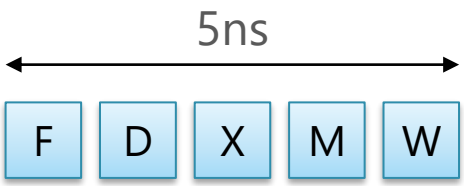
コンピュータ アーキテクチャ Ⅰ　第８回

塩谷 亮太 (shioya@ci.i.u-tokyo.ac.jp)
東京大学大学院情報理工学系研究科 創造情報学専攻

課題の解説

課題 7

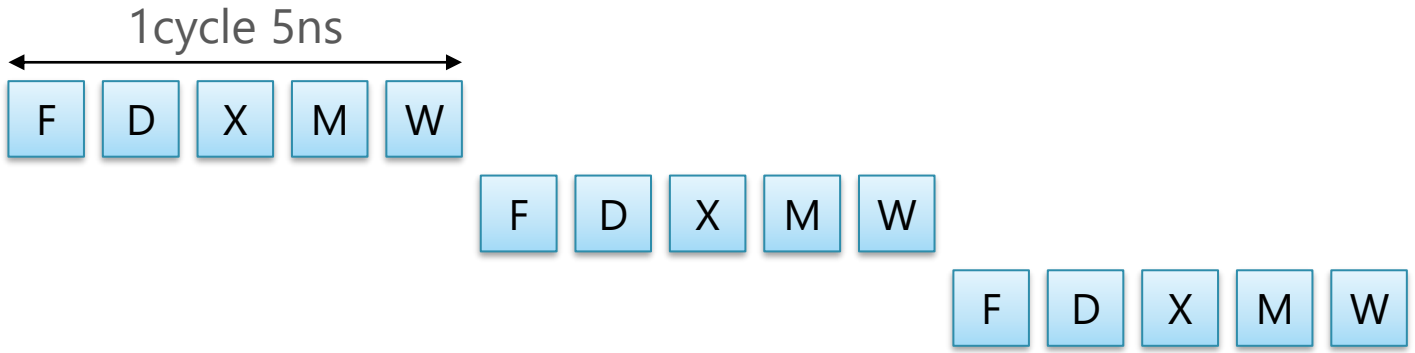
- 前提：
  - 各命令は以下の５つの処理を経て実行されるものとする
    - F：フェッチ，D：デコード，X：演算，
      M：メモリアクセス，W：書き戻し
  - これらの各処理には 1 nano second (ns) がかかるものとする
  - 演算しか行わずメモリアクセスを伴わない命令でも，必ず M を経るものとする
  - フォワーディングは常に行うものとする
  - 命令を実行するために「必要な時間」とは，最初の命令のフェッチ開始から最後の命令の書き戻しが終わるまでの時間とする
    - たとえば以下の１命令を実行するために「必要な時間」は 5ns



課題 7

- (1) その場合の最大動作周波数はいくつになるか述べよ
  - 1 秒 / 5ns = 200MHz
  - 各命令の処理は 5ns かかり，それを 1 サイクルで処理するため
- (2) 以下の命令列を実行するのに必要な時間を計算せよ
  - 5 ns × 3 命令 = 15ns

```
sub  x2←x3+x4
add  x1←x2+x3
add  x5←x6+x7
```



課題 7

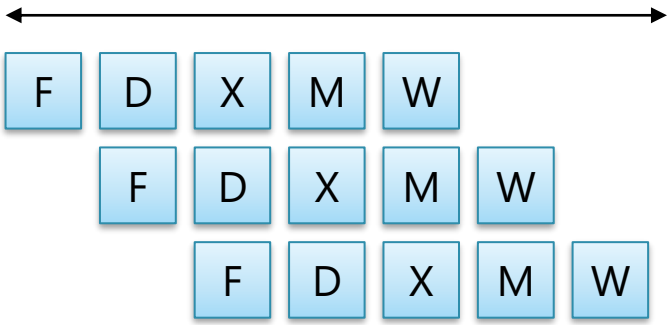
- 前記の前提の下で，F,D,X,M,W の 5 ステージからなるパイプライン・プロセッサを構成することを考える
  ただし，in-order スカラ・プロセッサであるものとする

- (3) その場合の最大動作周波数はいくつになるか述べよ
  - 1 秒 / 1 ns = 1GHz

- (4) 以下の命令列を実行するのに必要な時間を計算せよ

  sub x2←x3+x4
  add x1←x2+x3
  add x5←x6+x7

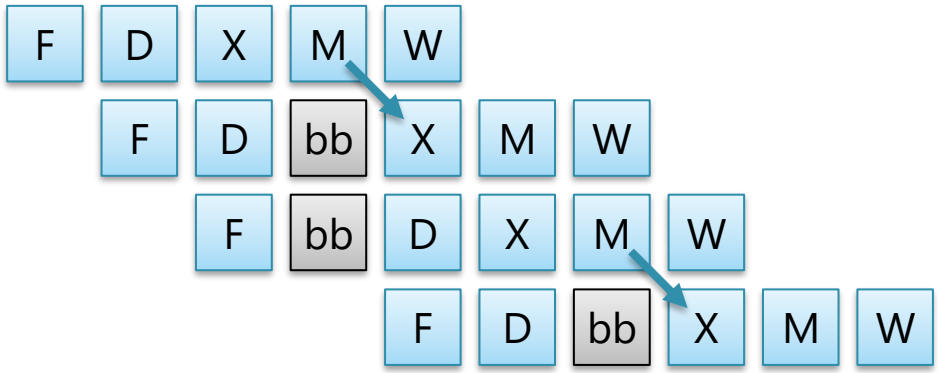
  - 7ns



課題 7

- (5) 以下の命令列を実行するのに必要な時間を計算せよ
  依存関係のために必要な場合のみパイプラインを適宜ストールして実行するものとせよ
  ld　x2←(x1)
  add x1←x2+x3
  ld　x2←(x3)
  add x5←x2+x7

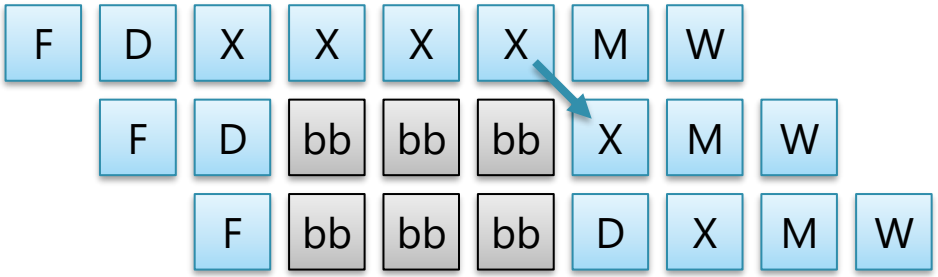
- 10ns（フォワーディングありの場合）
  1,3 命令目の ld では M が終わるまで値が取れないため,
  それに依存した後ろにある命令ではバブルが生じる（ストールが発生する）



課題 7

- (6) 以下の命令列を実行するのに必要な時間を計算せよ
  ここで mul は乗算命令であり X に４サイクルが必要である.
  依存関係を満たすために必要な場合のみ
  パイプラインを適宜ストールして実行するものとせよ
  mul x2←x1+x4
  add x1←x2+x3
  add x5←x2+x7

- 10ns （フォワーディングありの場合）

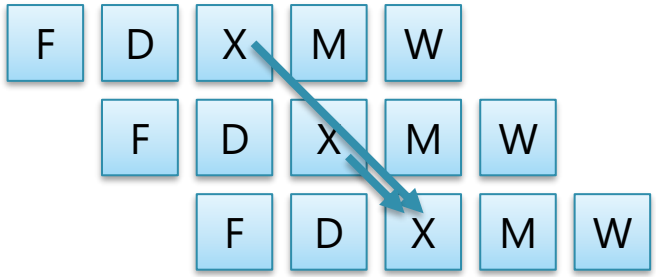


課題 7

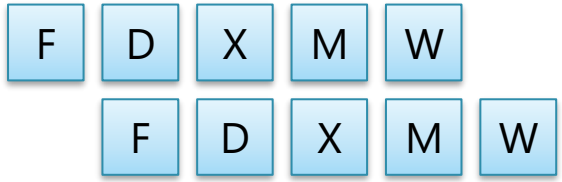
- (7) 以下の命令列を実行するのに必要な時間を計算せよ
  ここで beq はオペランドが等しい時に分岐する分岐命令である
  プロセッサは分岐予測を行うものとし，beq が分岐予測ミスを起こして LABEL に飛んだあとにフラッシュされてやり直した場合を想定せよ

```
  li x1←1
  li x2←2
  beq x1==x2, LABEL
  add x1←x2+x3
LABEL:
  add x2←x3+x4
```

- 13ns
  フォワーディング
  ありの場合



  やり直した後に
  結局 LABEL の下の
  命令も実行することに注意

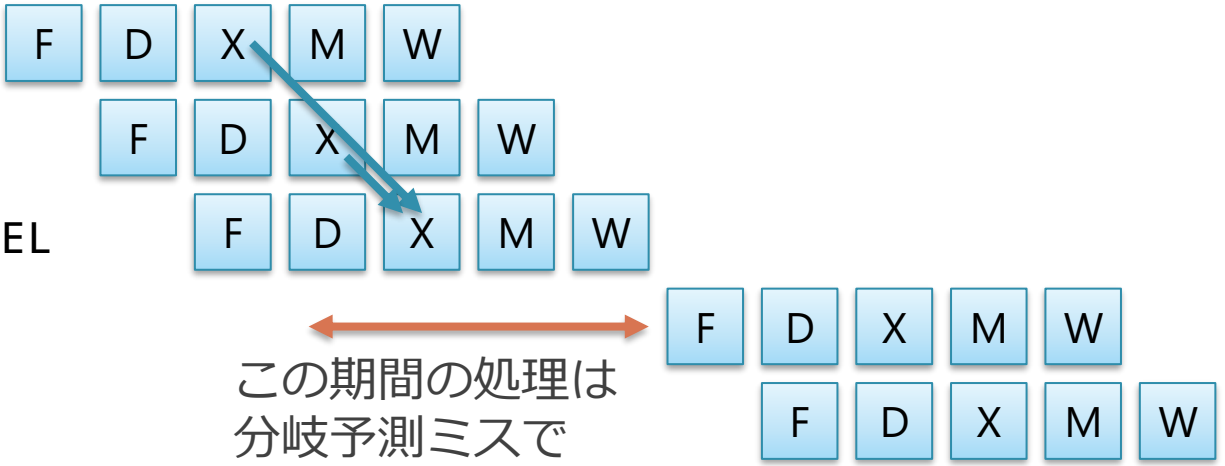


  パイプラインの各行は上の5つの命令を上から順にやった場合に対応

課題 7

- 13ns フォワーディングありの場合
  - やり直した後に結局 LABEL の下の命令（add x2←x3+x4）も実行することに注意

```
li x1←1
li x2←2
beq x1==x2, LABEL
add x1←x2+x3
add x2←x3+x4
```



性能のモデル

性能

- CPU の性能（処理速度）= 単位時間あたりに処理できる命令数
- 性能は以下の２要素のかけ算で決まる：
  1. クロック周波数 =
     １秒あたり何サイクル分の処理ができるか
  2. Instructions per cycle (IPC) =
     １サイクルあたり何命令処理できるか

