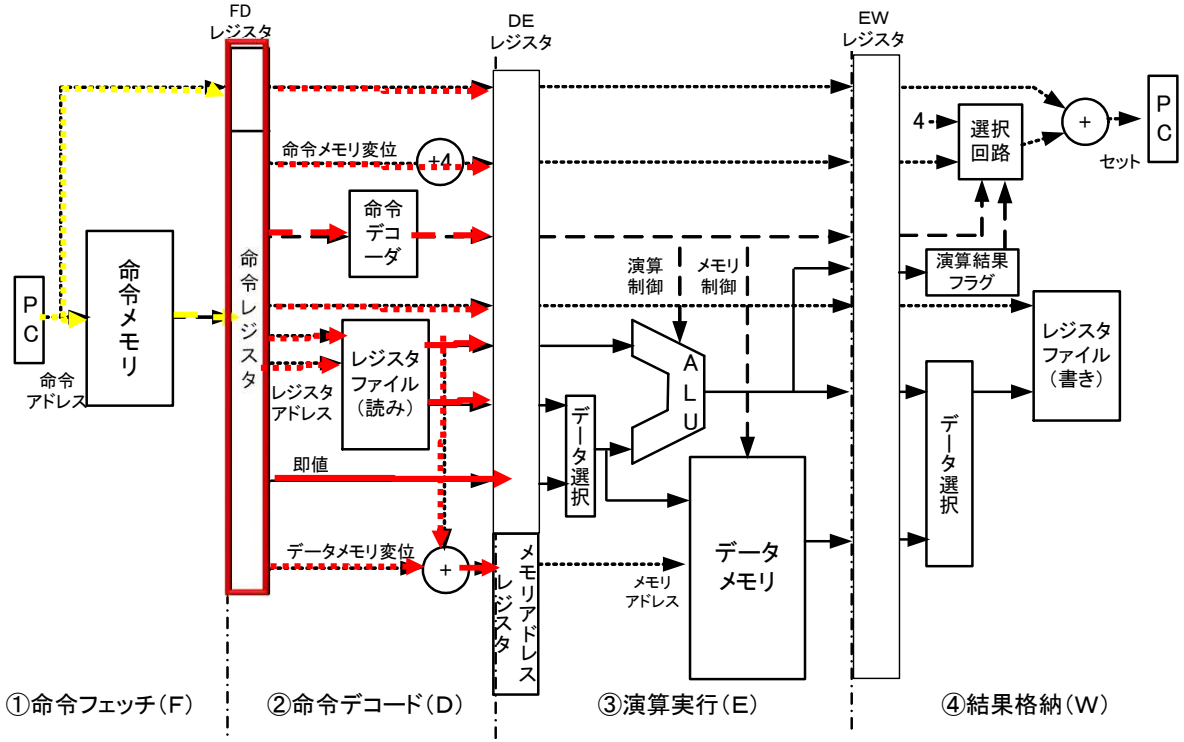

アドレスの流れ
制御の流れ
データの流れ
F1
D1
E1
W1
F2
D2
E2
W2
F3
D3
E3
W3
F4
D4
E4
W4

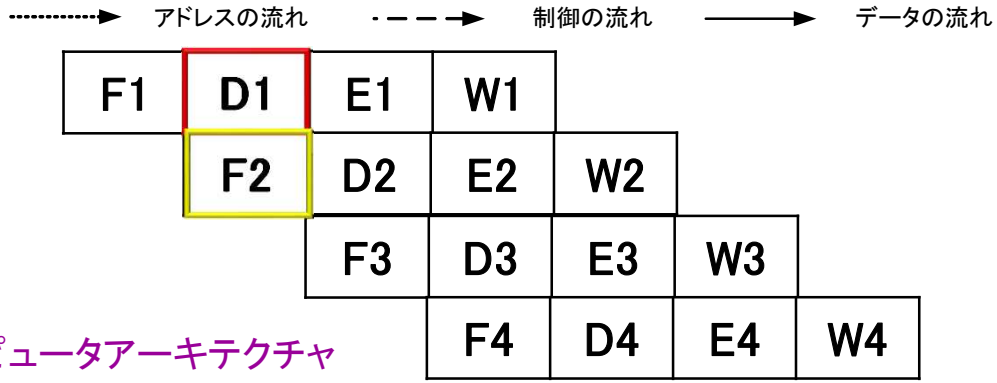
基本命令パイプライン


FD
レジスタ
DE
レジスタ
EW
レジスタ
命令メモリ変位
+4
命令
デコ
ーダ
命令
メモリ
PC
命令
アドレス
命令
レジ
スタ
レジスタ
ファイル
（読み）
レジスタ
アドレス
即値
データメモリ変位
+
演算
制御
メモリ
制御
ALU
データ
選択
メモリアドレス
レジスタ
データ
メモリ
メモリ
アドレス
4
選択
回路
+
PC
セット
演算結果
フラグ
レジスタ
ファイル
（書き）
データ
選択
①命令フェッチ（F）
②命令デコード（D）
③演算実行（E）
④結果格納（W）

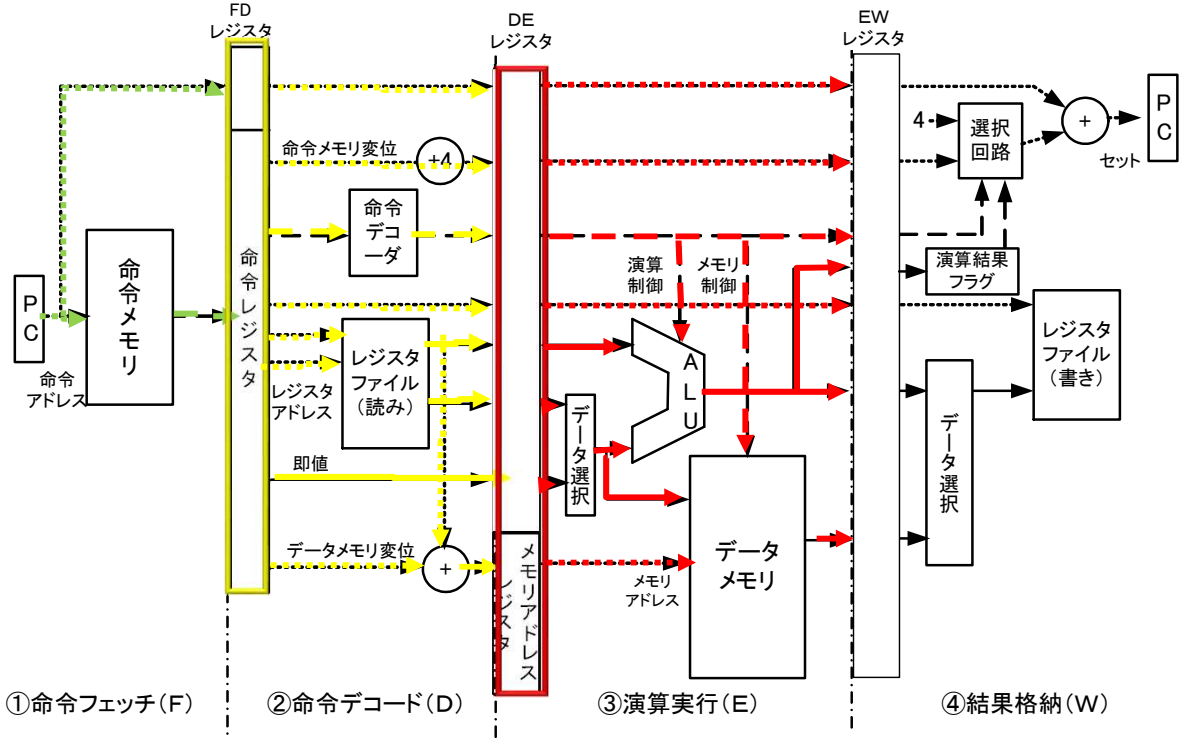

アドレスの流れ
制御の流れ
データの流れ
F1
D1
E1
W1
F2
D2
E2
W2
F3
D3
E3
W3
F4
D4
E4
W4

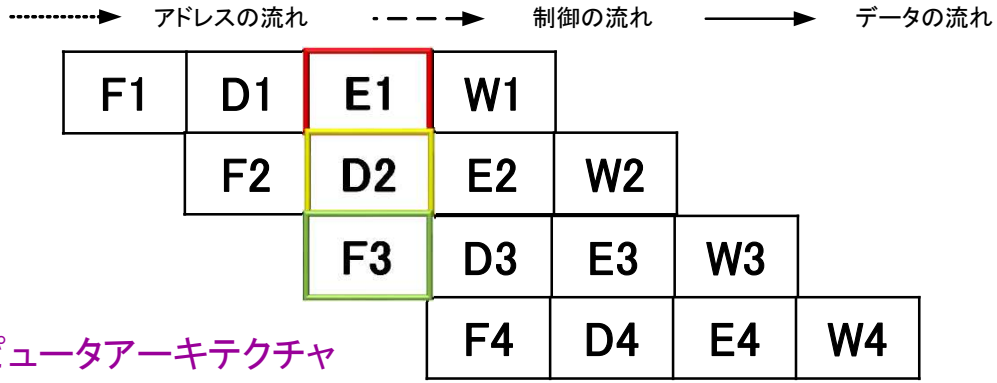
基本命令パイプライン


FD
レジスタ
DE
レジスタ
EW
レジスタ
命令メモリ変位
+4
命令
デコ
ーダ
命令
レジ
スタ
PC
命令
メモリ
命令
アドレス
レジスタ
ファイル
（読み）
レジスタ
アドレス
即値
データメモリ変位
演算
制御
メモリ
制御
ALU
データ
選択
メモリアドレス
レジスタ
メモリ
アドレス
データ
メモリ
4
選択
回路
PC
セット
演算結果
フラグ
レジスタ
ファイル
（書き）
データ
選択
①命令フェッチ（F）
②命令デコード（D）
③演算実行（E）
④結果格納（W）

