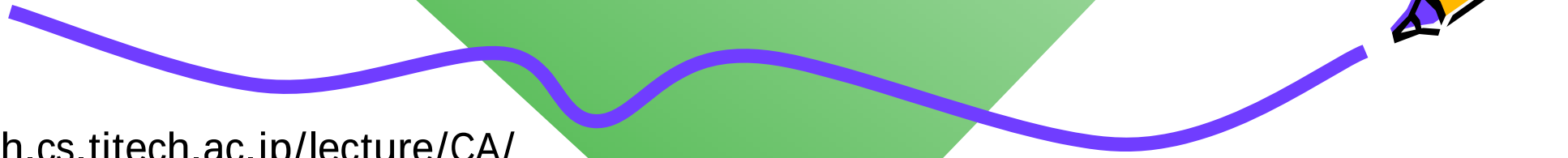
2018年度（平成30年度）版

Ver. 2018-10-23a

Course number: CSC.T363

コンピュータアーキテクチャ
Computer Architecture

8. スーパースカラプロセッサ
Superscalar Processor

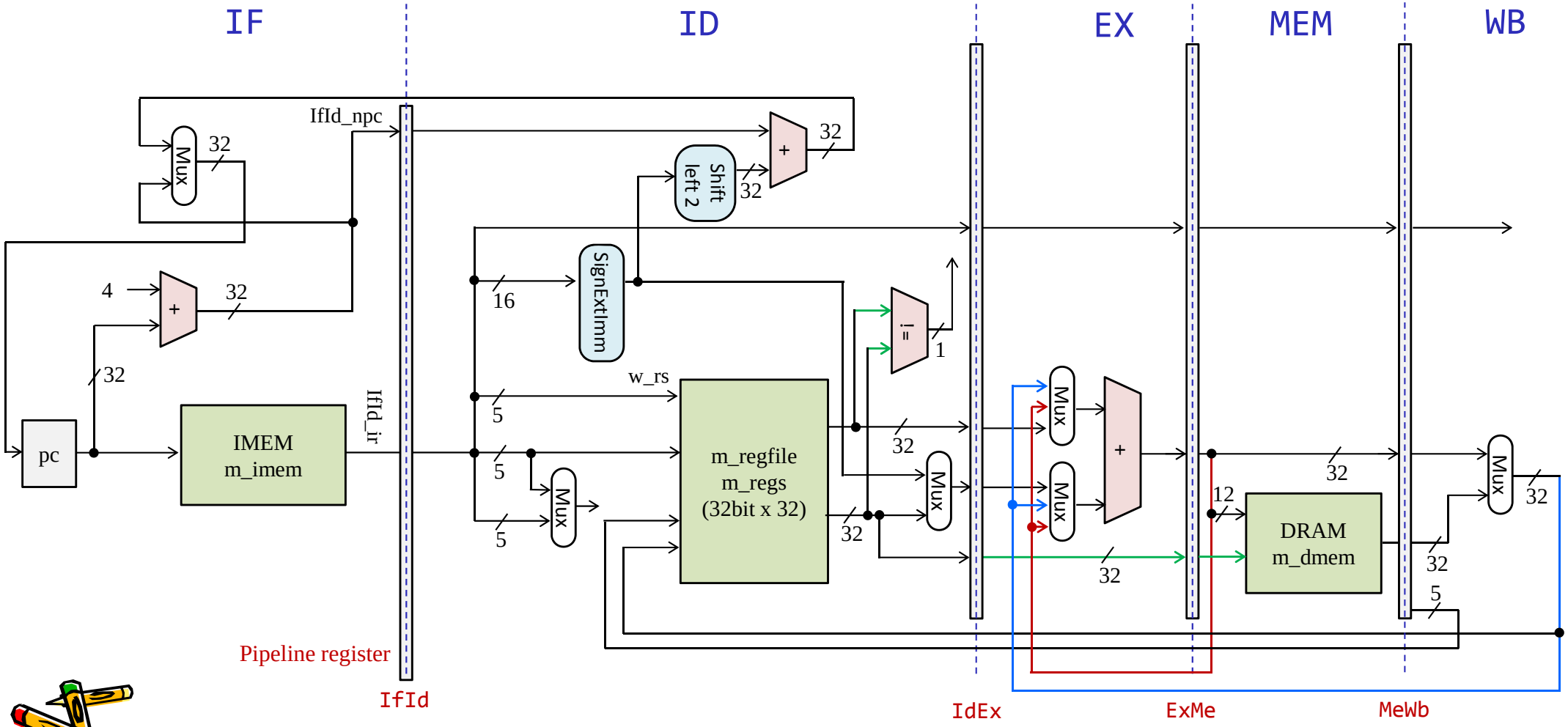


www.arch.cs.titech.ac.jp/lecture/CA/
Room No.W321
Tue 13:20-16:20, Fri 13:20-14:50

吉瀬 謙二　情報工学系
Kenji Kise, Department of Computer Science
kise _at_ c.titech.ac.jp

リファレンスデザインに含まれるプロセッサ

- Operating frequency: 50MHz
- リファレンスデザインには，5段パイプライン処理の典型的なMIPSプロセッサ（のサブセット）が採用されている．下の図からレジスタ名などが変わっているので注意．



Performance Factors

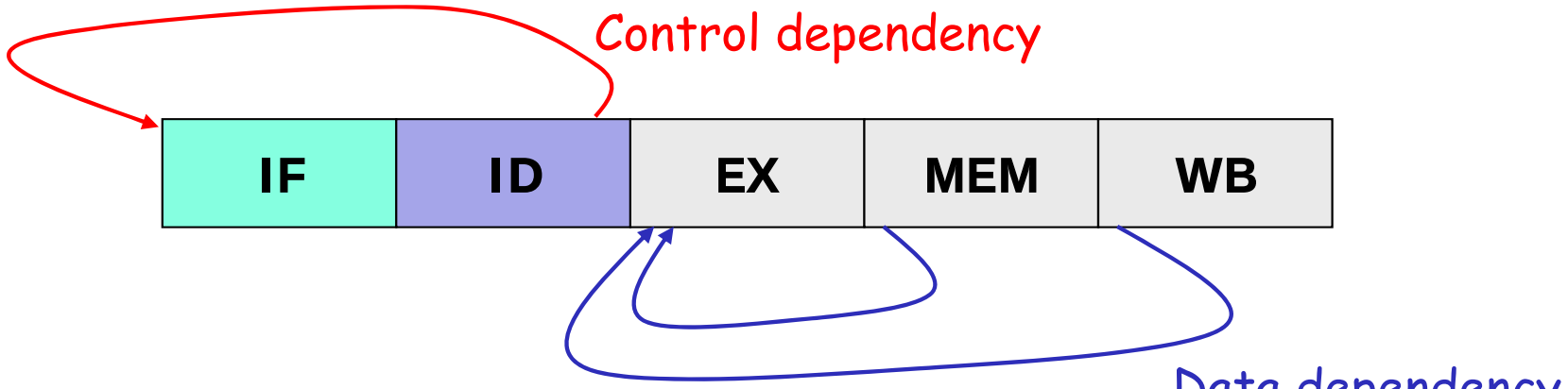
$$\text{CPU execution time for a program} = \frac{\text{\# CPU clock cycles for a program}}{\text{clock rate}}$$