性能のモデル

- 以下について検討していく
  - 理想的な場合の性能のモデル
  - ハザードを考慮した性能のモデル

理想的な場合の性能モデル

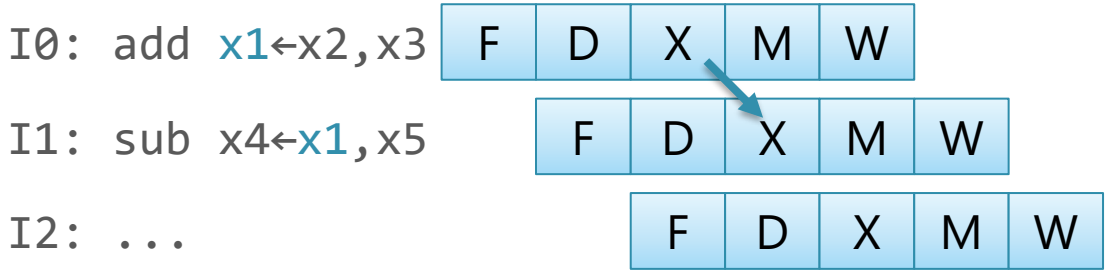
理想的な各モデルの性能

- 下記のそれぞれについて秒間何命令処理できるかを考える
  1. シングル・サイクル・プロセッサ
  2. スカラ・パイプライン・プロセッサ
  3. スカラ・パイプライン・プロセッサ（段数 2 倍）
  4. 2-way スーパスカラ・プロセッサ

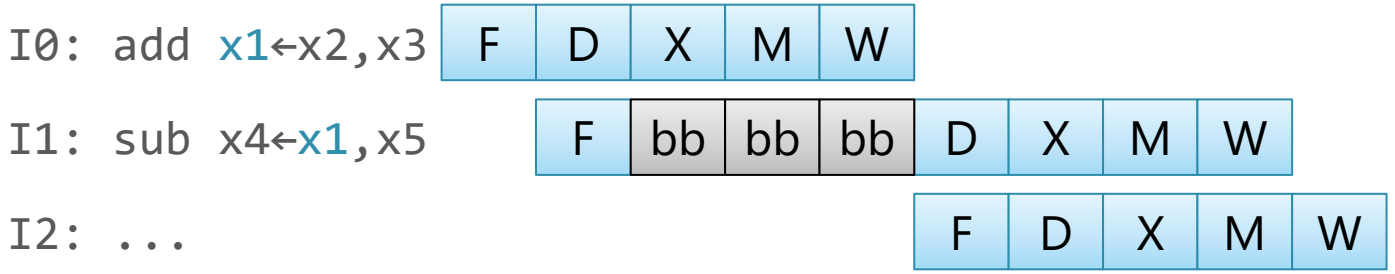
ここからの前提：パイプラインではフォワーディングの実装を仮定

- 結果が RF に書かれる前に，後続の命令は最速のタイミングでそれを使える（=フォワーディングを行う）と想定
- つまり，依存元命令の実行ステージ（以下では X）が終わり次第，直ちに依存先の命令は実行ステージを開始できる

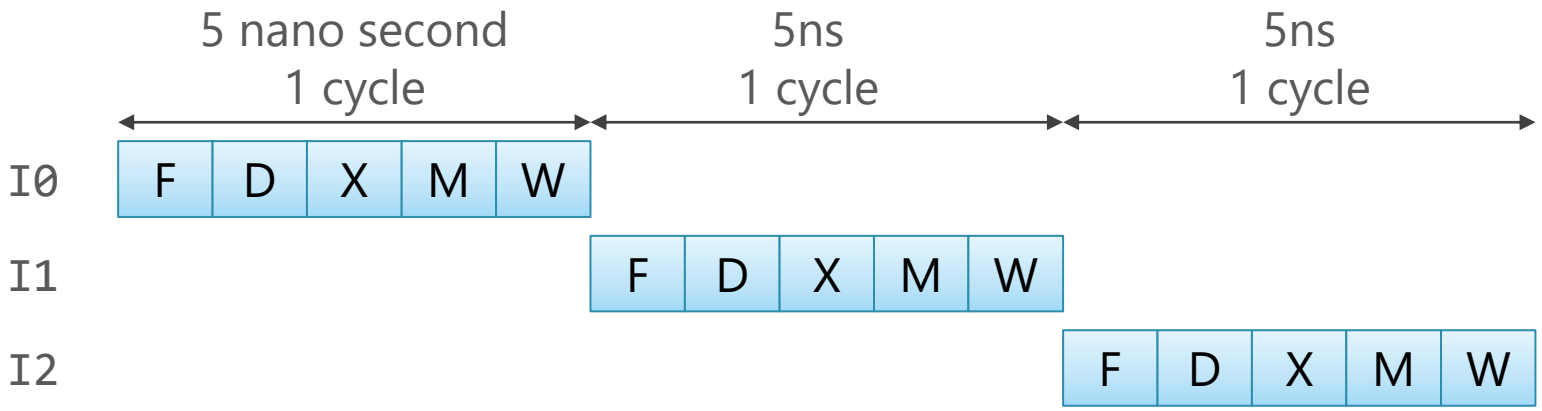
I0 の結果を I1 にフォワーディング



フォワーディングを実装しない場合，バブルが生じる

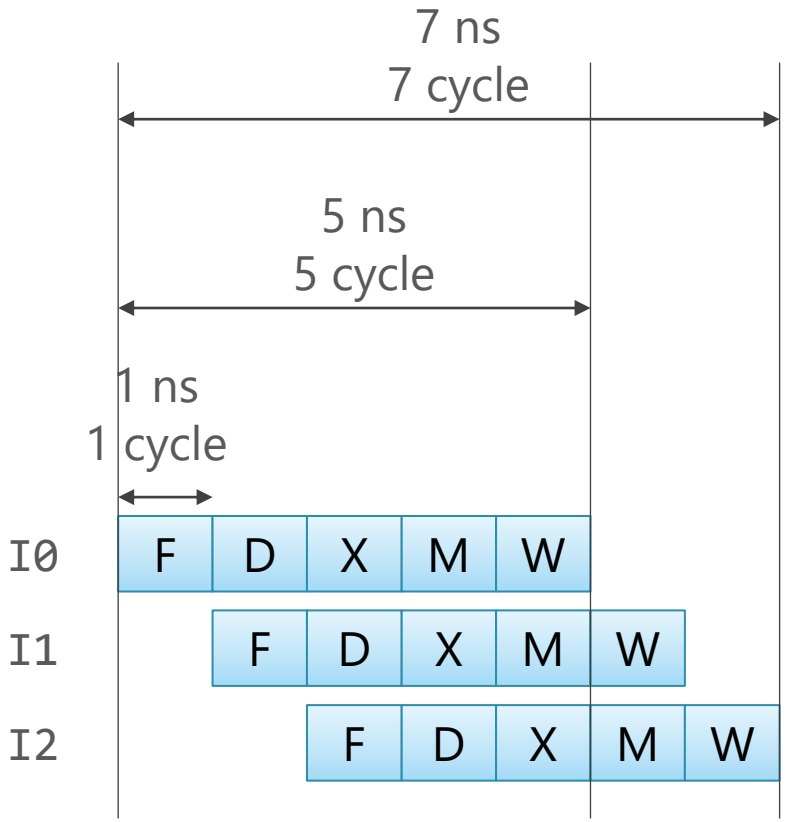


シングル・サイクル・プロセッサ の性能



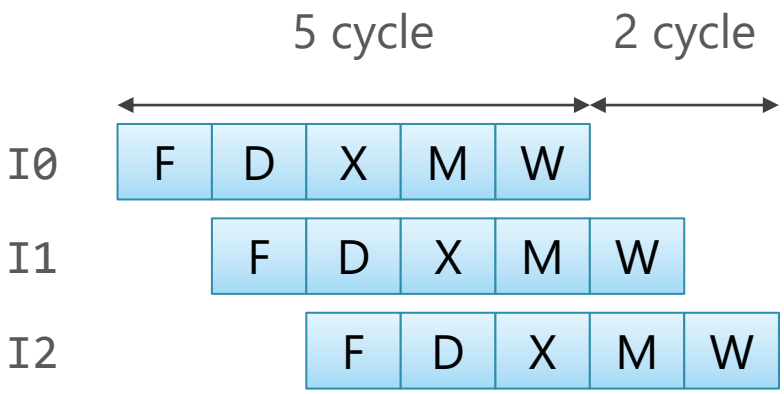
- IPC = 1命令 / 1サイクル = 1
- 周波数 = 1秒 / 5ns = 200 MHz（M = メガ 100万）
- 性能（1秒あたりの命令数）= IPC*周波数 = 秒間 200 M 命令

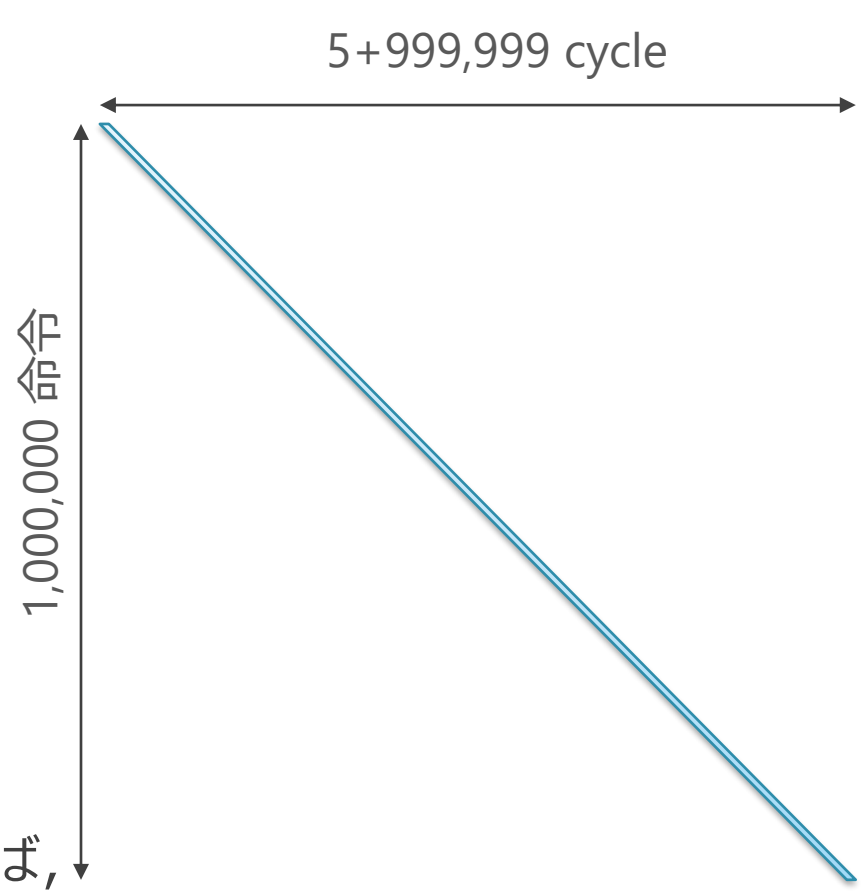
パイプライン化されたプロセッサ の性能



- シングル・サイクル・プロセッサの時とはサイクルの長さが変わっている事に注意
- 周波数 = 1秒 / 1ns = 1000 MHz = 1GHz
- IPC はどう考える？ IPC = 3 命令 / 7 サイクル？