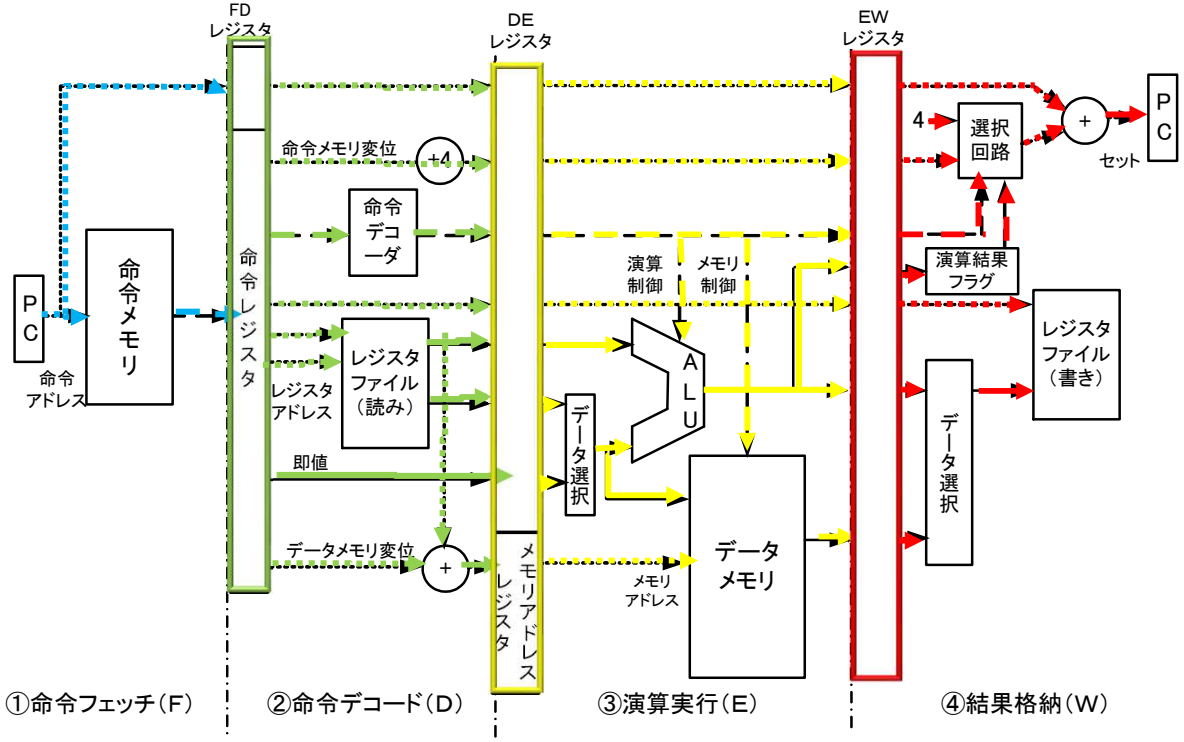

アドレスの流れ
制御の流れ
データの流れ
F1
D1
E1
W1
F2
D2
E2
W2
F3
D3
E3
W3
F4
D4
E4
W4

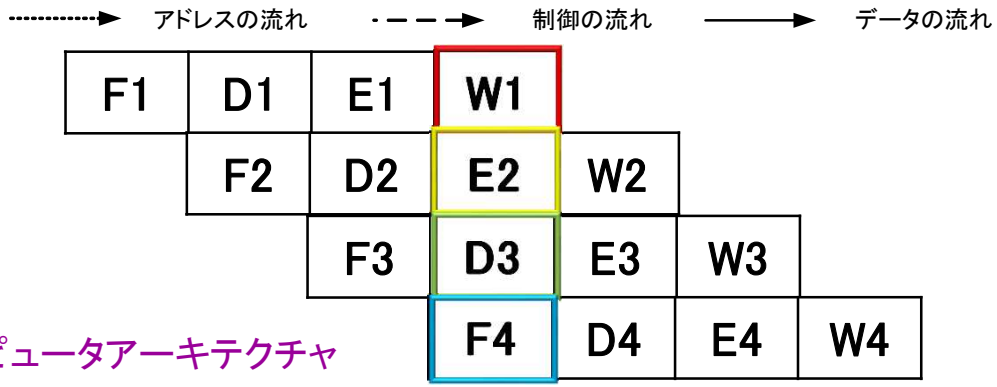
基本命令パイプライン


FD
レジスタ
DE
レジスタ
EW
レジスタ
命令メモリ変位
+4
命令
デコ
ーダ
命令
メモリ
PC
命令
アドレス
命令
レジスタ
レジスタ
ファイル
（読み）
レジスタ
アドレス
即値
データメモリ変位
演算
制御
メモリ
制御
ALU
データ
選択
メモリアドレス
レジスタ
データ
メモリ
メモリ
アドレス
4
選択
回路
PC
セット
演算結果
フラグ
レジスタ
ファイル
（書き）
データ
選択
①命令フェッチ（F）
②命令デコード（D）
③演算実行（E）
④結果格納（W）


アドレスの流れ
制御の流れ
データの流れ
F1
D1
E1
W1
F2
D2
E2
W2
F3
D3
E3
W3
F4
D4
E4
W4

基本命令パイプライン


FD
レジスタ
DE
レジスタ
EW
レジスタ
PC
命令
アドレス
命令メモリ
命令
レジスタ
命令メモリ変位
+4
命令
デコ
ーダ
レジスタ
ファイル
（読み）
レジスタ
アドレス
即値
データメモリ変位
+
メモリアドレス
レジスタ
データ
選択
演算
制御
メモリ
制御
ALU
データ
メモリ
メモリ
アドレス
4
選択
回路
+
PC
セット
演算結果
フラグ
レジスタ
ファイル
（書き）
データ
選択
①命令フェッチ（F）
②命令デコード（D）
③演算実行（E）
④結果格納（W）


アドレスの流れ
制御の流れ
データの流れ
F1
D1
E1
W1
F2
D2
E2
W2
F3
D3
E3
W3
F4
D4
E4
W4

パイプラインの阻害要因

- オーバヘッド

  本来の処理では存在しなかった余計な時間のこと

  - 最も時間のかかるステージの処理時間で全体のスループットが決まる（律速ステージ）
    - 対策： できるだけ各ステージの処理時間をあわせる
  - パイプラインレジスタによる遅延
    - 対策： 高速なレジスタを使う ⇒ 限界あり
- ハザード