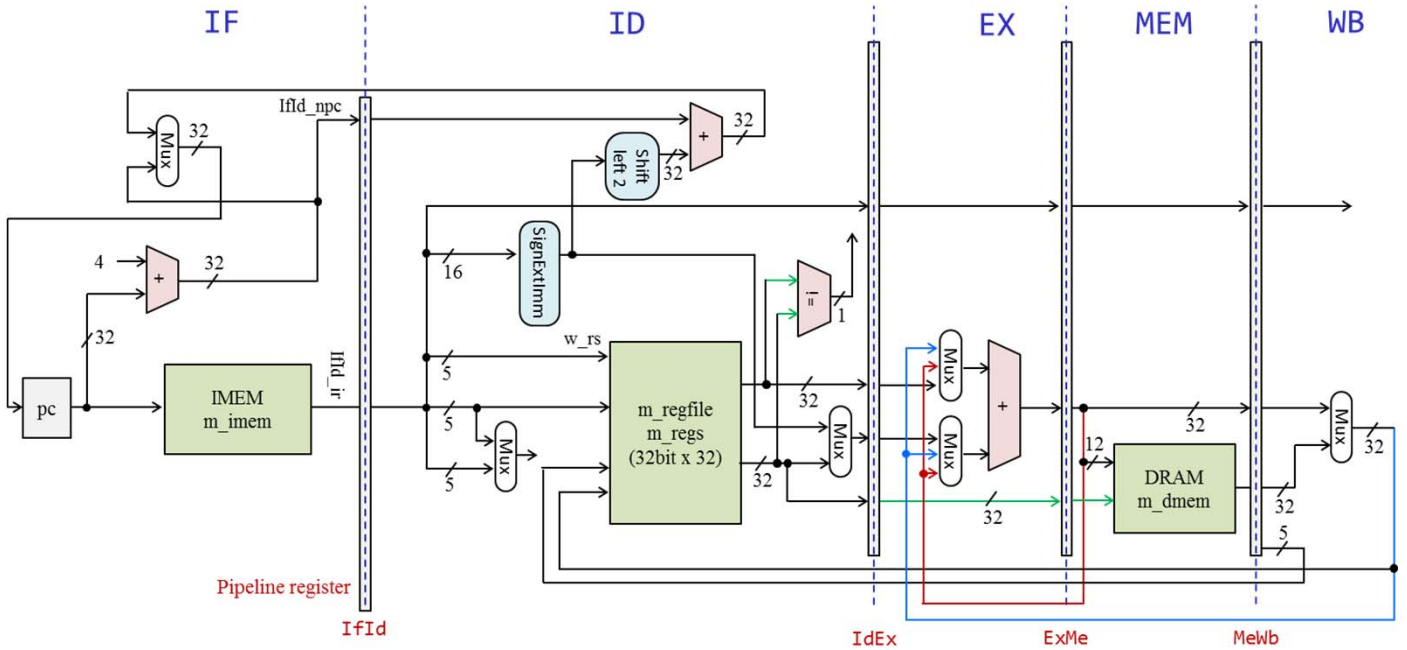
Performance = clock rate x 1 / # CPU clock cycles for a program

- Performance = f x IPC
  - f: frequency (clock rate)
  - IPC: retired instructions per cycle

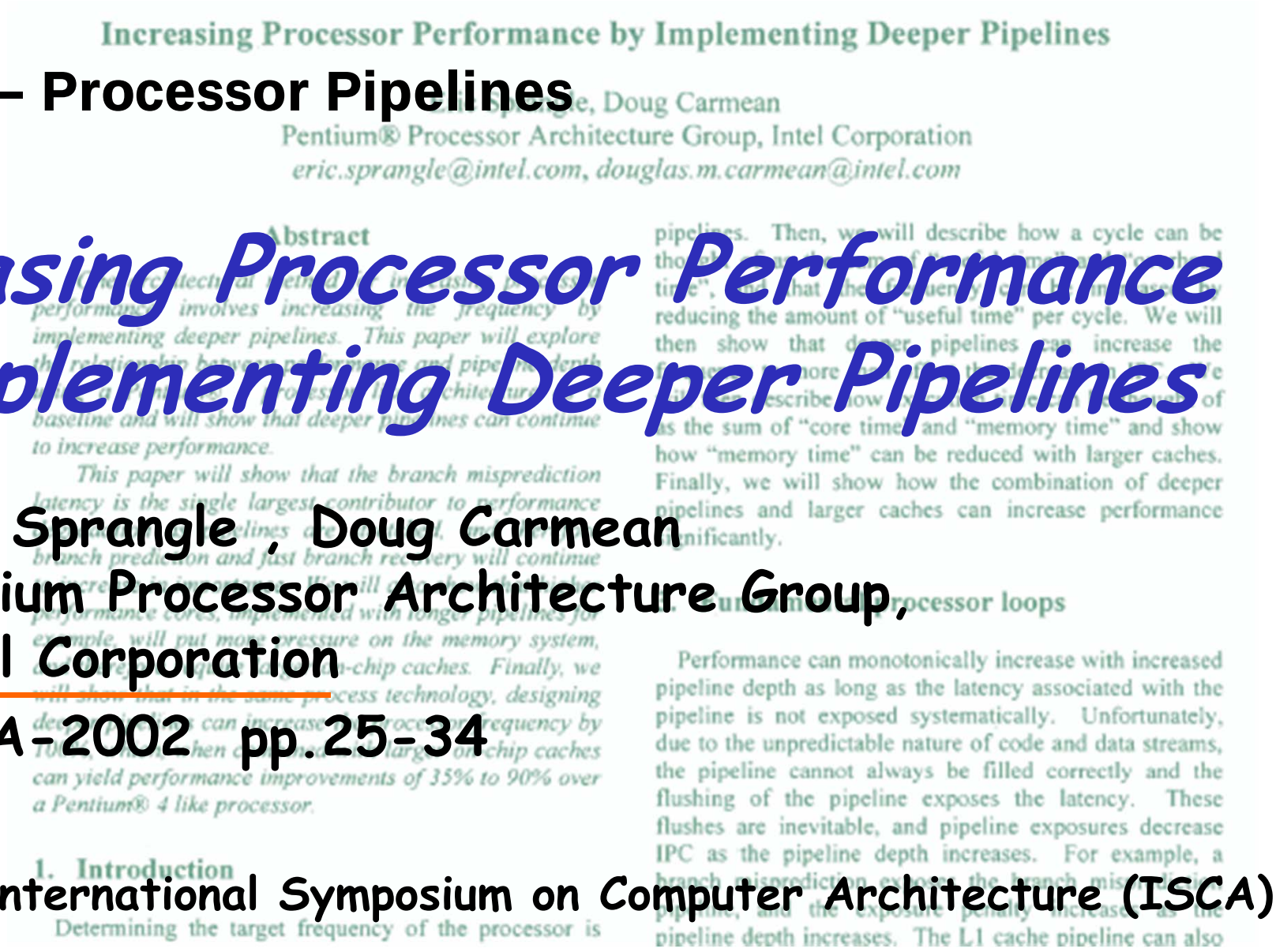
```
int flag = 1;

int foo(){
    while(flag);
}
```