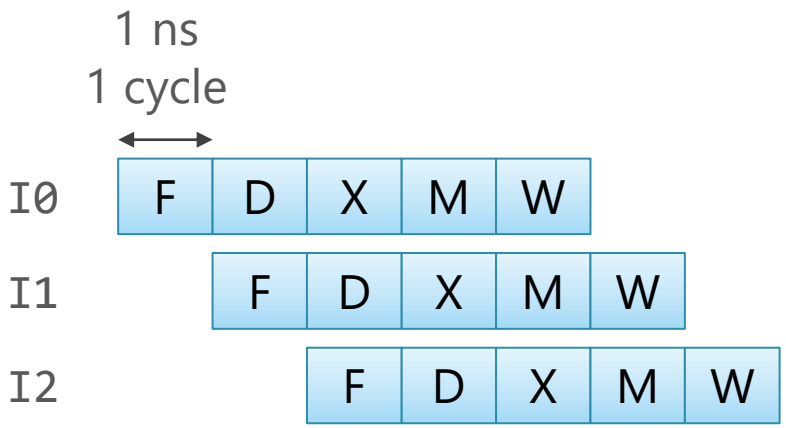
「十分に多く」の命令を実行した場合

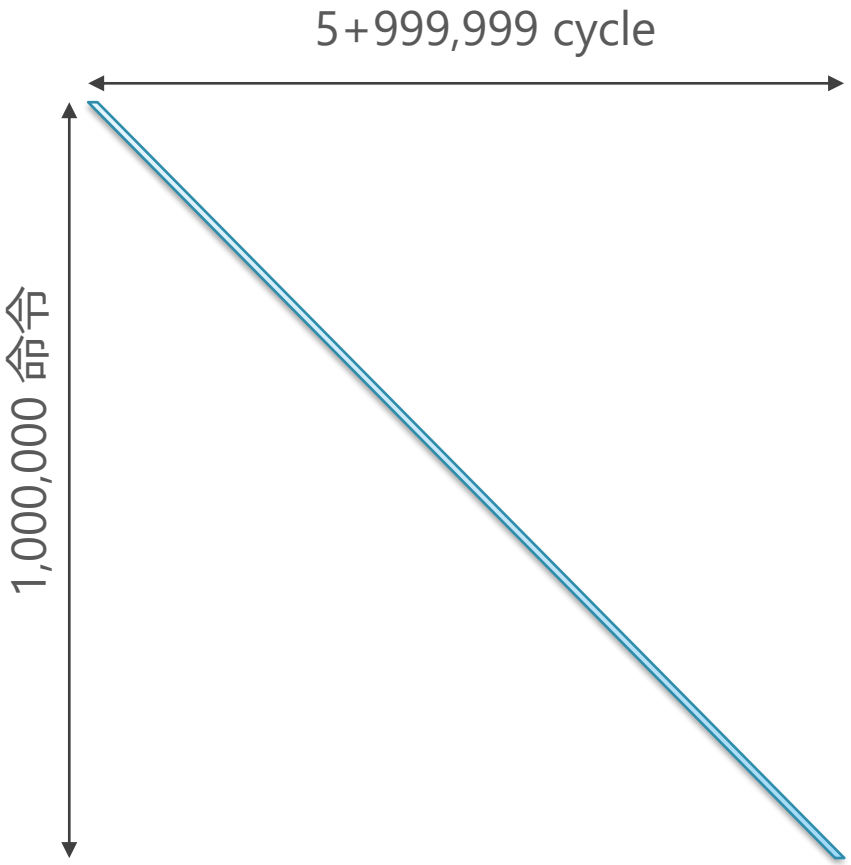




- 十分な長さの命令を実行すれば，
  パイプラインの長さ分は無視して近似できる
  - 右上の場合，ほぼ IPC = 1,000,000/(5+999,999) は，ほぼ 1
- 左端の F ないしは右端の W ステージの傾きのみを考えればよい

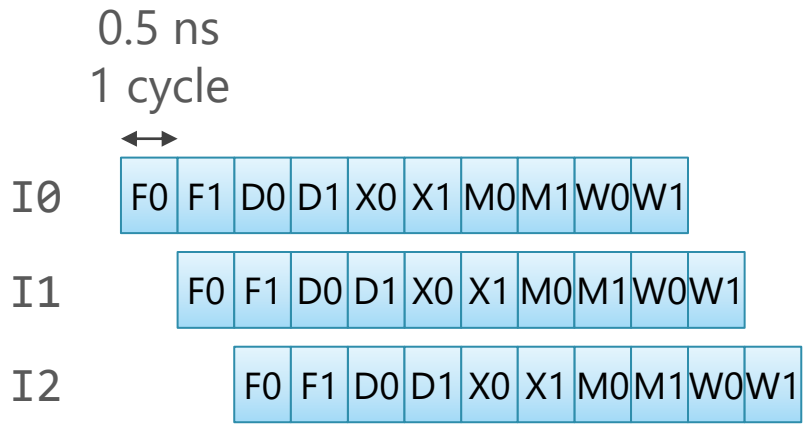
「十分に多く」の命令を実行した場合の性能





- IPC≈1命令 / 1サイクル = 1
- 周波数 = 1秒 / 1ns = 1000 MHz = 1GHz
- 性能（1秒あたりの命令数）= IPC*周波数 = 秒間 1G 命令

パイプライン段数を倍にした場合の性能

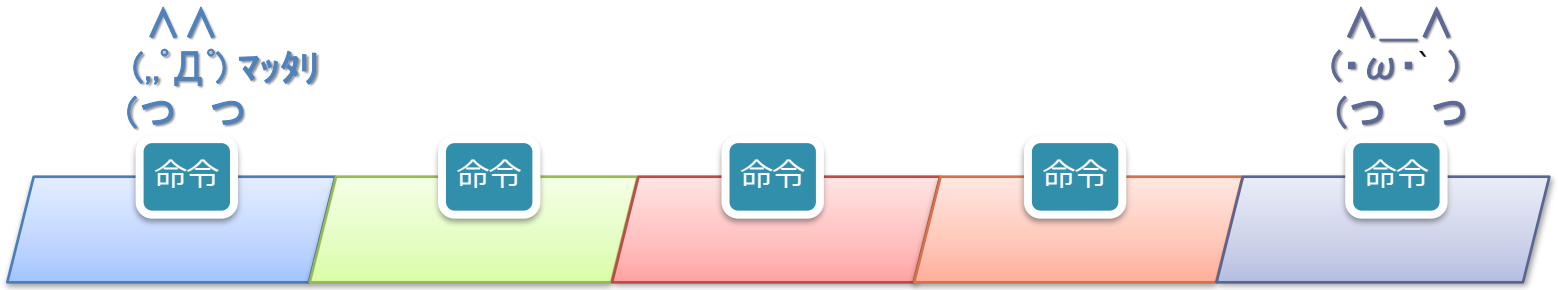


- １ページ前までのパイプラインのステージを半分に分割して（=各ステージで実施する仕事を半分にして）倍速で動かしている

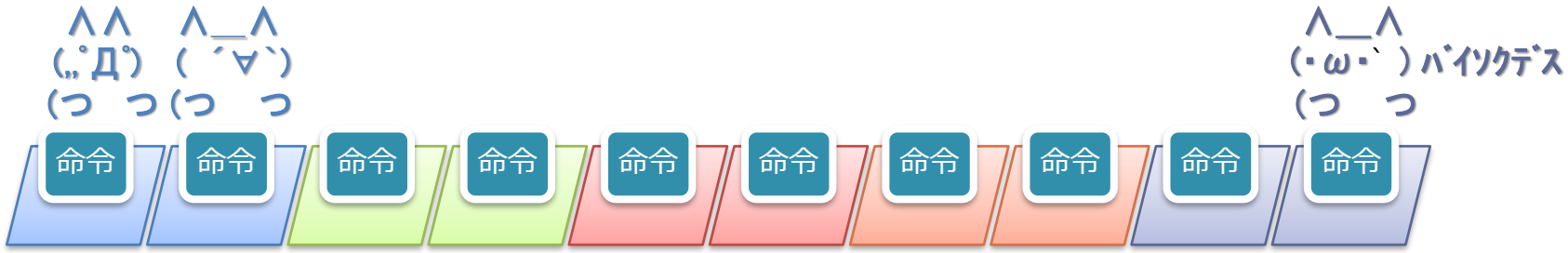
- IPC≈1命令 / 1サイクル = 1
- 周波数 = 1秒 / 0.5ns = 2000 MHz = 2GHz
- 性能（1秒あたりの命令数）= IPC*周波数 = 秒間 2G 命令

各ステージで実施する仕事を半分にして，かわりに倍速で動かしている

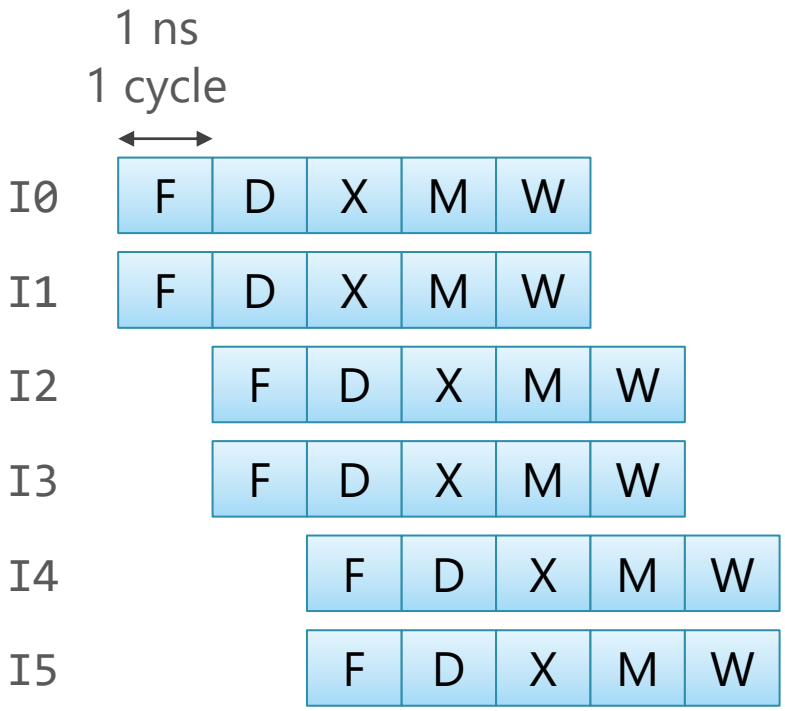
IPC≈1命令 / 1サイクル = 1
周波数 = 1秒 / 1ns = 1000 MHz = 1GHz

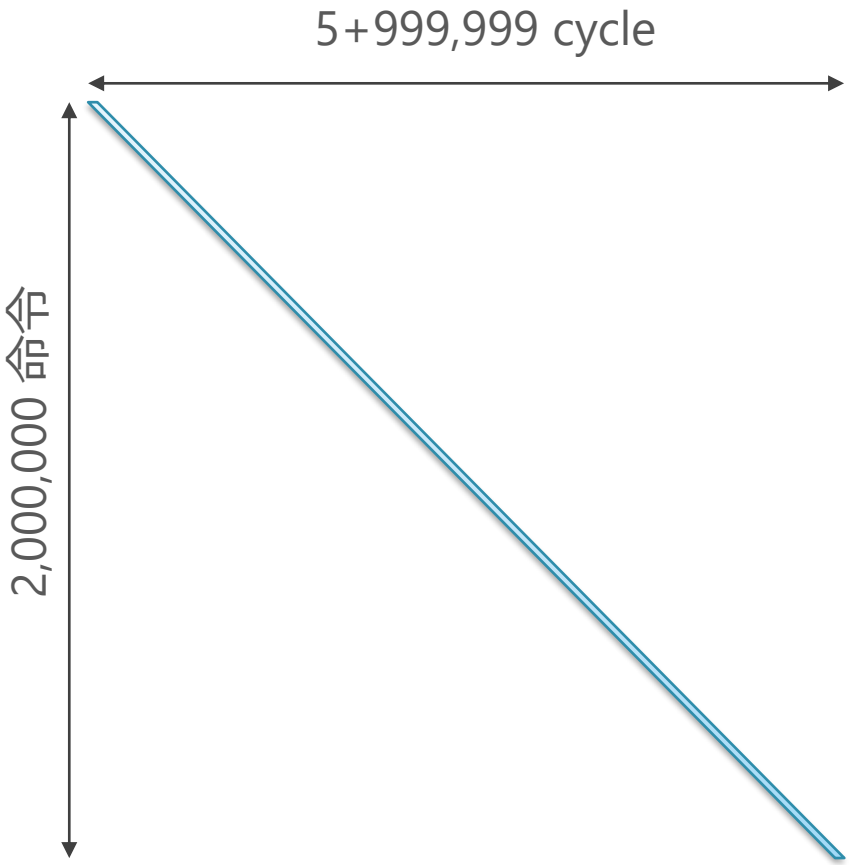


IPC≈1命令 / 1サイクル = 1
周波数 = 1秒 / 0.5ns = 2000 MHz = 2GHz
１人が１サイクルに行う仕事を半分にして，かわりに倍速で次に送る