  クロックごとにパイプライン動作させられない状態のこと

  - 構造ハザード
  - データハザード
  - 制御ハザード
- ストール

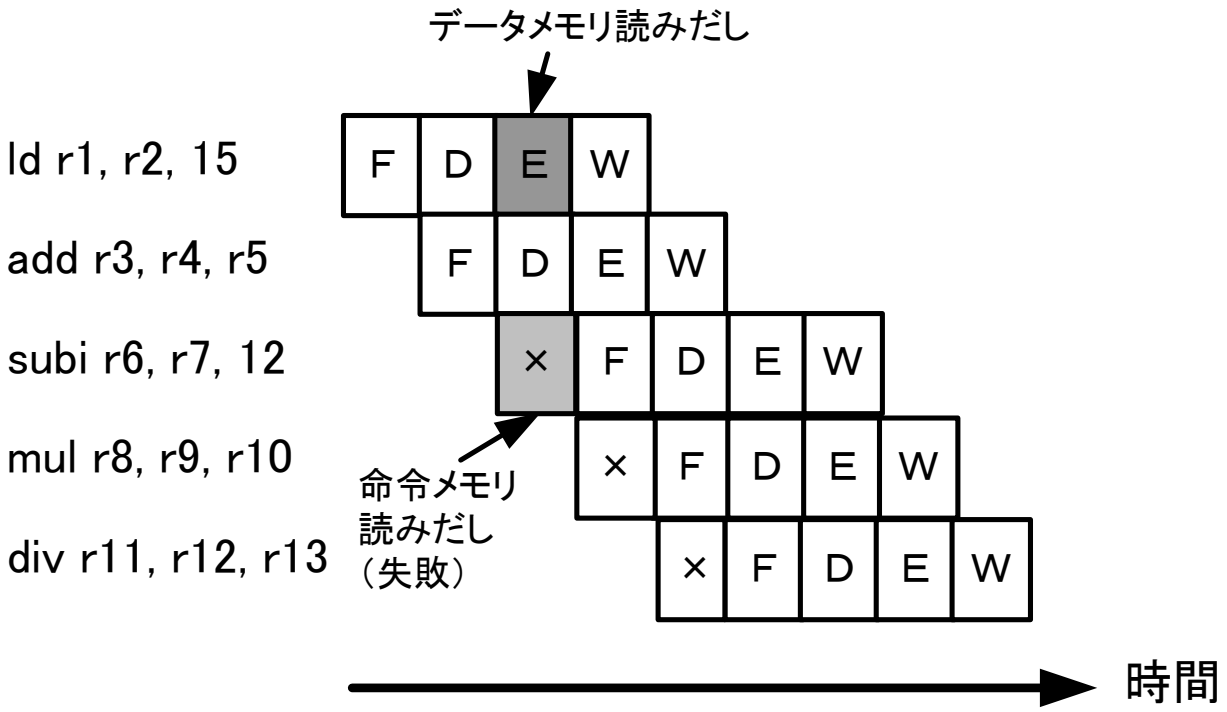
  ハザードによって命令の実行が止められる状態のこと

構造ハザード

- 構造ハザード

  コンピュータの内部構成が原因のハザードのこと

  - あるステージを実行中の命令Aと別のステージを実行中の命令Bが、同じハードウェア資源を使わなければならない場合などに生じる



×の箇所で、メモリアクセスが衝突するので構造ハザードが起こる。

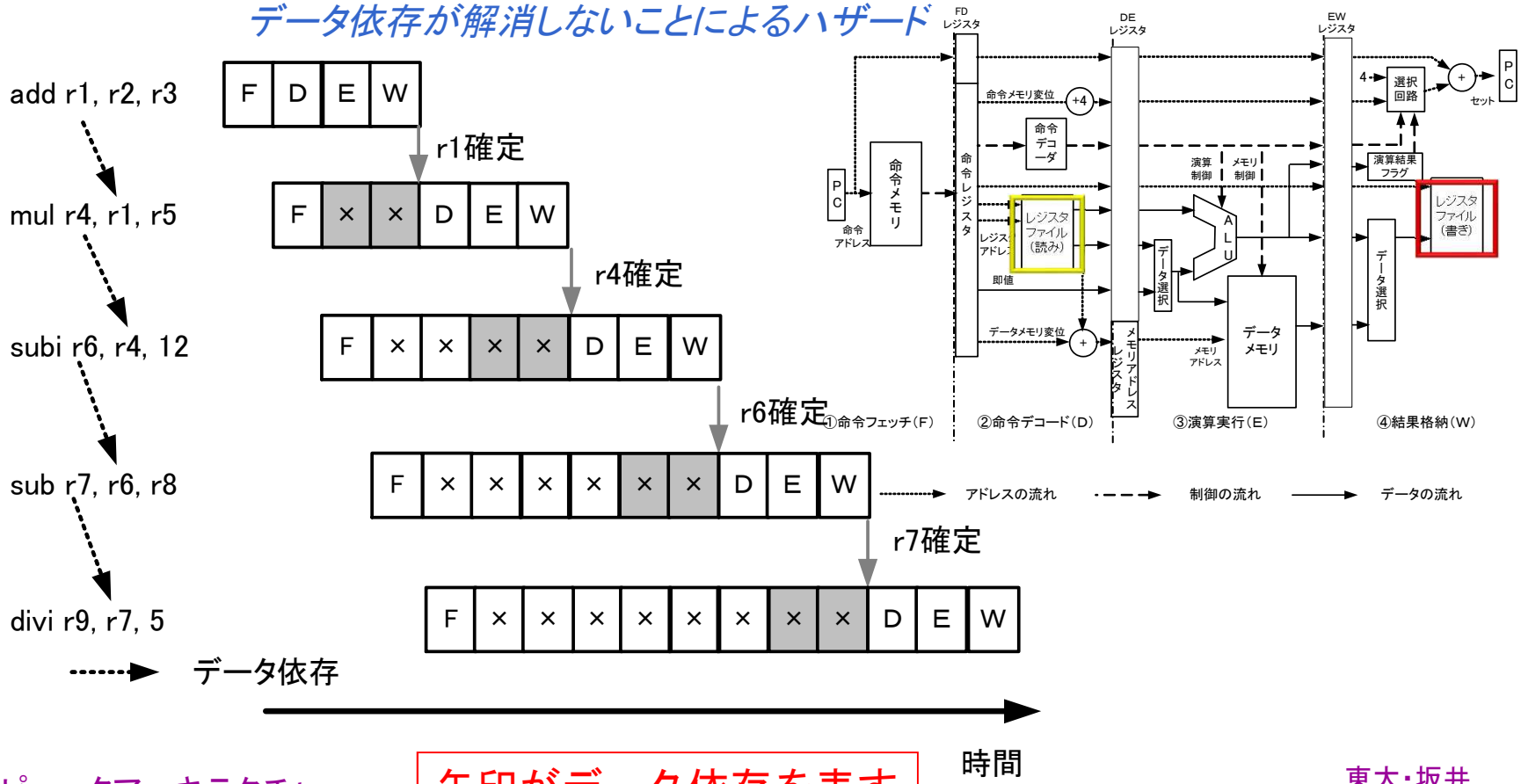
データハザード

- データ依存

  命令Aがデータを生成し、これを命令Bが使うとき、
  BはAにデータ依存であるという

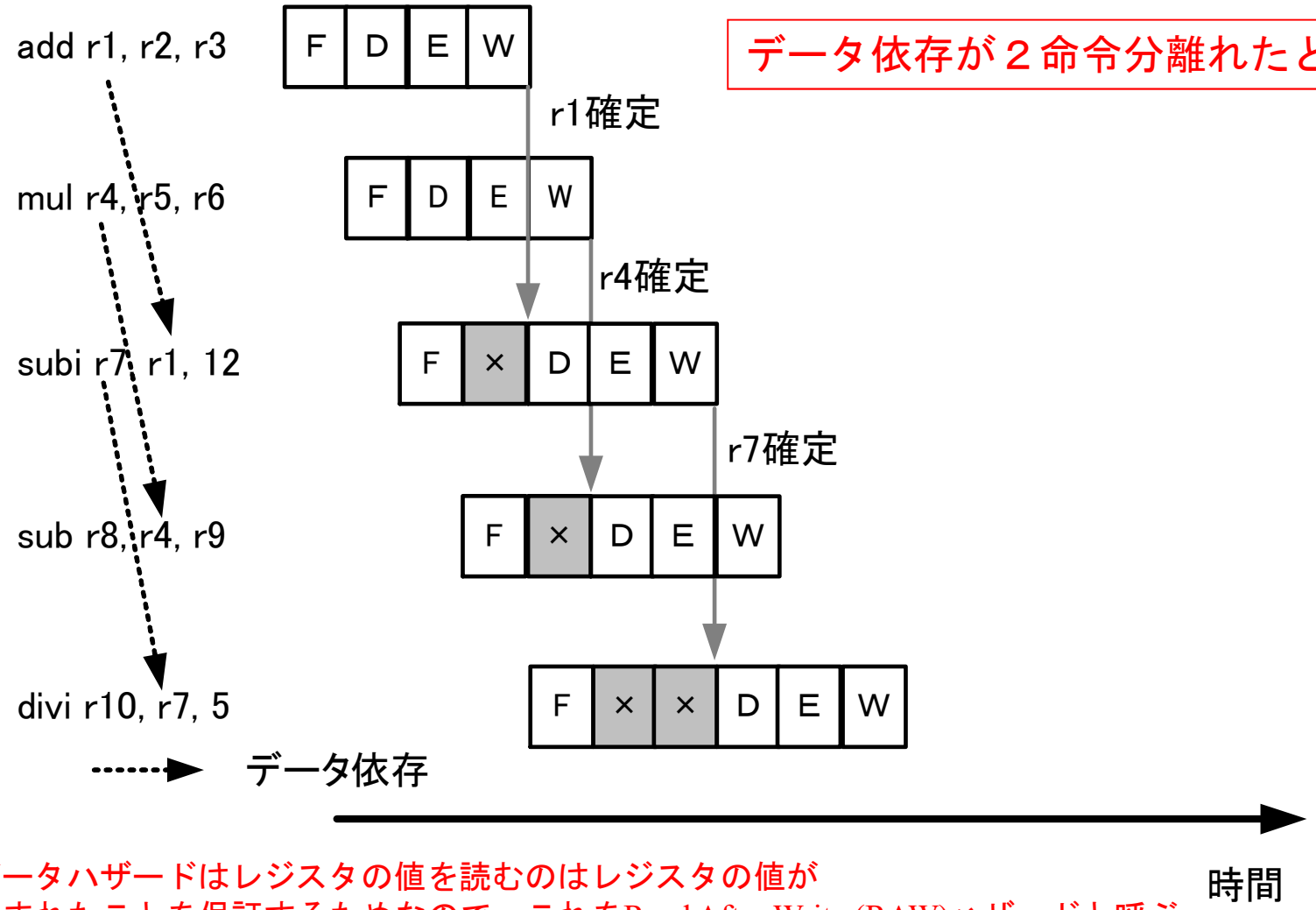
- データハザード

  データ依存が解消しないことによるハザード



矢印がデータ依存を表す

データハザード（続）



データ依存が２命令分離れたとき

メモ：このデータハザードはレジスタの値を読むのはレジスタの値が
書き込まれたことを保証するためなので、これをRead After Write (RAW)ハザードと呼ぶ
RAWハザード以外に、WARハザード、WAWハザードというものもあるが、それは後ほど説明する

図4.6,4.7： パイプラインレジスタのストールを考えた図

add r1,r2,r3	F	D	E	W												
mul r4,r1,r5		F	F	F	D	E	W									
subi r6,r4,12					F	F	F	D	E	W						
sub r7,r6,r8								F	F	F	D	E	W			
divi r9,r7,5											F	F	F	D	E	W