スカラプロセッサのパイプラインと制約ループ





Session 1 – Processor Pipelines


Increasing Processor Performance by Implementing Deeper Pipelines

e, Doug Carmean

Pentium® Processor Architecture Group, Intel Corporation

eric.sprangle@intel.com, douglas.m.carmean@intel.com

Abstract

performance involves increasing the frequency by implementing deeper pipelines. This paper will explore

to increase performance.

This paper will show that the branch misprediction latency is the single largest contributor to performance

branch prediction and fast branch recovery will continue

example, will put more pressure on the memory system,

will show that in the same process technology, designing

can yield performance improvements of 35% to 90% over a Pentium® 4 like processor.

1. Introduction

Determining the target frequency of the processor is

pipelines. Then, we will describe how a cycle can be

reducing the amount of "useful time" per cycle. We will then show that deeper pipelines can increase the

as the sum of "core time" and "memory time" and show how "memory time" can be reduced with larger caches. Finally, we will show how the combination of deeper pipelines and larger caches can increase performance

ocessor loops

Performance can monotonically increase with increased pipeline depth as long as the latency associated with the pipeline is not exposed systematically. Unfortunately, due to the unpredictable nature of code and data streams, the pipeline cannot always be filled correctly and the flushing of the pipeline exposes the latency. These flushes are inevitable, and pipeline exposures decrease IPC as the pipeline depth increases. For example, a

pipeline depth increases. The L1 cache pipeline can also

Increasing Processor Performance by Implementing Deeper Pipelines

Eric Sprangle , Doug Carmean
Pentium Processor Architecture Group,
Intel Corporation

ISCA-2002 pp.25-34

International Symposium on Computer Architecture (ISCA)

研究の背景と目的

- プロセッサの動作周波数の決定はプロセッサ設計者の直面する本質的な課題となっている.
- パイプラインが深くなると，設計の複雑さと工程は劇的に増加する.

- パイプラインの深さとキャッシュサイズの関数として，プロセッサ性能を予測するモデルを構築して，シミュレーションにより性能を評価する.
- Pentium 4プロセッサをベースラインとして，深いパイプラインが性能向上につながることを示す.

Simulated 2GHz Pentium 4 like processor config

Core
- 3-wide fetch/retire
- 2 ALUs (running at 2x frequency)
- 1 load and store / cycle
- In-order allocation/de-allocation of buffers
- 512 rob entries, load buffers and store buffers

Memory System
- 64 kB/8-way I-cache
- 8 kB/4-way L1 D-cache, 2 cycle latency
- 256 kB/8-way unified L2 cache, 12 cycle latency
- 3.2 GB/sec memory system, 165ns average latency
- Perfect memory disambiguation
- 16 kB Gshare branch predictor
- Streaming based hardware prefetcher

Skeleton という実行駆動のシミュレータを用いて評価する.

Simulated Benchmark Suites

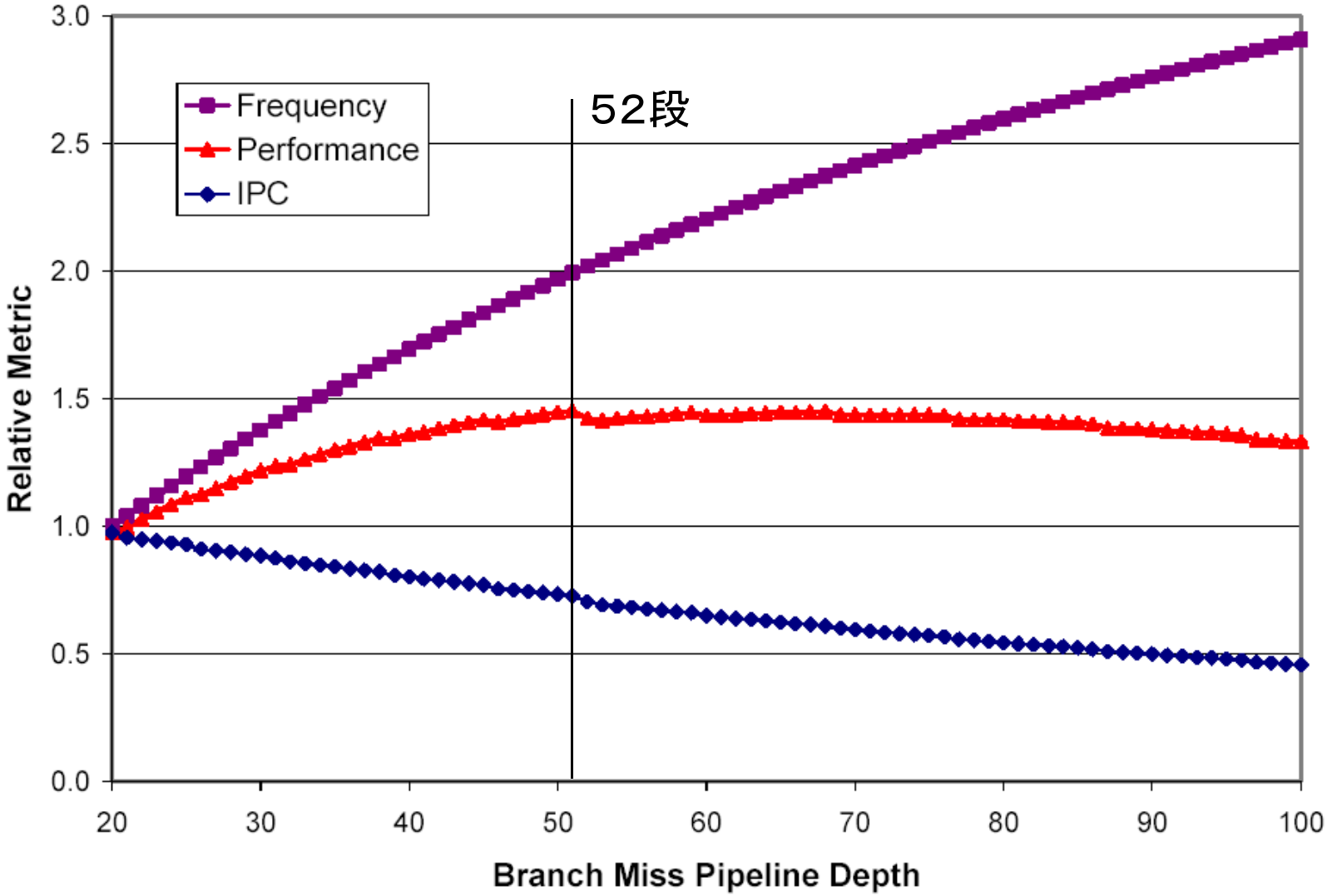
Suite	Number of Benchmarks	Description
SPECint95	8	spec.org
Multimedia	22	speech recognition, mpeg, photoshop, ray tracing, rsa
Productivity	13	sysmark2k internet/business/ productivity, Premiere
SPECfp2k	10	spec.org
SPECint2k	12	spec.org
Workstation	14	CAD, rendering
Internet	12	webmark2k, specjbb