2-way スーパスカラ・プロセッサの性能





- IPC ≈ 2命令 / 1サイクル = 2
- 周波数 = 1秒 / 1ns = 1000 MHz = 1GHz
- 性能（1秒あたりの命令数）= IPC*周波数 = 秒間 2G 命令

ここまでの性能は理想的な性能

- 理想的な各モデルの性能
  - シングル・サイクル・プロセッサ：　秒間 200M 命令
  - スカラ・パイプライン・プロセッサ：　秒間 1G 命令
  - スカラ・パイプライン・プロセッサ（段数２倍）：
    秒間 ２G 命令
  - 2-way スーパスカラ・プロセッサ：　秒間 ２G 命令

- 現実には･･･
  - 回路的な要因により（第５回講義），
    - スーパスカラにするとステージごとの回路が増えて遅延が増加 → 周波数は落ちる
    - パイプライン段数を２倍にしても周波数は倍にならない
  - 各種のハザードにより，ストールが起きて IPC が下がる（第６回講義）

ハザードを考慮した性能のモデル

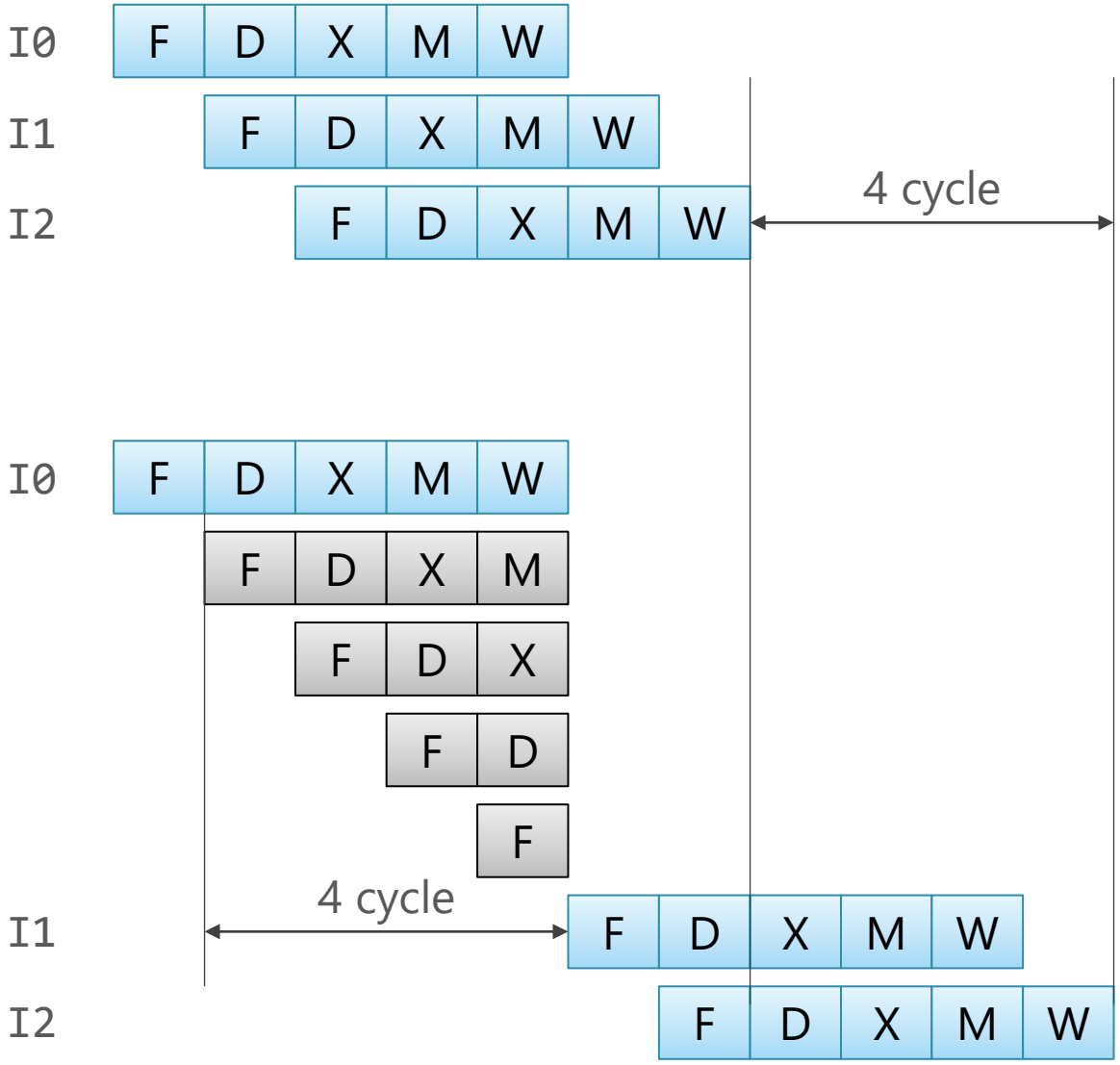
性能のモデル

- 以下について検討
  - 理想的な性能のモデル
  - ハザードを考慮した性能のモデル

ハザードによる IPC の低下

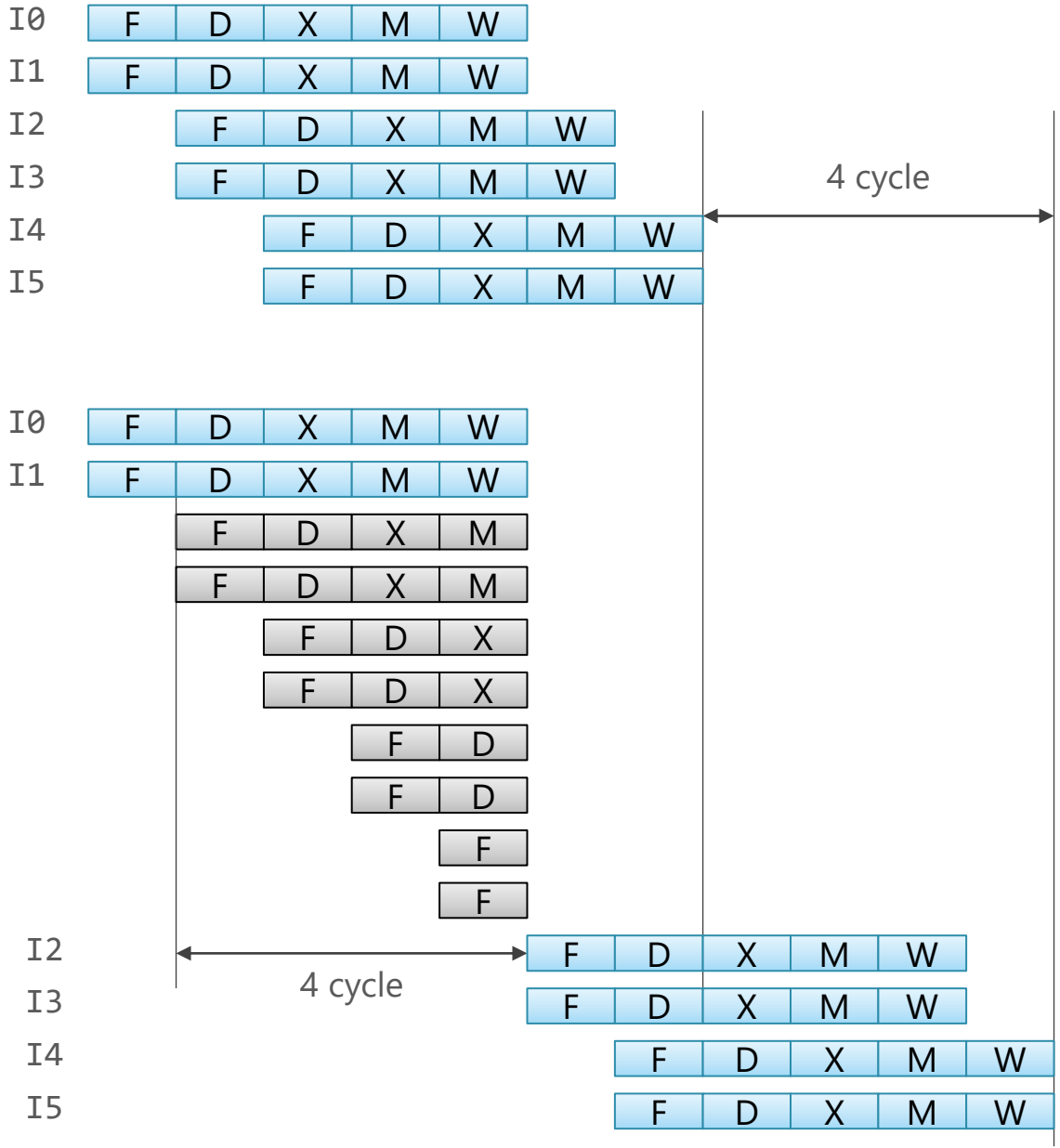
- 以下を例として，具体的な IPC の数字の変化を説明
  - 分岐予測ミス
  - ロードに依存した命令によるデータハザード
- キャッシュによる影響も大きい
  - （この講義ではまだ説明していないが，後日説明