図4.6

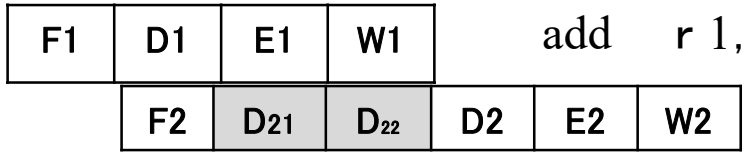
add r1,r2,r3	F	D	E	W						
mul r4,r5,r6		F	D	E	W					
subi r7,r1,12			F	F	D	E	W			
sub r8,r4,r9					F	D	E	W		
divi r10,r7,r5						F	F	D	E	W

図4.7

@Tishiyuki Nakata

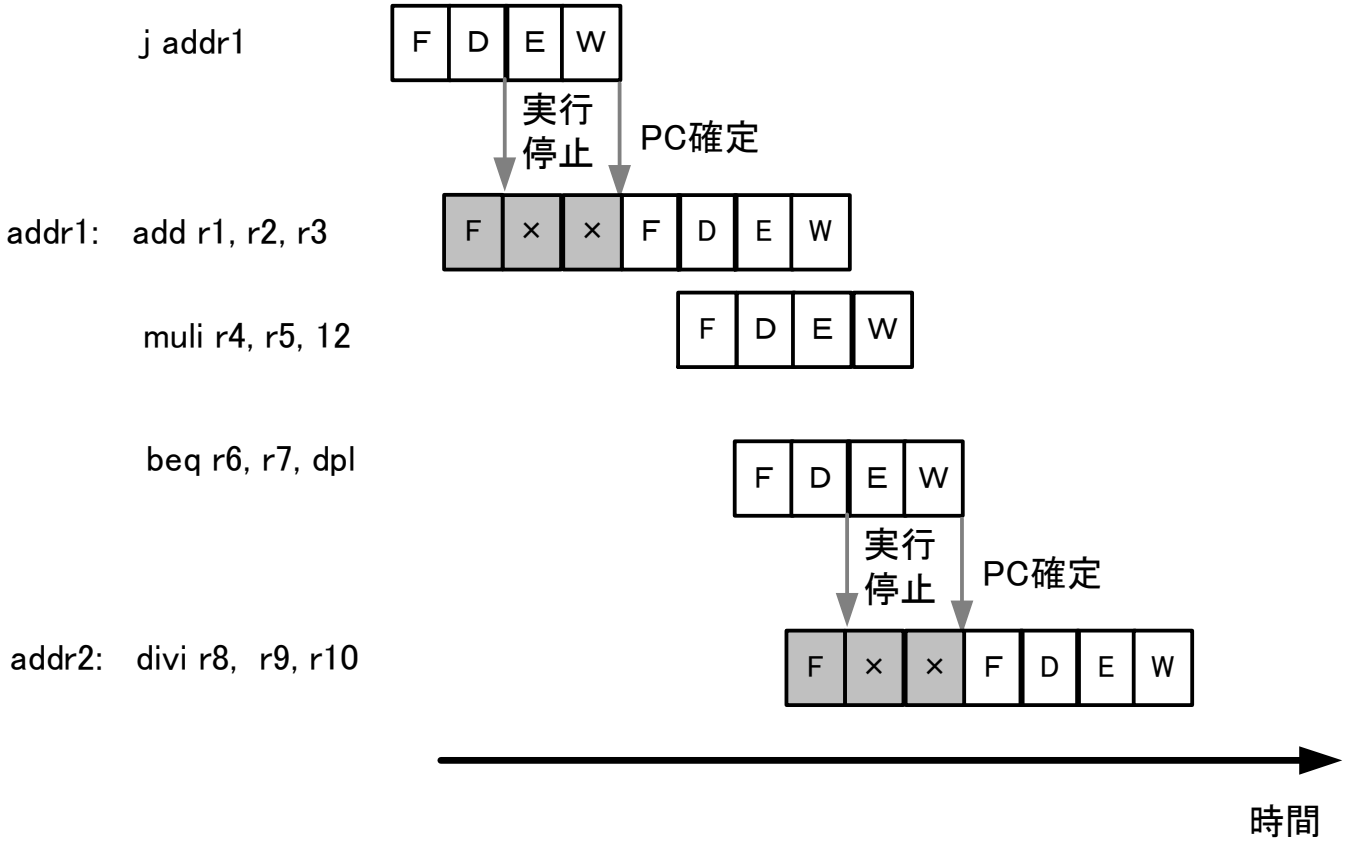
メモ:ストールの実現方法

- データハザードの場合のストールの方法はどういう手法があるのであろう?
- 実装方法
  - Valid bit方式:
    - D1時にFDレジスタのrdレジスタに
      無効ビットを立てる。W1時に無効ビットを外す。
    - 無効ビットが立っている間はDE命令レジスタにNOPを入れる。PCの更新とFDレジスタの更新を禁止する。
  - フィールドデコード方式
    - DE命令レジスタかEW命令レジスタのrdの値がFD命令レジスタのrs,rtの値と一致
- 対応方法:ED命令レジスタの値をNOPに変更。PCの値とFDレジスタの値を保持