Overhead of the pipeline

- Conservative ASIC design
  - Clock skew + jitter = 51ps
  - Standard 0.18um process, flop overhead is 3 FO4 = 75ps
  - Pipeline overhead = 51ps + 75ps = 125ps
- Custom design
  - Most of clock skew and jitter overhead can be hidden.
  - Pipeline overhead = 75ps
- Extreme custom design
  - Sub-50ps at the cost of a much larger design cost
- Pentium 4 overhead
  - Pipeline overhead = 90ps
  - Use 90ps as a baseline overhead time

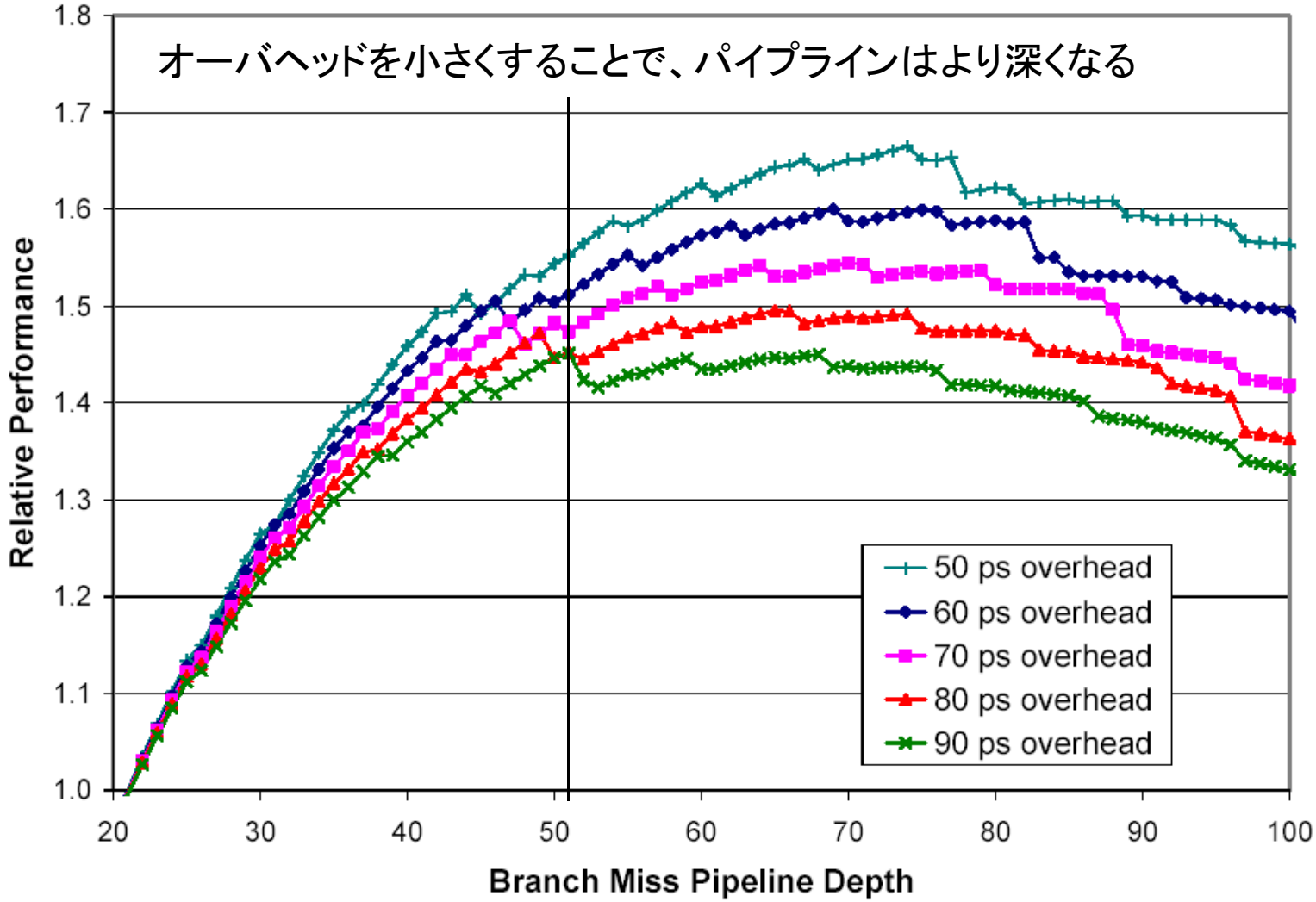
パイプライン段数を変化させた動作周波数およびIPCと性能

- パイプラインが52段で、動作周波数が2倍になるまで性能が向上



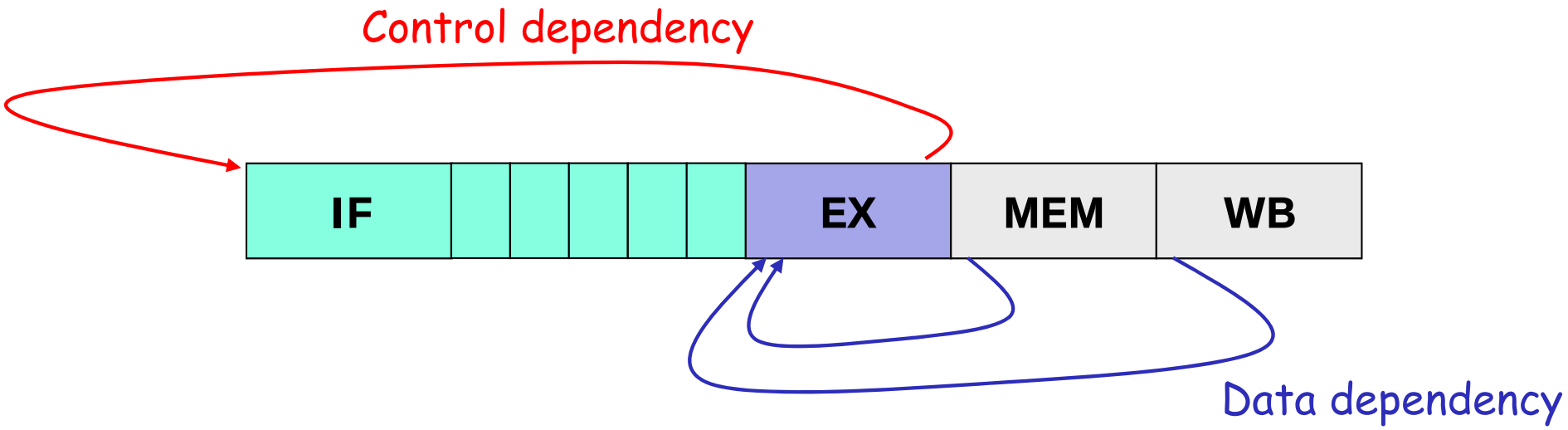
パイプライン段数とオーバヘッドを変化させた時の性能

- 評価では，パイプライン段数に影響を受けず一定のオーバヘッドを想定