分岐予測ミスによる実行サイクルの増加



- I2 の実行が 4 サイクル遅れている
- 4サイクル分の処理を取り消してやり直しているため

スーパスカラの場合

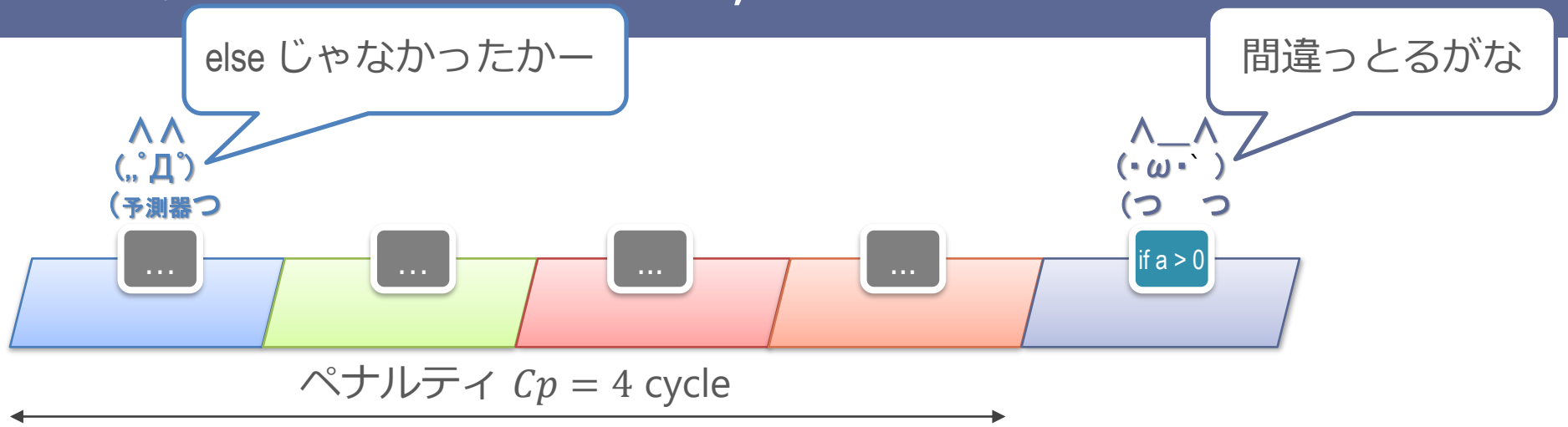


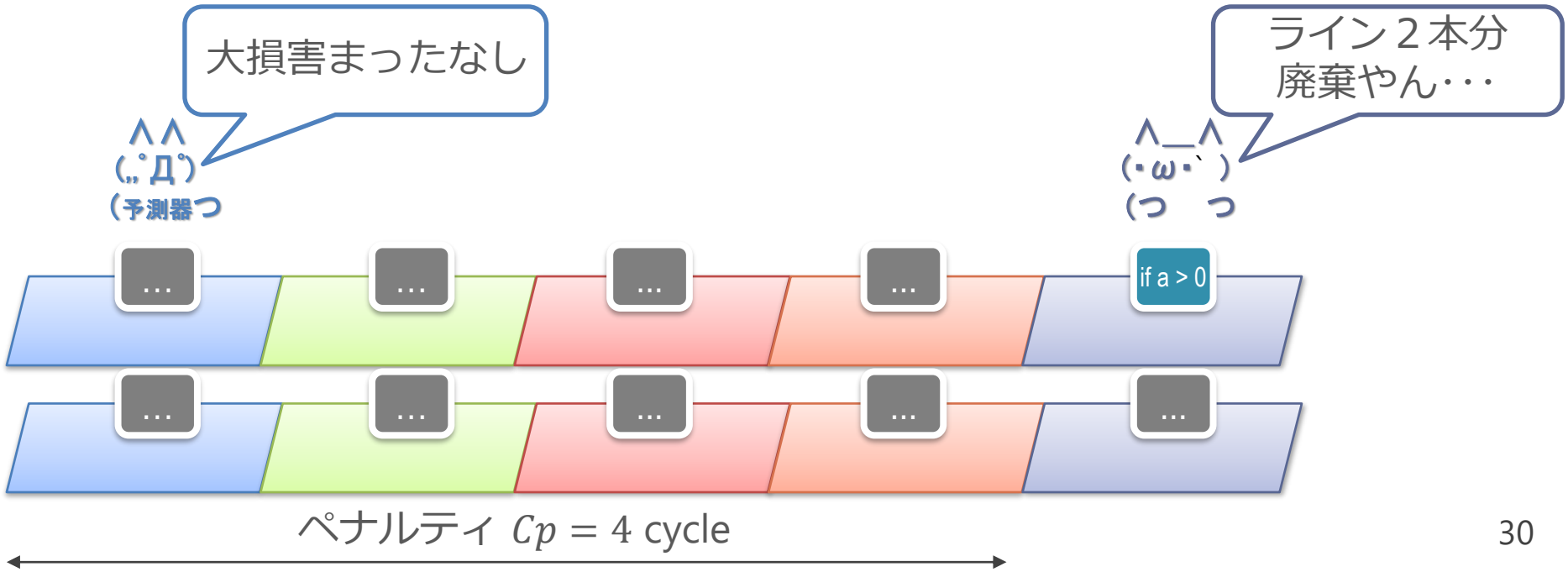
- I5 の実行が 4 サイクル遅れている
- 4サイクル分の処理を取り消してやり直しているため

分岐予測ミスによる実行サイクルの増加のモデル

- まとめると・・・
  - 予測ミス毎に，実行サイクル数がペナルティの分だけ伸びる
    - ペナルティは（パイプライン段数 – 1）サイクル
    - スカラでもスーパスカラでも同じ

ペナルティは（パイプライン段数 – 1）サイクル
取り消される命令数は増えるが，時間は同じ





分岐予測ミスによる実行サイクルの増加のモデル

- 以下のようにおいた場合，
  - ミスがない理想的な実行の際のサイクル数： $\boldsymbol{Ct}$
  - 分岐予測ミスの発生回数： $\boldsymbol{Nm} = Ni \times Pb \times Pm$
    - 実行命令数： Ni
    - プログラム中の分岐命令の出現率： Pb
    - 分岐命令毎の予測ミス発生率： Pm
  - 分岐予測ミス・ペナルティ： $\boldsymbol{Cp}$
- 実行サイクル数 Cr は，理想サイクル数 Ct に対して，
  - $\boldsymbol{Cr = Ct + Nm \times Cp}$
  - 意味：予測ミスの発生回数×ペナルティ だけ実行時間が伸びる

具体的な値を入れてみる

- 以下のように置いた場合，
  - 理想的な実行の際のサイクル数：$\boldsymbol{Ct = 1M}$（メガ）
  - 分岐予測ミスの発生回数：　$\boldsymbol{Nm = Ni \times Pb \times Pm = 0.025M}$
    - 実行命令数：　$Ni = 1M$
    - 分岐命令の出現率：　$Pb = 0.25$
      （プログラムは大体４命令に１つぐらい分岐が出てくる
    - 分岐命令毎の予測ミス発生率：　$Pm = 0.1$
  - 分岐予測ミス・ペナルティ：　$\boldsymbol{Cp = 4}$
- 実行サイクル数 Cr は
  - $\boldsymbol{Cr = Ct + Nm \times Cp = 1M + 0.1M = 1.1M}$
  - 分岐予測ミスにより 10% 実行サイクル数が伸びている

IPC で考えると

- 最終的な性能を考える上で IPC の方が都合がよい
- 実行サイクル数 Cr を命令数 Ni で正規化すると，
  - $\frac{Cr}{Ni} = \frac{Ct}{Ni} + \frac{(Nm \times Cp)}{Ni} = \frac{Ct}{Ni} + \frac{(Ni \times Pb \times Pm \times Cp)}{Ni} = \frac{Ct}{Ni} + Pb \times Pm \times Cp$
- IPC は命令数を実行サイクル数で割ったもの = つまり上記の逆数
  - $IPCr = \frac{1}{\frac{Ct}{Ni} + Pb \times Pm \times Cp} = \frac{1}{\frac{1}{IPCt} + Pb \times Pm \times Cp}$
  - ここで $IPCr$ は実際の IPC，$IPCt$ は理想 IPC

IPC の式のまとめ

- $IPCr = \frac{1}{\frac{1}{IPCt} + Pb \times Pm \times Cp}$

  - 実際の IPC： $IPCr$

  - 予測ミスがなかった場合の理想 IPC： $IPCt$
  - プログラム中の分岐命令の出現率： Pb
  - 分岐命令毎の予測ミス発生率： Pm
  - 分岐予測ミス・ペナルティ（サイクル）：Cp

  - （実行命令数が項から消えている

スカラの５段パイプライン・プロセッサの IPC

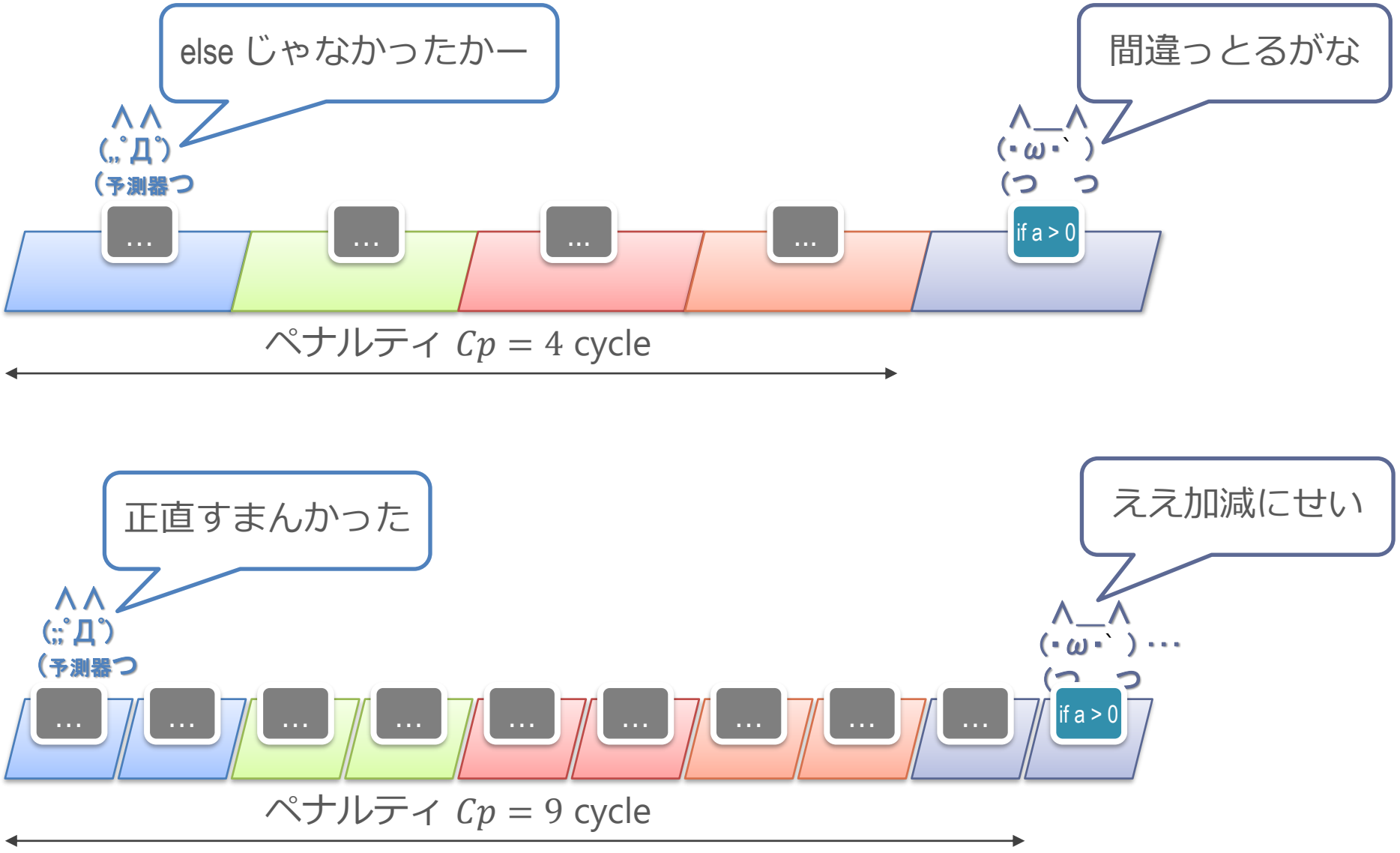
- $IPCr = \frac{1}{\frac{1}{IPCt} + Pb \times Pm \times Cp}$

- ここで，
  理想 IPC　　$IPCt = 1,$
  分岐発生率　$Pb = 0.25,$
  予測ミス率　$Pm = 0.2,$
  ペナルティ　$Cp = 4$ （5段なので$5 - 1 = 4$サイクル）とすると，
  - $IPCr = \frac{1}{1 + 0.25 \times 0.2 \times 4} \simeq 0.83$
  - つまり分岐予測ミスが無かった場合の理想 IPC と比べて
    17% ぐらい IPC が落ちている

パイプライン段数を２倍にした時の IPC

- $IPCr = \frac{1}{\frac{1}{IPCt} + Pb \times Pm \times Cp}$
- パイプライン段数を２倍にすると･･･
  - $IPCt = 1$，$Pb = 0.25$，$Pm = 0.2$ は変わらない
  - ペナルティは $Cp = 9$ （10段なので $10 - 1 = 9$）に
  - $IPCr = \frac{1}{1 + 0.25 \times 0.2 \times \mathbf{9}} \simeq 0.69$
  - 分岐予測ミスが無かった場合の理想 IPC と比べて 31% ぐらい性能が落ちている
    - パイプライン段数が５段の時より IPC 低下が大きい