制御ハザード

- 制御依存
  制御命令とそれ以後の命令の依存
- 制御ハザード
  制御依存によるハザード

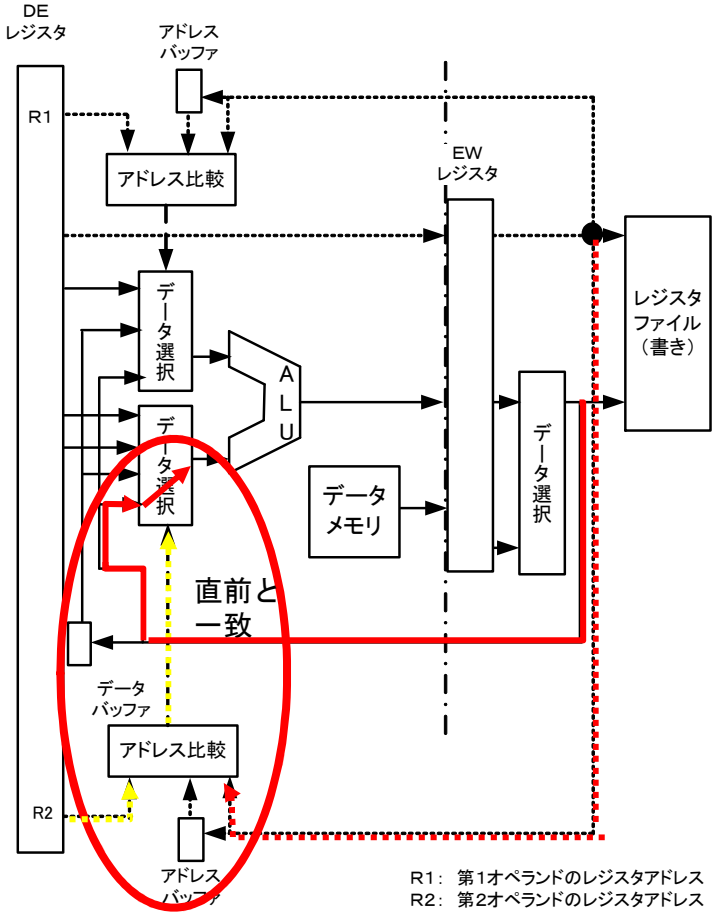


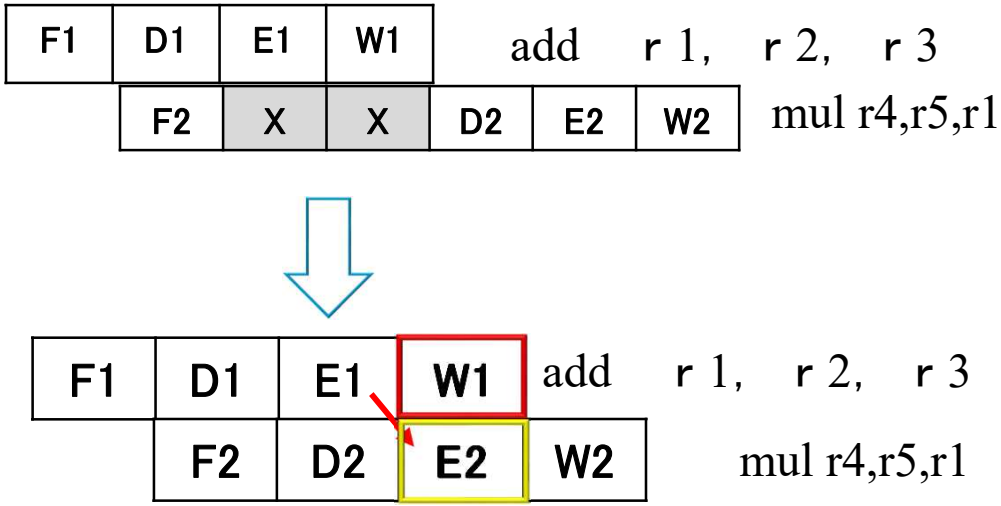
フォワーディング（1）

- フォワーディング　（＝バイパシング、ショートカット）

  Eステージの結果を、Wステージを経ることなく、直接に後続の命令のEステージに送り込むこと

  – データハザードを解決する！



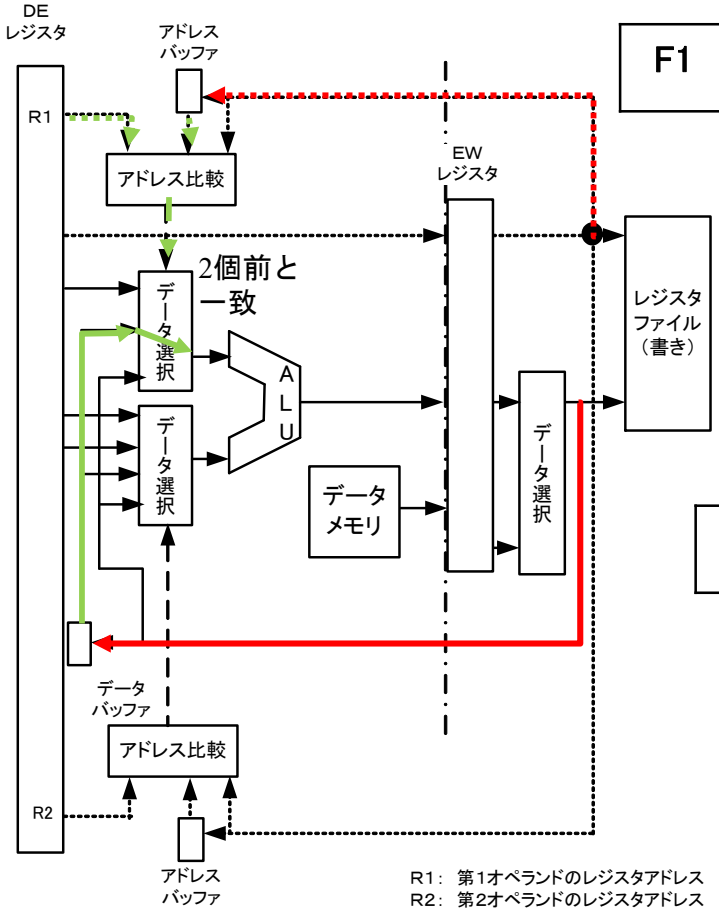


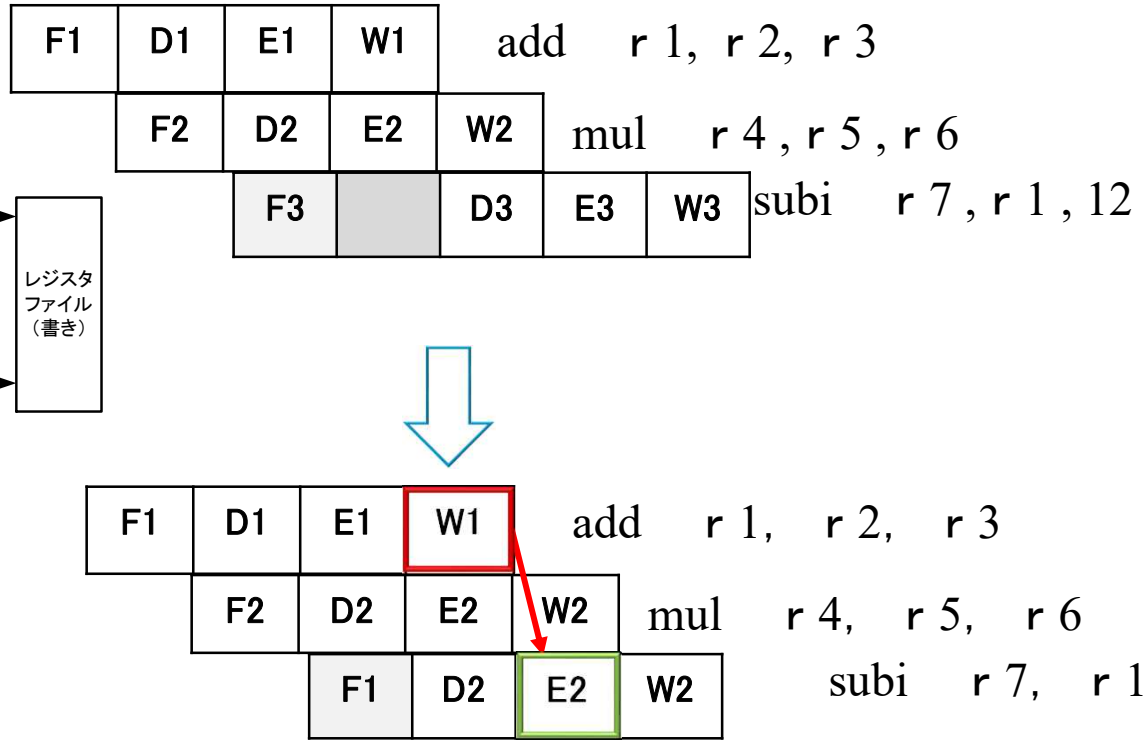
フォワーディング（2）

- フォワーディング　（＝バイパシング、ショートカット）

  Eステージの結果を、Wステージを経ることなく、直接に後続の命令のEステージに送り込むこと

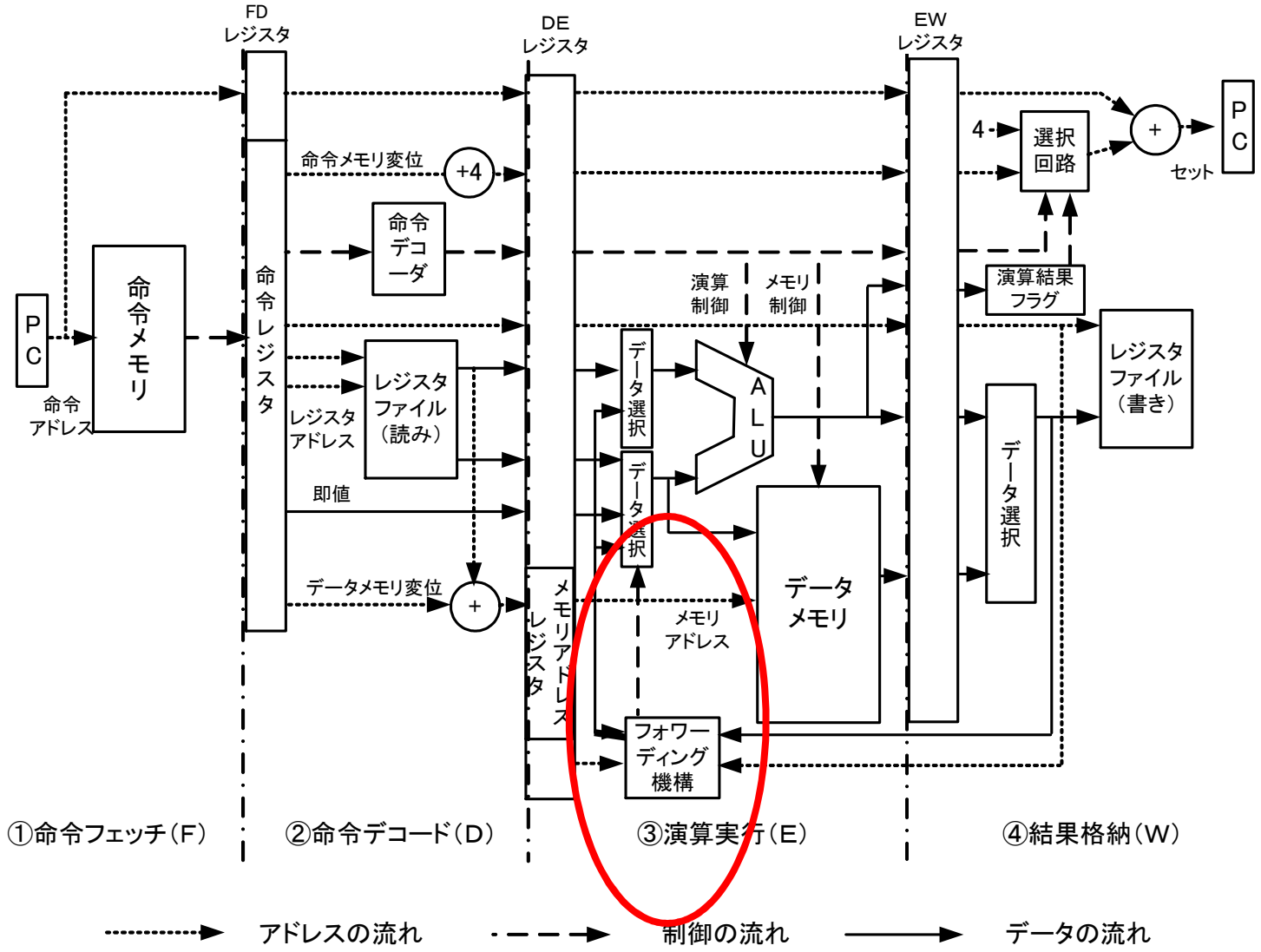
  – データハザードを解決する！