- 2GHz のパイプラインピッチ 500ps, Pentium 4のオーバヘッドは 90ps



Super pipelined processor（スーパーパイプライン）

- A super pipelined processor has split the main computational pipeline into more stages.



プロセッサの命令パイプラインの例

Basic Pentium® III Processor Misprediction Pipeline

1	2	3	4	5	6	7	8	9	10
Fetch	Fetch	Decode	Decode	Decode	Rename	ROB Rd	Rdy/Sch	Dispatch	Exec

Basic Pentium® 4 Processor Misprediction Pipeline

1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20
TC Nxt IP		TC Fetch		Drive	Alloc	Rename		Que	Sch	Sch	Sch	Disp	Disp	RF	RF	Ex	Flgs	Br Ck	Drive

The Microarchitecture of the Pentium® 4, Intel Technical Report

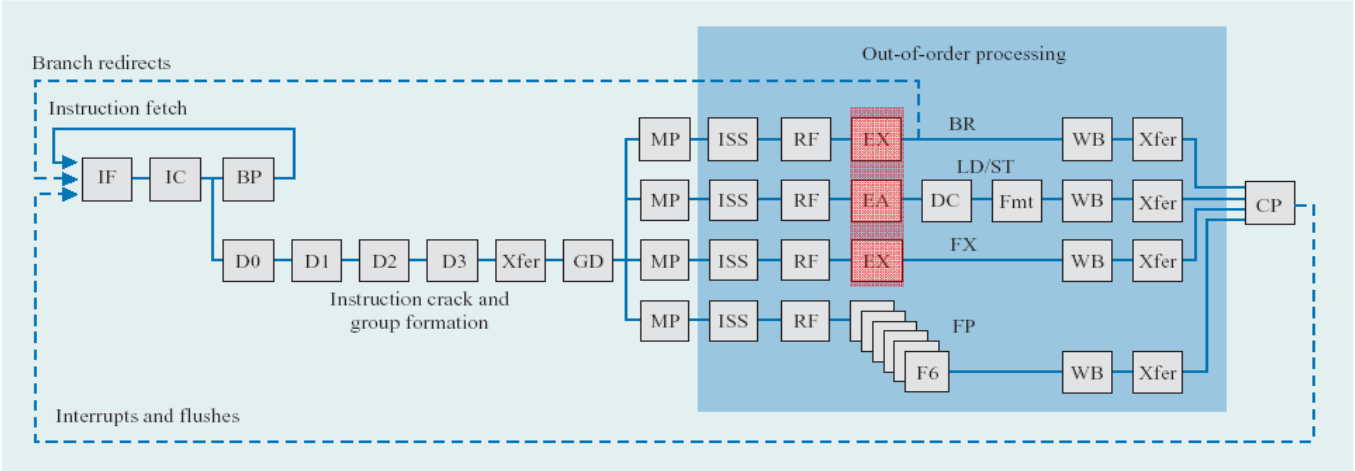
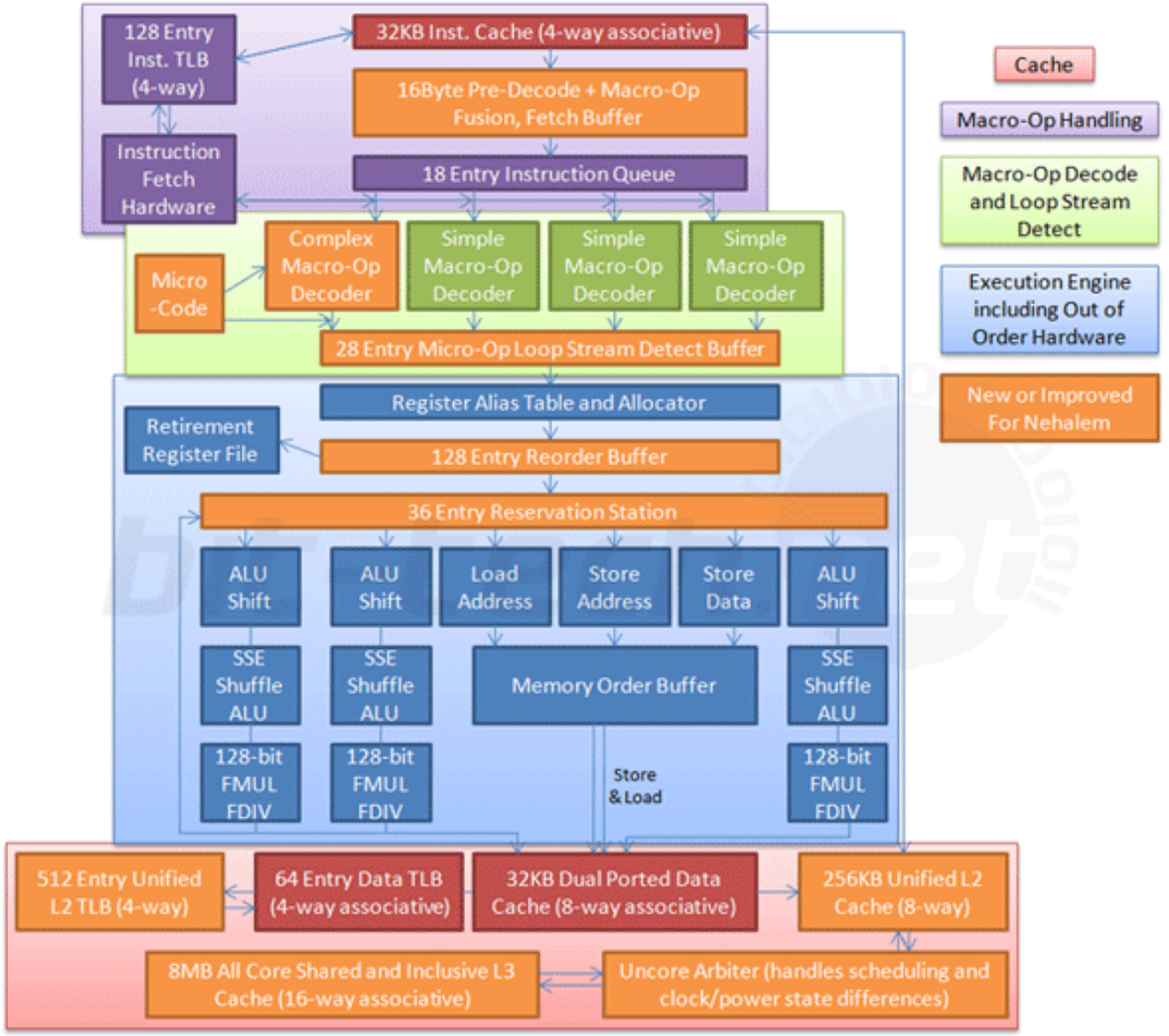