パイプライン段数が深い時は予測器の精度が重要



パイプライン段数を２倍にした時の性能

- 全体の性能は周波数× IPC で決まる
  - 段数を倍にしたことで，周波数は２倍に
  - IPC は 0.69
  - 性能は $2 \times 0.69 = 1.38$

パイプライン段数を２倍にした時の IPC

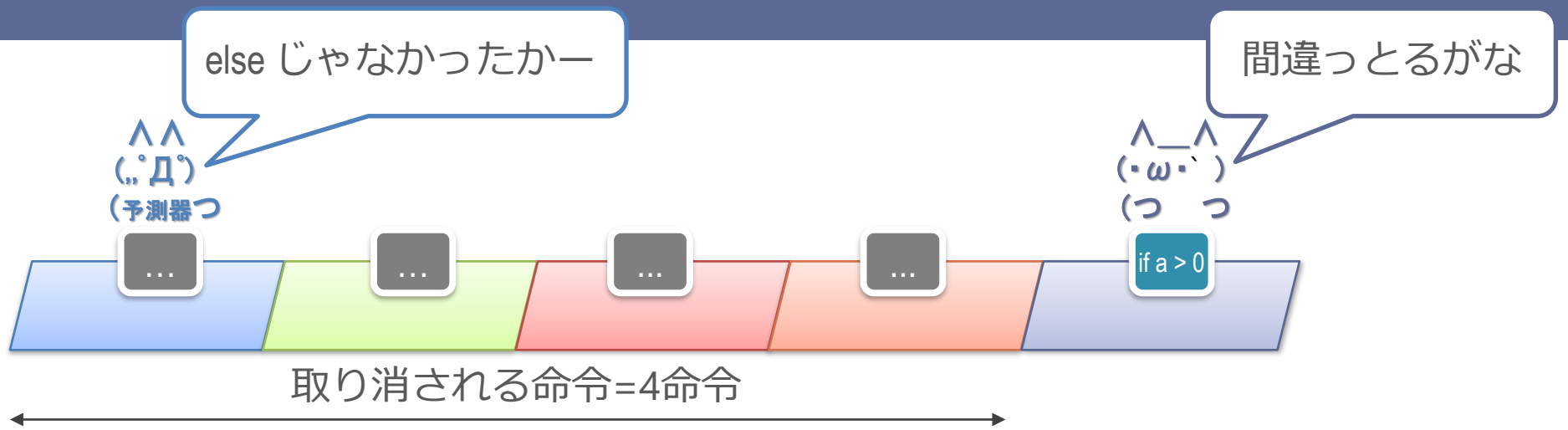
- まとめると，
  - ５段パイプの理想的な性能 ： 1
  - 分岐予測ミスありの性能： 0.69
  - １０段パイプの理想的な性能： 2
  - １０段で予測ミスあり： 1.38 (0.69 の 1.66倍)

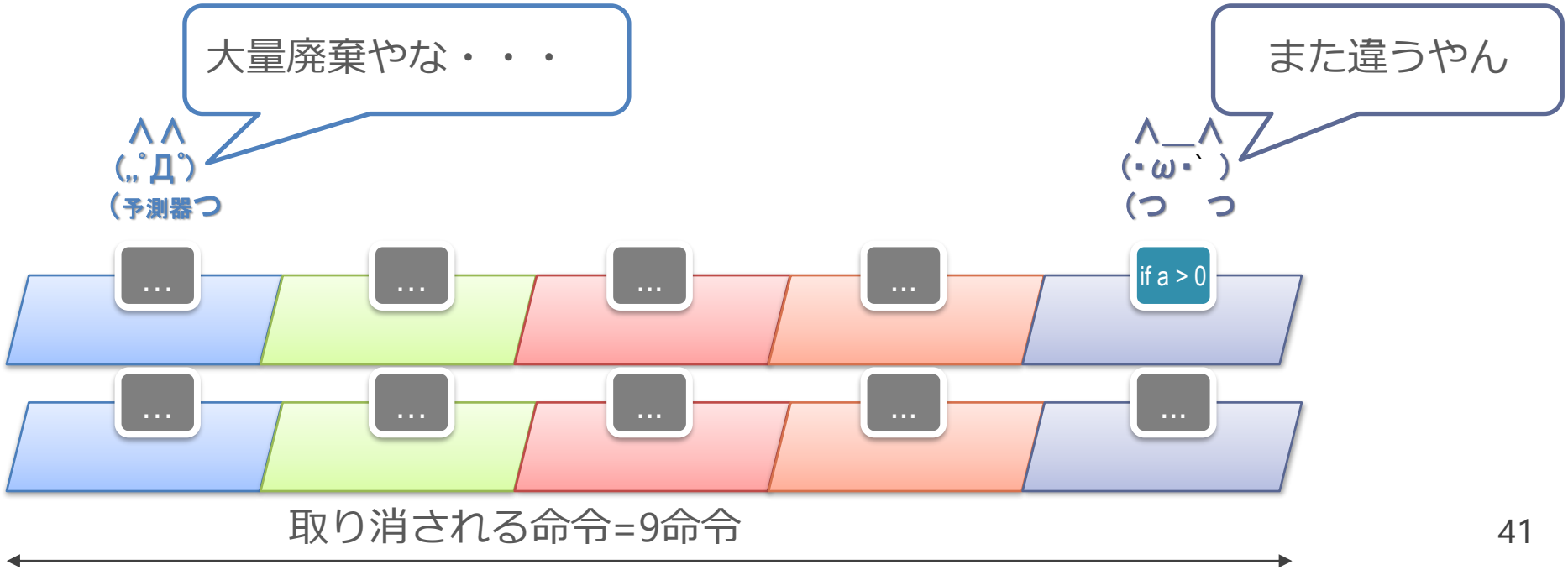
- わかること：
  1. パイプライン段数を深くすると IPC への影響が増加する
     - 同じ分岐予測ミス発生率でも，より大きく IPC が下がる
  2. 予測ミス率 0.2 で１０段パイプだと，全体の３割ぐらいの時間は無駄な実行をしている
     - 1.38 / 2 = 0.69

スーパスカラにした時の IPC

- $IPCr = \frac{1}{\frac{1}{IPCt} + Pb \times Pm \times Cp}$
- スーパスカラにすると・・・
  - $Pb = 0.25$，$Pm = 0.2$，$Cp = 4$ は変わらない
  - パイプラインが 2 本に増えたので $IPCt = 2$ に
  - $IPCr = \frac{1}{1/\mathbf{2} + 0.25 \times 0.2 \times 4} \simeq 1.42$
  - 分岐予測ミスが無かった場合の理想 IPC に比べると大幅に低い

スーパスカラにすると，その分取り消される命令が増える

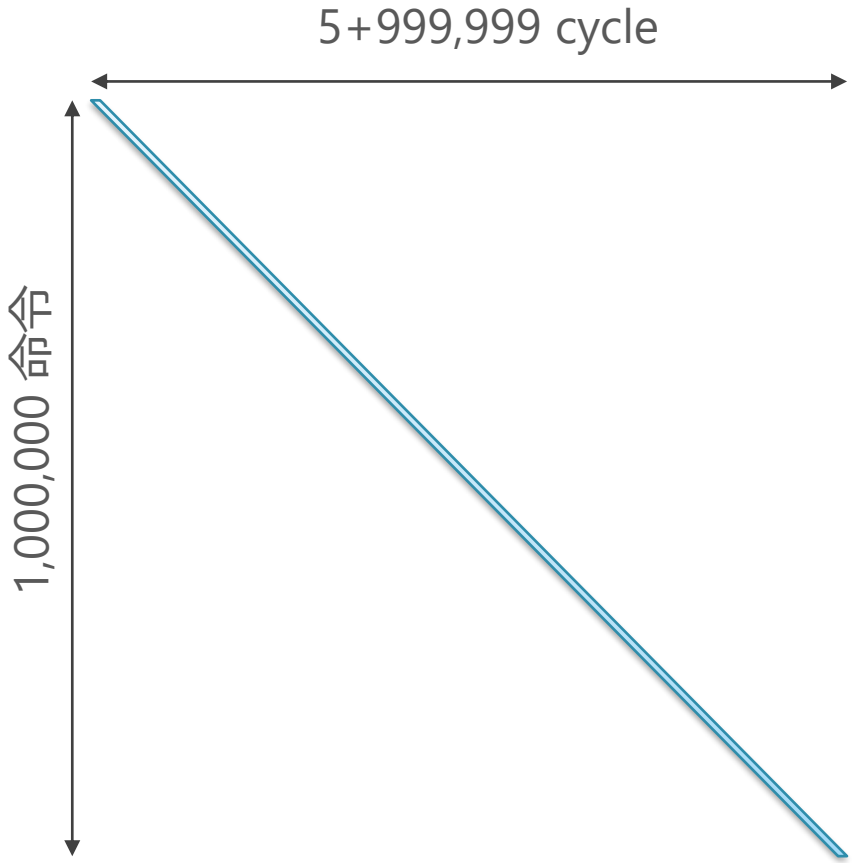


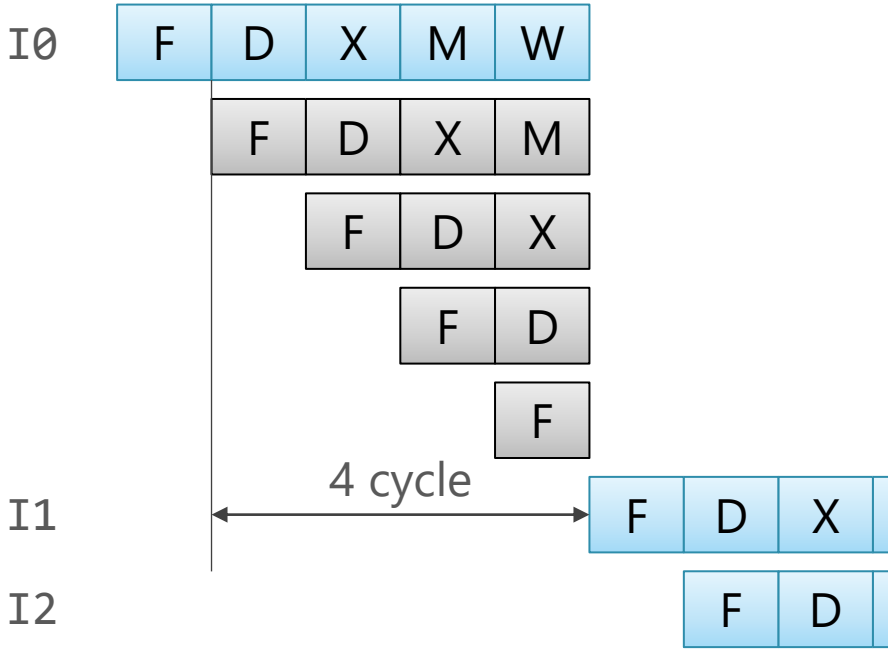


分岐予測ミスによる IPC の低下のまとめ

- 以下のような場合に，より予測ミスの影響が大きくなる
  - パイプラインを深くする（周波数を向上させる）場合
  - スーパスカラにしてパイプライン本数を増やす場合
- 裏を返すと･･･
  - パイプラインを深くしたり本数を増やす回路を追加する代わりに，予測器を強化した方が性能が上がることも
  - バランスが大事
    - へちょい分岐予測器のまま他を強化しても無駄と言う事も