アドレスの流れ　制御の流れ　データの流れ

フォワーディング機構の入ったパイプライン

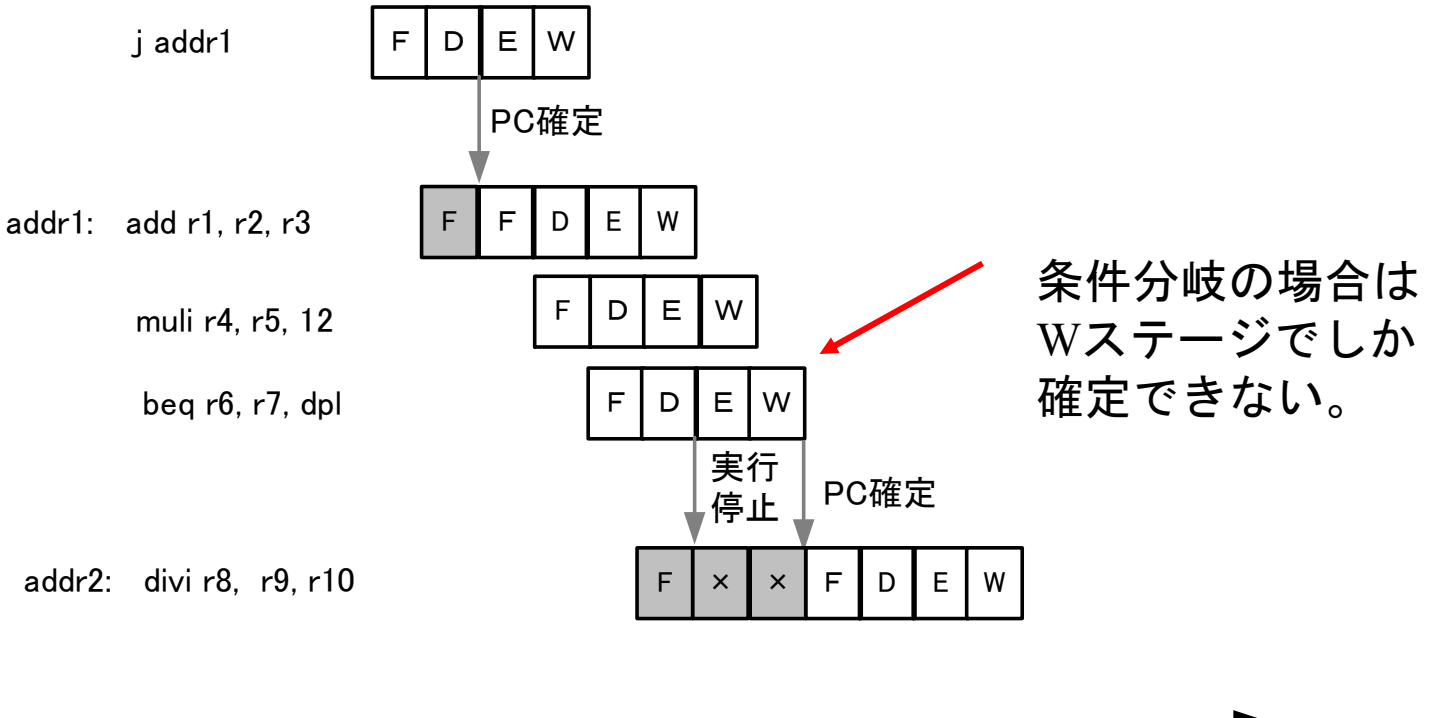


命令アドレス生成を早める

- 無条件分岐命令のうちjump addr のA形式の命令の場合のアドレス生成タイミング

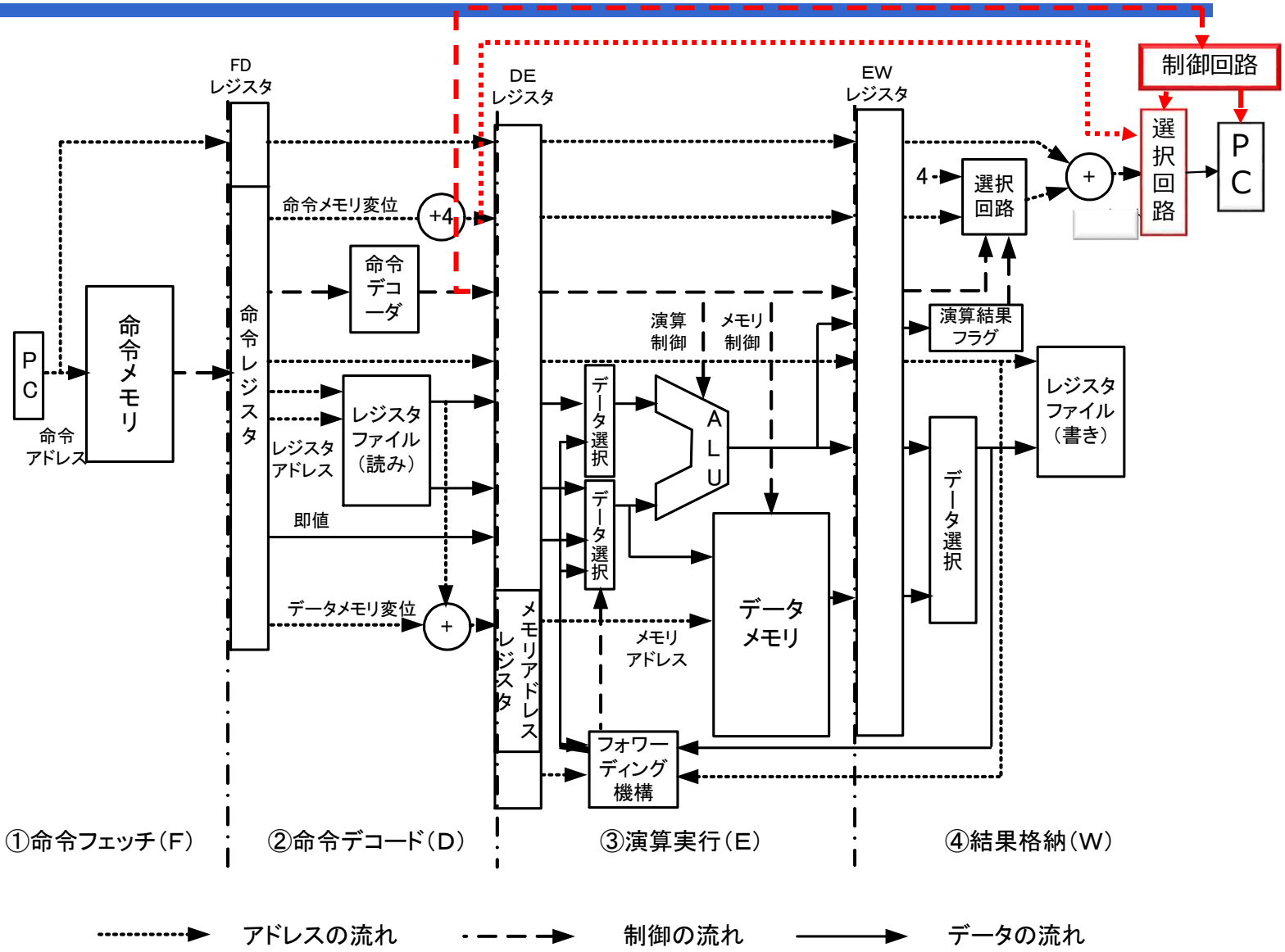
  Wステージではなく、Dステージで次の命令アドレスを確定させる

  - 制御ハザードを緩和する！



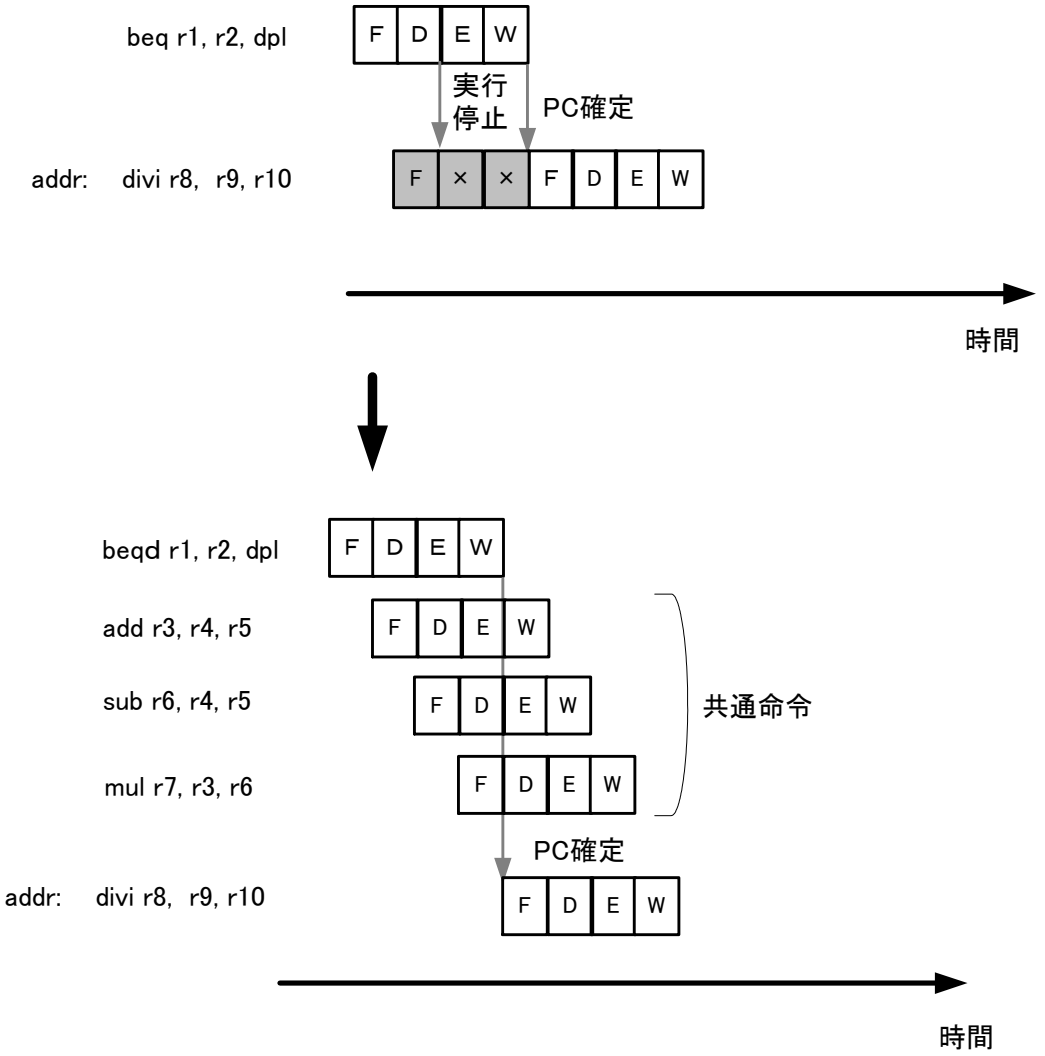
メモ：jmp addr命令を高速化するための付加回路

jmp addrのaddrの値が実際より4少ない値と仮定



遅延分岐

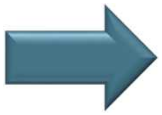
- 遅延分岐
  - (1)分岐のあるなしにかかわりなく実行する命令(共通命令)を分岐命令の次のアドレス(遅延スロット、deyaled slot)に入れておく
  - (2)遅延分岐命令は、定められた数の共通命令をパイプライン実行したあとでPCをセットする