Figure 4

POWER4 instruction execution pipeline.

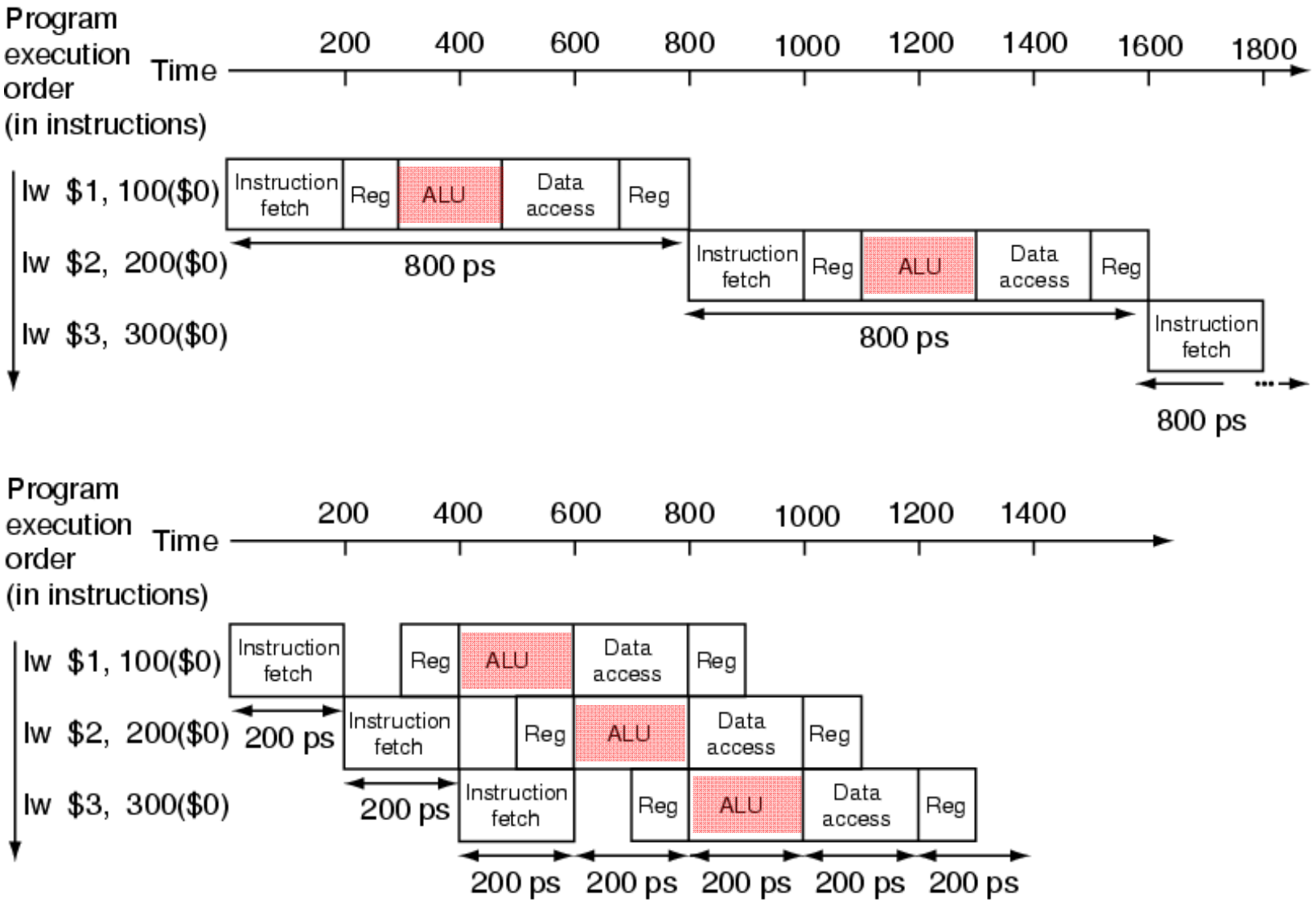
POWER4 System Microarchitecture, IBM Journal