パイプラインの長さと性能



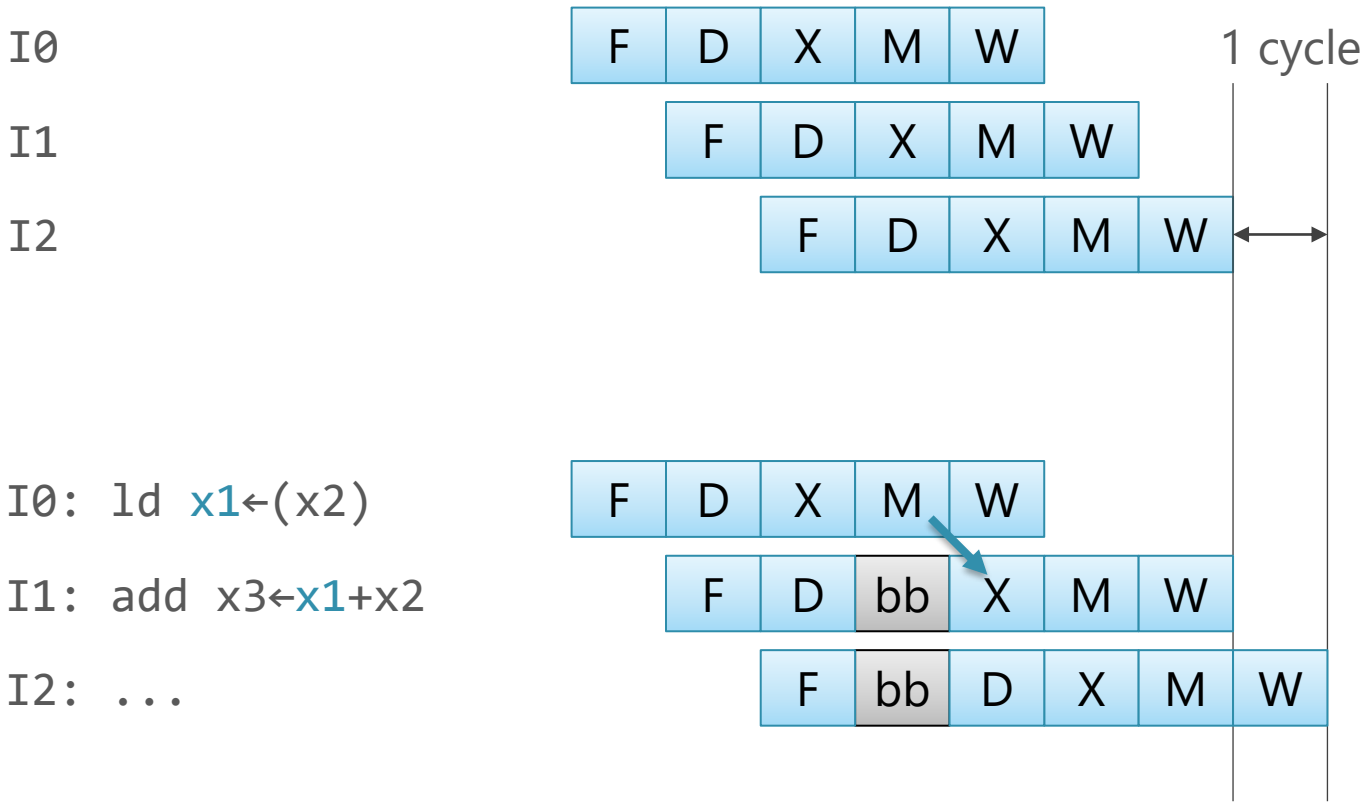


- パイプライン段数は，理想的なパイプラインではほとんど性能に影響しない
  - 最初の 1 命令分しか実行時間に影響しないから
- 分岐予測ミスは，起きる度にパイプラインの長さが表面化する

ハザードによる IPC の低下

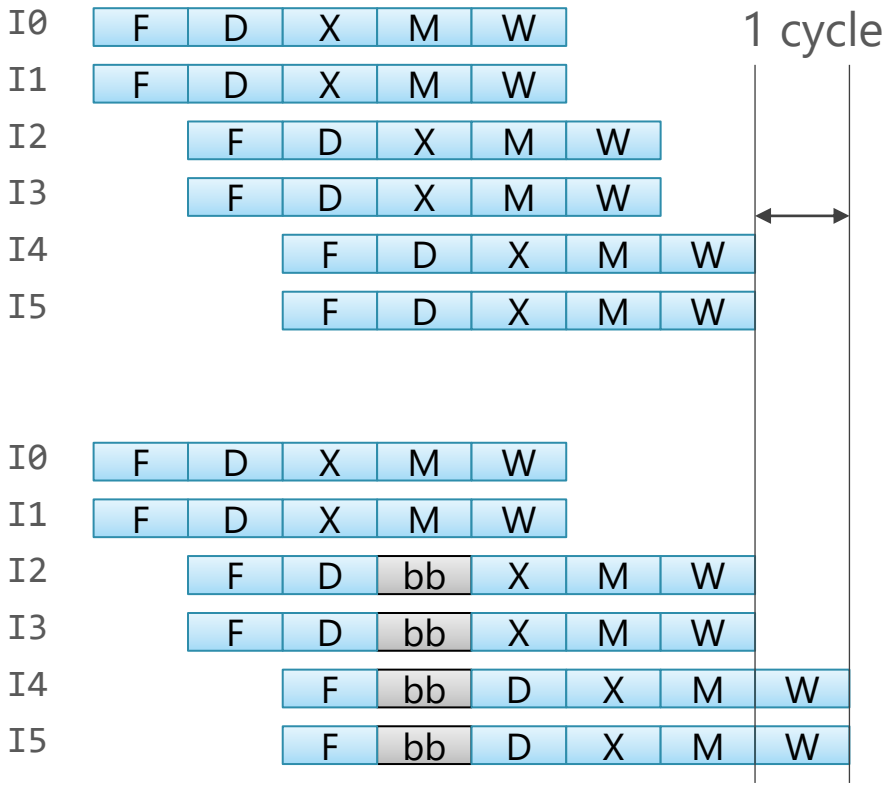
- 以下を例として，具体的な IPC の数字の変化を説明
  - 分岐予測ミス
  - ロードに依存した命令によるデータハザード

ロードに依存した命令によるデータハザード



- パイプライン全体が，理想的な場合と比べて
  1サイクル遅れている

スーパスカラの場合も同じ



- パイプライン全体が，理想的な場合と比べて
  １サイクル遅れている

ロードによるデータハザードの実行サイクルの増加のモデル

- 以下のようにおいた場合，
  - ミスがない理想的な実行の際のサイクル数： Ct
  - ロードのデータハザードの発生回数： $Nm = Ni \times Pl \times Ph$
    - 実行命令数： Ni
    - ロードの出現率： Pl
    - ロード命令毎のハザード発生率： Ph
  - ハザード時のサイクル数の増加： Cp
- 実行サイクル数 Cr は，理想サイクル数 Ct に対して，
  - $Cr = Ct + Nm \times Cp$
  - 分岐予測ミスの時とほぼ同じ

性能のモデルの一般化

一般化できる

- 以下のようにおいた場合，
  - 理想的な実行の際のサイクル数：Ct
  - 何らかのハザードの発生回数：　$Nh = Ni \times Pi \times Ph$
    - 実行命令数：　Ni
    - ハザードを起こす命令の出現率：　Pi
    - その命令毎のハザード発生率：　Ph
  - ハザード時のサイクル数の増加：Cp
- 実行サイクル数 Cr は，理想サイクル数 Ct に対して，
  - $Cr = Ct + Nh \times Cp$

一般化できる

- ハザード a，ハザード b，ハザード c・・・とあった時
- 実行サイクル数 Cr は，理想サイクル数 Ct に対して，
  - $Cr = Ct + Nma \times Cpa + Nmb \times Cpb + Nmc \times Cpc + \cdots$
  - これを積み上げ棒グラフで表したのが CPI Stack

CPI Stack

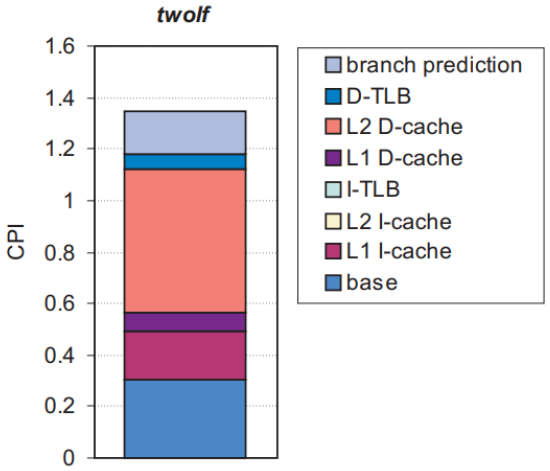


Figure 1. Example CPI stack for the twolf benchmark.

- Cycles Per Instruction (CPI) は 1 命令あたりのサイクル数
  - $Cr = Ct + Nma \times Cpa + Nmb \times Cpb + Nmc \times Cpc + \cdots$ を命令数で正規化して図示
  - $Ct = 0.25$ ぐらいなので，４way スーパスカラが基準？
- どのようなハザードによって性能が決まっているのかを直感的に知ることができる

図は Stijn Eyerman et al, A Performance Counter Architecture for Computing Accurate CPI Components より

Out-of-order スーパスカラ・プロセッサの場合

- Out-of-order スーパスカラ・プロセッサではもっと複雑
  - 長時間かかる命令による待ちがあった場合,
    それを待つ間に別の後ろにある命令を実行できる
  - 色々なハザードが同時並行で起きうる
- 単純な線形加算のモデルでは扱えない
  - $Cr = Ct + Nma \times Cpa + Nmb \times Cpb + Nmc \times Cpc + \cdots$
  - これはハザード発生時に CPU 全体がストールするからなり立つ
  - ハザード同士がオーバラップして起きると上手く扱えない
- 色々な方法が研究されている

ここまでのまとめ

- 性能は以下の２要素のかけ算で決まる：
  1. クロック周波数
  2. Instructions per cycle (IPC)

- 色んなバランスが大事
  - クロック周波数だけを上げても IPC が上がらないとだめ
  - スーパスカラの同時実行数を増やしても IPC が増えない事もある
  - 予測器だけ強くしても IPC があまり上がらないことも

実際には構成をどう決めるのか？

- だいたい最初に以下あたりの条件が決まっている
  - 要求性能
  - 使える回路面積（チップの大きさ）
  - 許容出来る消費電力

いろんな戦略がありえる

- 回路面積が一定の場合，どちらが良いか？
  - パイプラインの本数を増やす
  - 予測器を複雑にして精度を上げる
- 消費電力の上限が一定の場合
  - 消費電力はクロック周波数の２～３乗で増える
  - クロック周波数をあえて下げるかわりに，
    パイプライン本数を増やして IPC を上げる
- 色々シミュレーションやモデル化して決める

まとめ

- 性能は以下の２要素のかけ算で決まる：
  1. クロック周波数
  2. Instructions per cycle (IPC)
- 以下について検討
  - 理想的な場合の性能のモデル
  - ハザードを考慮した性能のモデル
- 性能は色々な要素の相互作用で決まる

課題 8

- 以下のような条件を考える
  - 10段のパイプラインを持つ 2-way スーパスカラプロセッサであり，理想的には IPC=2 で実行できる
  - 全実行命令におけるなんらかのデータハザードの発生率は 0.2
  - このデータハザード発生時は 1 サイクル実行時間が伸びるものとする
  - 全実行命令における分岐命令の出現率は 0.2
  - 分岐予測ミス率は 0.3

- (1) この CPU を改良する際,
  「3-way スーパスカラにする」
  「2-way のまま15段パイプラインにする」
  「2-way のまま分岐予測器を改良する」
  のどれが最も性能が上がるかを性能を計算して検討せよ
  - この CPU を 3-way スーパスカラにすると理想的には IPC=3 で実行できるがデータハザードの発生率は 0.3 に上昇するとする
  - また，分岐予測器を改良すると分岐予測ミス率が 0.2 にまで削減されるとする