  - ⇒ 制御ハザードをなくす



メモ：遅延分岐用共通命令

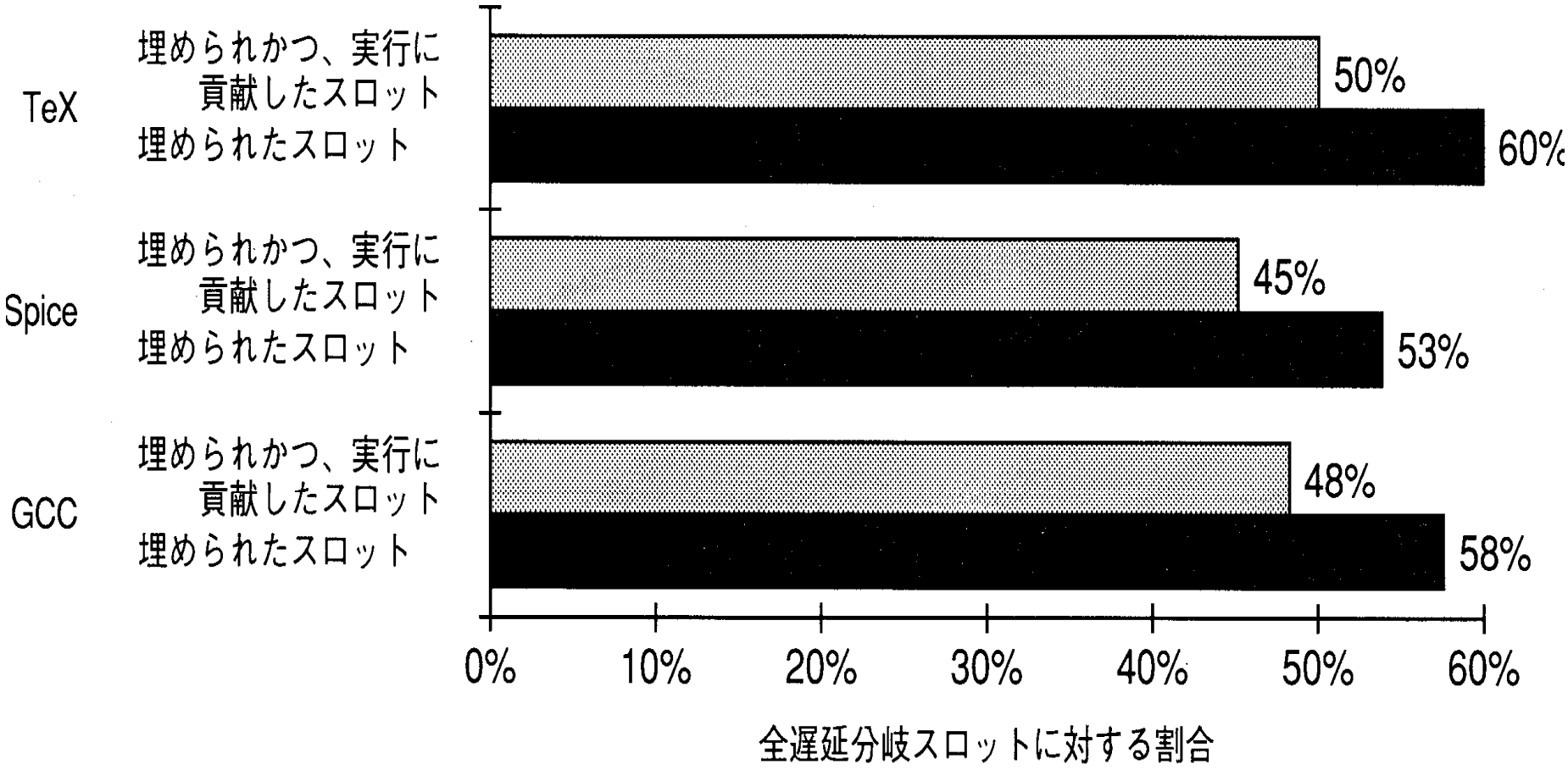
- そんなにたくさん共通命令があるかという疑問があるかもしれない。実はコンパイラが遅延分岐を意識していたら、比較的容易に提供可能

```
add r3,r4,r5
sub r6,r4,r5
mul r7,r3,r6
beqd r1,r2,dpl

addr divir8,r9,r10
```



```
beqd r1,r2,dpl
add r3,r4,r5
sub r6,r4,r5
mul r7,r3,r6

addr divir8,r9,r10
```

遅延スロットの充足率

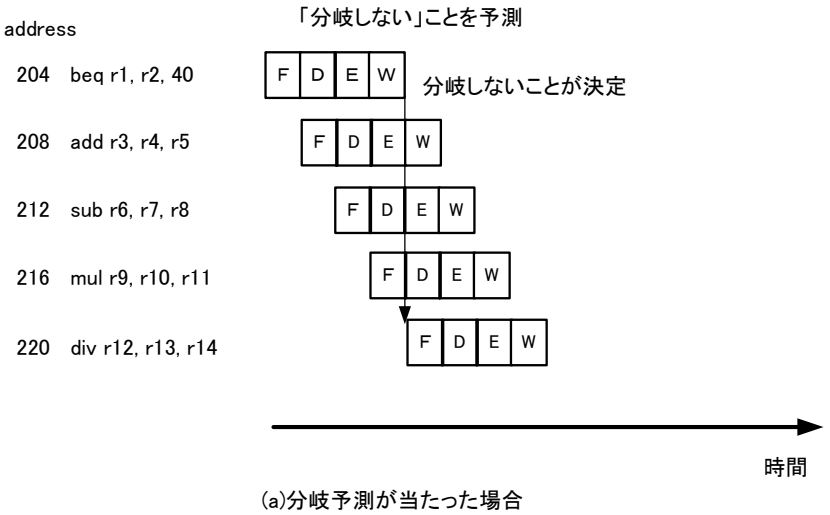

TeX
Spice
GCC
埋められかつ、実行に貢献したスロット
埋められたスロット
50%
60%
45%
53%
48%
58%
0%
10%
20%
30%
40%
50%
60%
全遅延分岐スロットに対する割合