プロセッサの命令パイプラインの例



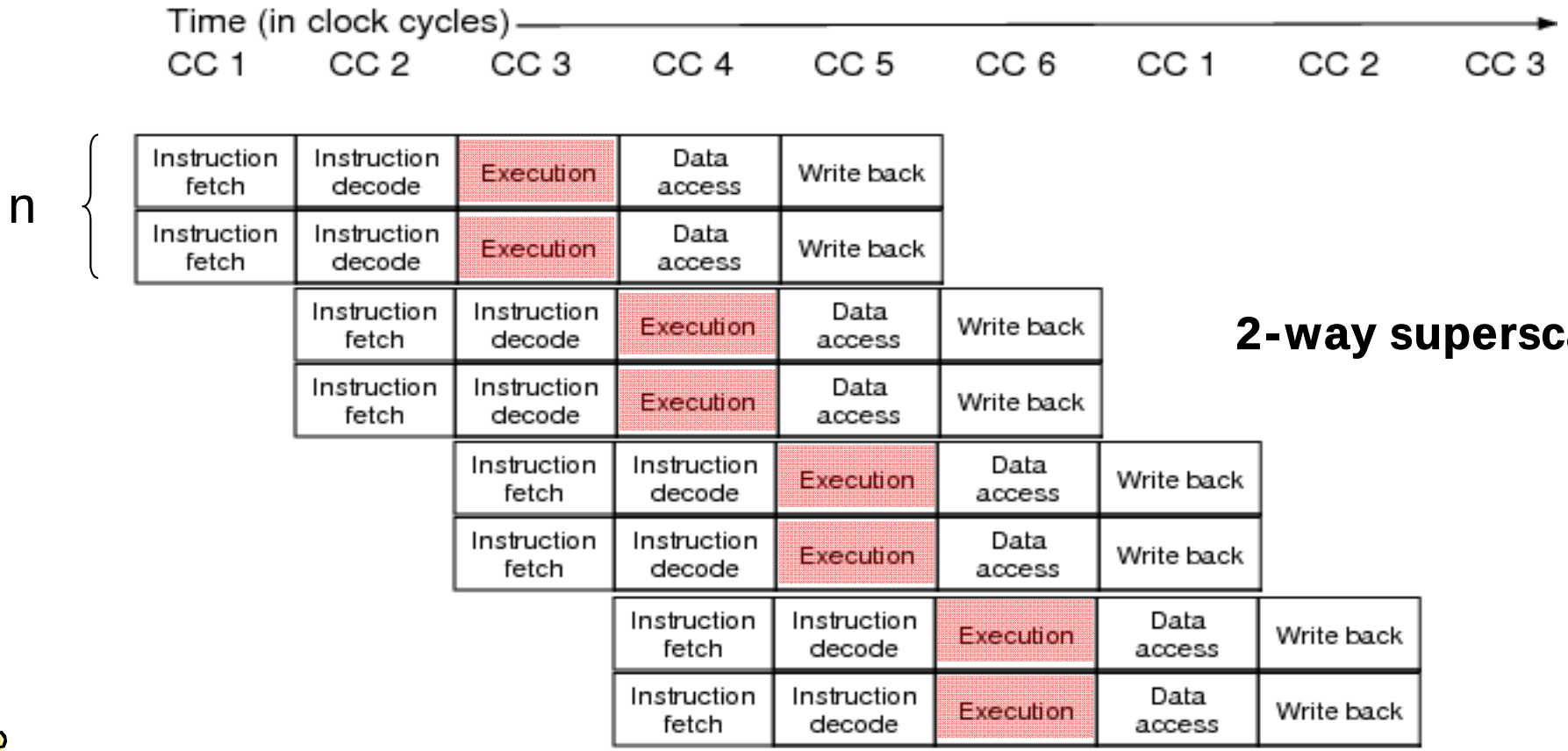
Intel Core i7 Nehalem Architecture (14 stages), bit-tech