課題 8

- (2) ここで効率を表す「性能エネルギー比（性能 / 消費エネルギー）」という指標を導入する．この数字が大きいほど小さなエネルギーで速く動くことを意味する．前述の 3 つの改良方針のうち，以下の追加の条件のもとで，最も性能エネルギー比が良いものはどれかを計算して検討せよ．
  - クロック周波数を N 倍にすると，消費エネルギーは N^2 倍増える
  - 3-way スーパースカラにすると消費エネルギーは 1.5 倍になる
  - 分岐予測器を改良すると消費エネルギーは 1.1 倍になる

提出方法

- 以下を提出：
  1. 課題８：
     - 提出は Moodle の「課題８」のところからお願いします
     - 紙に書いた場合は写真を撮ってアップロードしてください
  2. 感想や質問：
     - 「感想や質問」のところに投稿してください
     - わからない場所がある場合，具体的に書いてもらえると良いです

- 提出締め切り
  - Moodle に設定した締め切りまで（6/25 日曜日の 23:59 頃，要確認）

- 注意：
  - 課題の出来は，ある程度努力したあとがあれば良しです
    - 必ずしも正解していなくても良いです

質問とか感想

質問とか感想

- 最大周波数が良くわかりませんでした。
- 課題に出てきた最大動作周波数がよく分からなかったです。

質問とか感想

- 依存の関係や取り除けているのか（破壊が起こっていないか）の確認のところで、頭の中がごちゃごちゃになってしまいました。なぜOoOの発行を1としたのかふと気になりました。

質問とか感想

- 他のレジスタを使うようにプログラムを書き換えることの欠点はレジスタの量に限りがあることでしたよね。私たちが普段使っているパソコンのレジスタはどのくらいですか？

質問とか感想

- 基本情報が取れるか不安になってきました

質問とか感想

- 掛け算だとなぜ演算が4回分必要になるのですか

質問とか感想

- しかし、解説は理解できても、いきなり課題を解こうとすると全くわからなくなってしまいます。実践とは難しいものですね。

質問とか感想

- 課題が難しかったです。用語に馴染みがなく、用語を聞いて何をすることを表しているのか、パッとわからないことでも、より課題を難しく感じました。期末試験が怖いです。。
  用語を覚えるコツなどありますでしょうか？

質問とか感想

- 静的、動的という用語はシステムプログラミングで学んだのですが、静的は事前にプログラム内の命令を並び替えておき、動的はCPUが実行時に並び替えるという違いがあることをよくわかっていなかったので授業で学ぶことができてよかったです。

質問とか感想

- 静的スケジューリングをするときに、表面上は入れ替えることができても、アドレスなど見た目だけではぱっと見わからにことが関わってくると、判断が難しいなと思いました。

質問とか感想

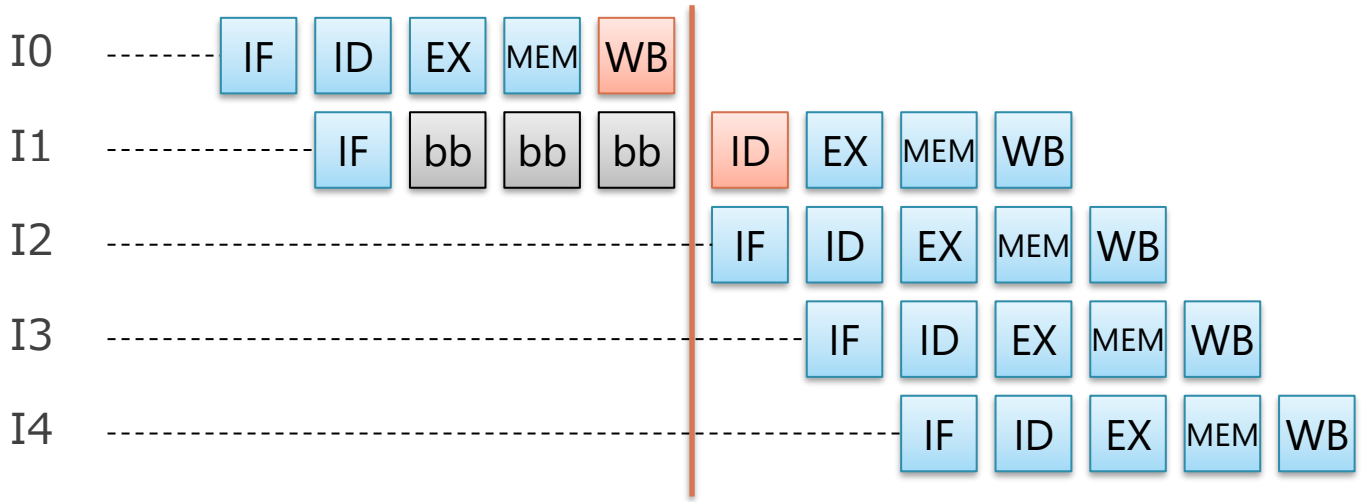
- 第6回スライドp53では
  ```
  b = a+1:    IF ID EX MEM WB
  c = b-1:       IF bb bb  bb  ID  EX  MEM  WB
  ```
  と書かれているのですが、

- 第7回スライドp64では
  ```
  mul x1←x2*4:   IF  ID  EX  EX  EX  EX  MEM  WB
  add x3←x1+1:       IF  ID  bb  bb  bb  EX   MEM  WB
  ```
  と書かれています。

- bbを書く場合、前者のようにIFとIDの間に書くときと、後者のようにIDとEXの間に書くときの違いはなんなのでしょうか？

  - 上はフォワーディングがない（=WB が終わるまで ID が開始できない）場合の想定で,
    下はフォワーディングがある（=EX が終わり次第次の EX が開始できる）と言う違いです

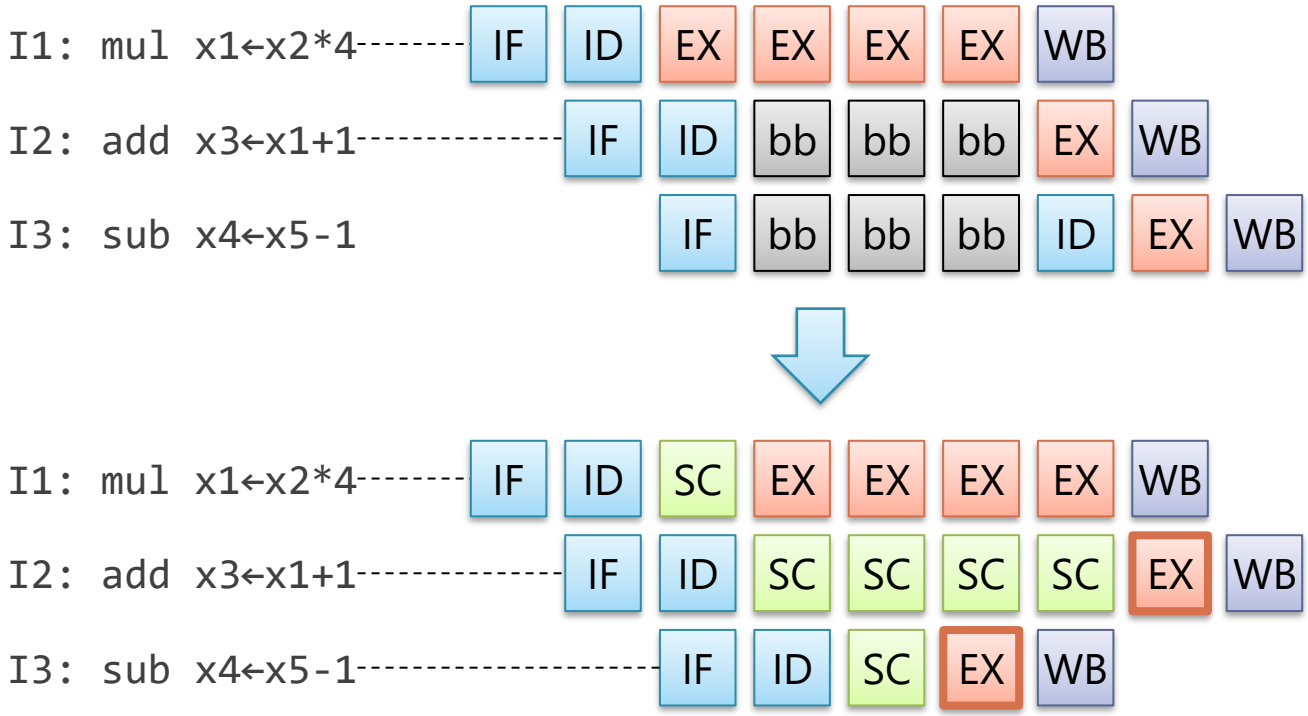
1．ストールさせる



- I0 の WB が終わるまで，後続の命令を遅らせる
  - I1 の ID が，I0 の WB の右にくるまでストール
  - I1 は I0 の結果を使える
- 欠点：プログラムの実行がとても遅くなる

in-order 実行と out-of-order 実行の違い

- Out-of-order 実行だと，I3 が I2 を追い越して実行できる
  - 各命令はスケジュール（SC）に入って実行可能になるまで待つ
  - 発行キューは，その時実行可能なものから順に EX に命令を送る
    - x1 は計算中なので SC で待つ
    - x5 は既に結果が得られていたので，先に送信



質問とか感想

- そもそもIF,ID,EX…などの役割の理解に怪しい部分もあり…..教えていただきたいです。よろしくお願いします。

パイプライン化（５回目の講義資料より）

- 回路のまとまりをオーバラップさせる単位にする
  - この単位をステージと呼ぶ
- ステージ
  1. IF：　命令フェッチ
  2. ID：　デコードとレジスタ読み出し
  3. EX：　実行
  4. MEM：メモリ・アクセス
  5. WB：　レジスタ書き込み

質問とか感想

- 私の理解力が低いせいなのですが、課題をやるときなんとなくの理解でしかできていないので、講義資料に例題とかもあるともっとわかりやすいなと思っています…

  - 講義資料のページの一番下にあります

質問とか感想

- 以前ハッカソンに出た時にユーザーから見える部分の開発担当はフロントエンド、見えない部分の開発担当はバックエンドと呼ぶことを初めて知りましたが、コンピュータアーキテクチャでもOut-of-orderの説明に出てきて驚きました。フロントエンドやバックエンドの意味はコンピュータアーキテクチャでもWebアプリ開発でも同じなのでしょうか。

質問とか感想

- フラッシュしたりラベルに飛ぶという動作に、シングル・サイクル・プロセッサの動作は行われますか？

質問とか感想

- 課題の必要な時間を求める問題が全然わかりませんでした。授業ではここでbbが生まれて、、など言われて納得できている気持ちになっているのですが、自分で考えてみると、全然理解していなかったのだと気づきます。

質問とか感想

- 新しいネコちゃん達が登場してきて可愛かったです。この授業では大分ネコちゃんたちに助けられているのでありがとうと言いたいです。

質問とか感想

- Cなどでプログラムを書いていても、実際にCPU側で書いた通りの順番で命令が行われているのかを疑ったことがなかったので、スケジューリングという概念を斬新に感じた。またスケジューリングや（前の授業の）ハザードの回避方法についてはなんとなく理解していたつもりだったが、いざ課題で応用してみるとなると難しかった。

質問とか感想

- また、分岐予測のアルゴリズムについて具体的に教えていただきたいです。特に、どのような種類があり、それぞれの精度の違いはどのようなものなのでしょうか。また、これらのアルゴリズムが実際にどのように実装され、どのように動作するのかについても詳しく知りたいです。

  - 院生むけの講義資料ですが，興味があればこちらも
  - https://github.com/shioyadan/advanced-computer-organization/blob/master/aco-shioya-06.pdf
  - https://github.com/shioyadan/advanced-computer-organization/blob/master/aco-shioya-appendix-bpred.pdf

現代の分岐予測器の性能（64 KB 使用の場合）

PIERRE MICHAUD, An Alternative TAGE-like Conditional Branch Predictor, TACO 2018 より