分岐予測

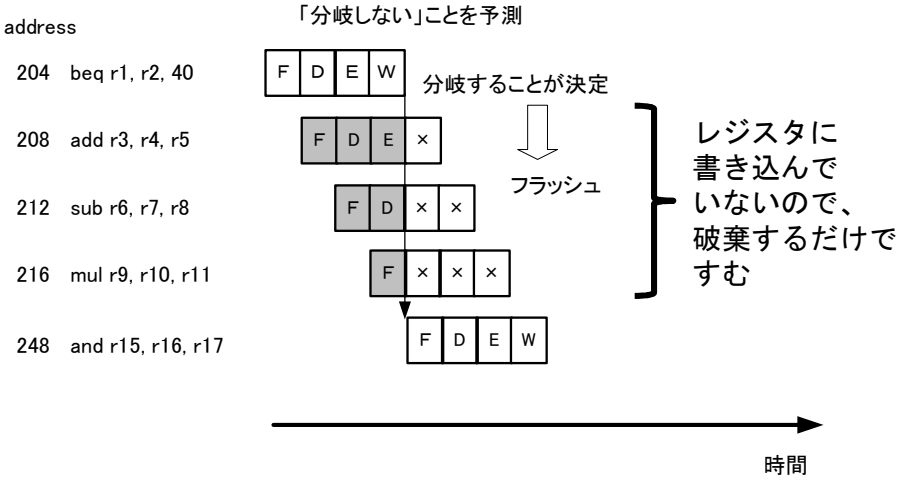
- 分岐予測

分岐が起こるかどうかを予測して処理を進め、予測がはずれた場合に分岐命令以下の命令を破棄する

- 制御ハザードを緩和する！



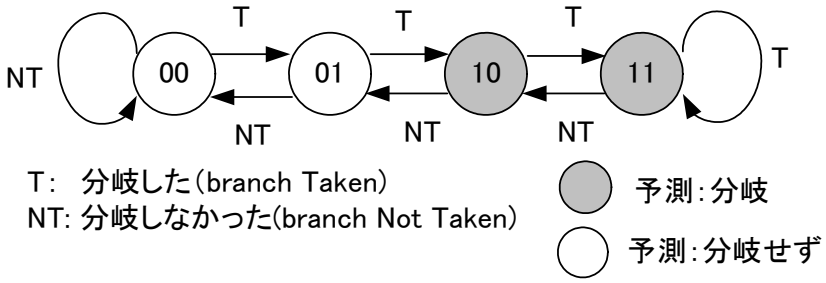
(a)分岐予測が当たった場合

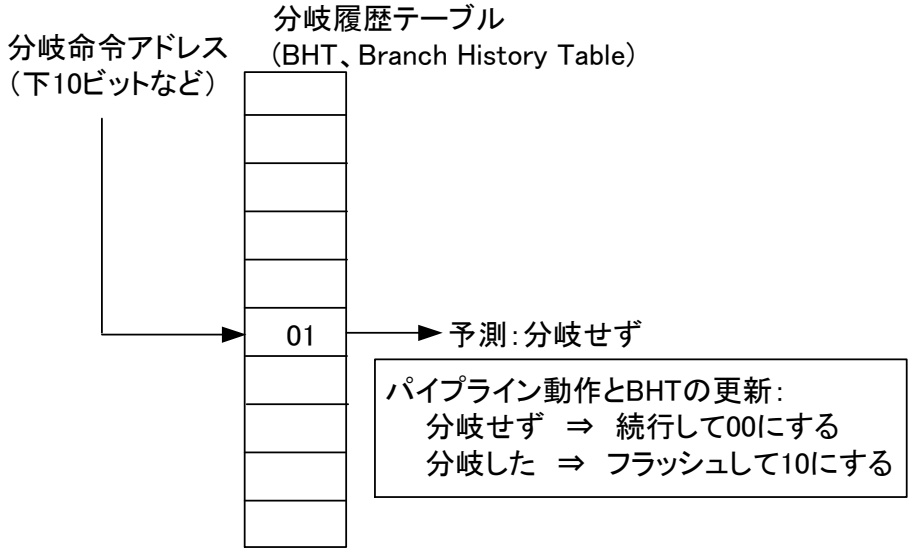


(b)分岐予測がはずれた場合

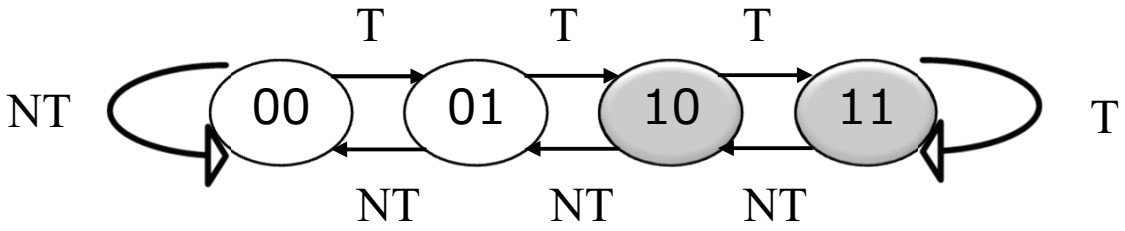
分岐予測の方式

- 静的な方式
  - 常に分岐する(しない)ほうを予測する
  - 命令アドレスが小さくなるほうを予測する
- 動的な方式
  - 2ビット予測器(右図)
  - 2レベル適応型予測器





2ビット予測器の動き例



状態値	動き	予測
01	分岐した	外れ
10	分岐した	あたり
11	分岐した	あたり
11	分岐した	あたり
11	分岐した	あたり
11	分岐しなかった	外れ
10	分岐しなかった	はずれ
01	分岐しなかった	あたり
00		

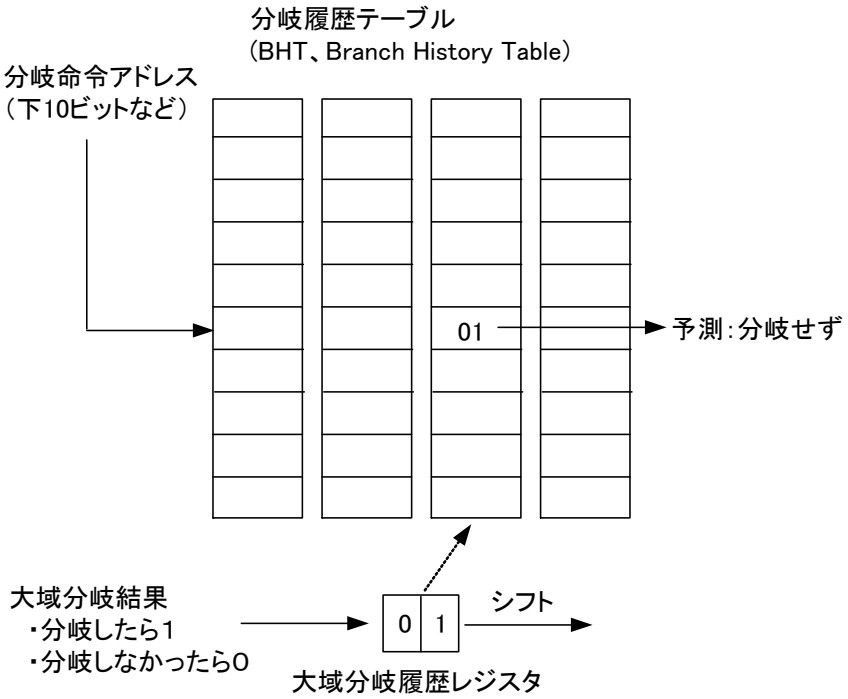
For loopなんかは殆ど11になる。

2レベル適応型予測器

- 2レベル適応型予測器

  各分岐命令ごとの履歴と、すべての分岐の履歴の両方を使って精度の高い分岐予測を実現する

  ⇒　平均成功率90%超



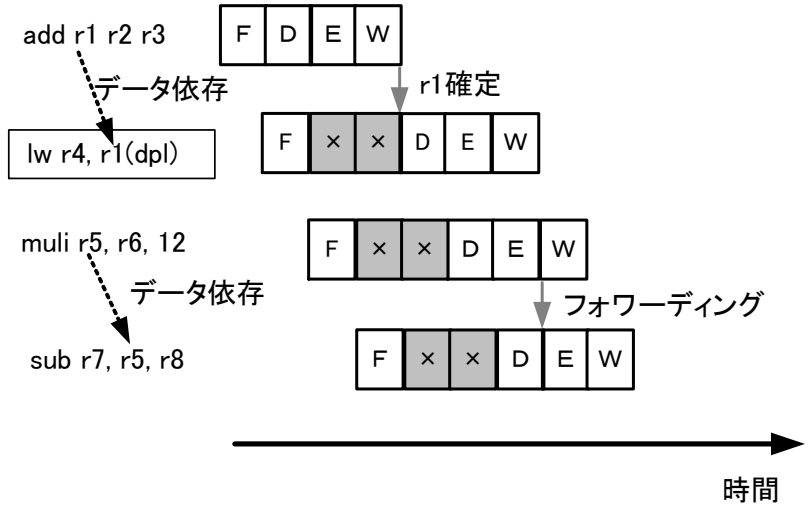
BHT値	分岐結果	パイプライン動作	BHT更新	大域分岐履歴レジスタ値
01(分岐せず)	分岐せず	続行	00	00
01	分岐	フラッシュ	10	10

命令スケジューリング

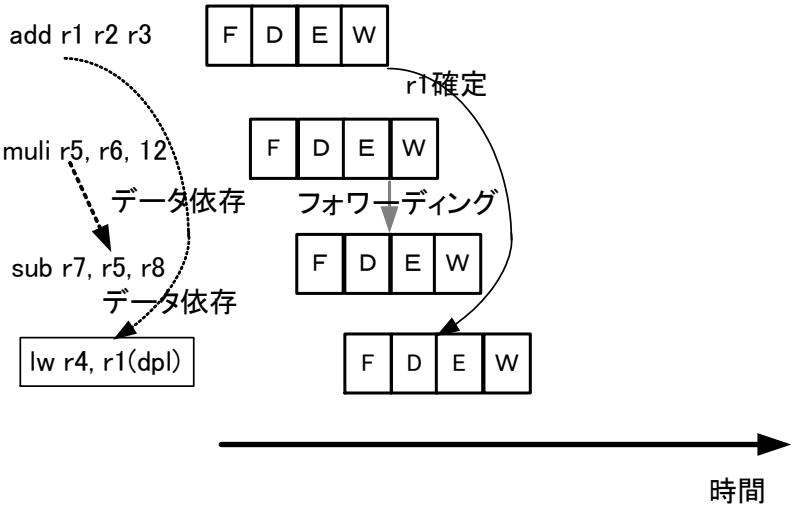
- 命令スケジューリング

  命令の位置を最適化するスケジューリング

  - 依存関係のある命令をプログラムの中でできるだけ離した位置に置く
  - 制御ハザードの防止のために、遅延分岐を使い、遅延スロットに共通命令を充填する
  - ループの本体を大きくするなどして、分岐命令の生起間隔を大きくする
- 構造ハザード、制御ハザードについてはこの効果が大きい。
- データハザードについては、必要ない（例外：メモリアクセス）



(a)メモリアドレス生成のデータ・ハザード



(b)命令スケジューリングによるハザードの解消