Single Cycle and Pipelined Processor

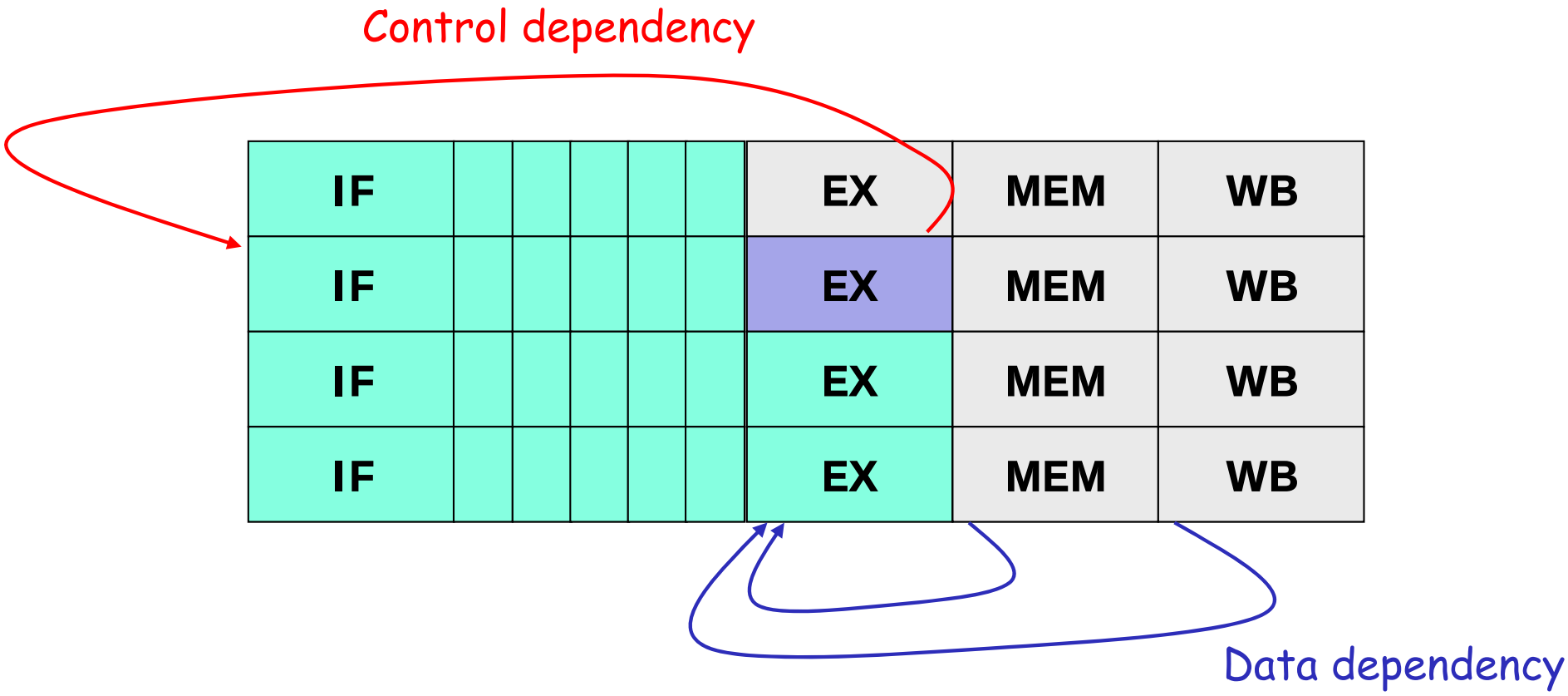


スーパースカラプロセッサと命令レベル並列性

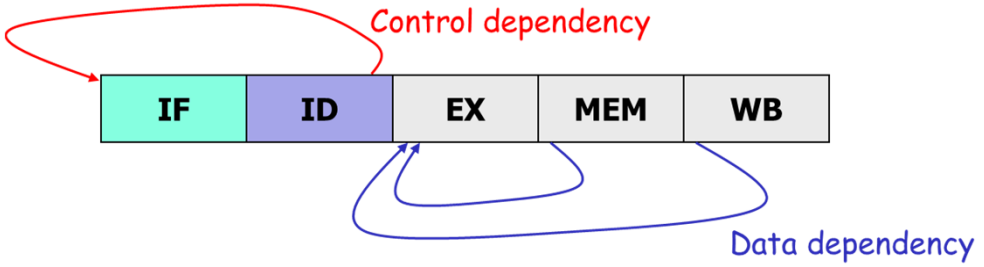
- 複数のパイプラインを利用して IPC (instructions per cycle) を 1以上に引き上げる，複数の命令を並列に実行
  - n-way スーパースカラ



スーパースカラプロセッサと制約ループ



Conventional 5 stage pipeline



プロセッサが命令を処理するための5つのステップの修正

- IF (Instruction Fetch)
  メモリから命令をフェッチする.
- ID (Instruction Decode)
  命令をデコード(解読)しながら, レジスタの値を読み出す.
  分岐命令である可能性を考慮し, 読み出されたレジスタの間で一致比較を行う. 命令のオフセットフィールドを符号拡張し, インクリメントされたPCに符号拡張されたオフセットを足し合わせて分岐先のアドレスを計算する. 条件が成立した場合には分岐先アドレスをPCにセットして, このステージで分岐命令を完了させる.
- EX (Execution)
  命令操作の実行またはアドレスの生成を行う.
- MEM (Memory Access)
  必要であれば, データ・メモリ中のオペランドにアクセスする.
- WB (Write Back)
  必要であれば, 結果をレジスタに書き込む.

戦略4：遅延分岐 (delayed branch), MIPSが採用

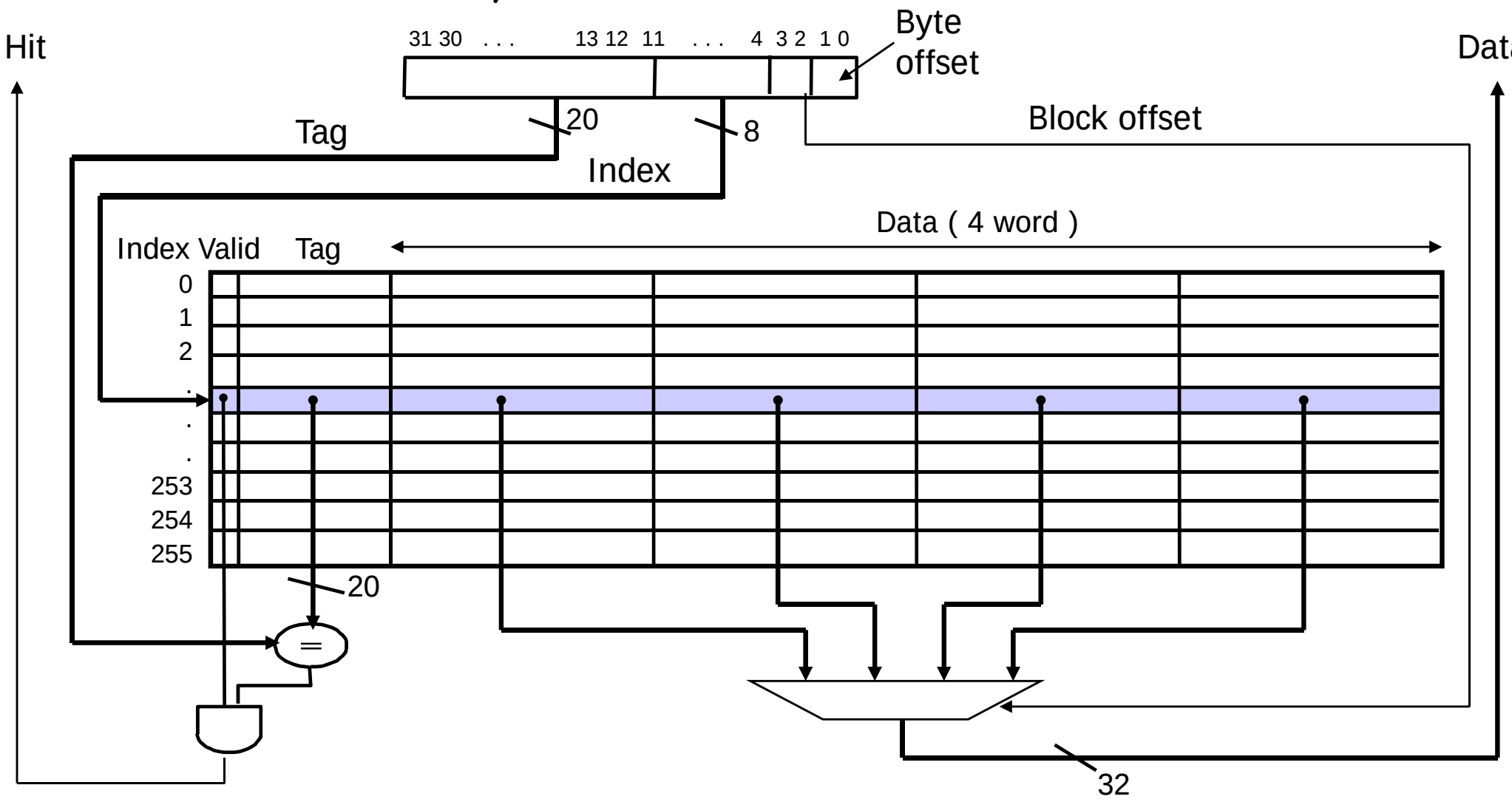
- 分岐命令の後続の幾つかの命令を実行した後に，分岐する．1サイクルの遅延を持つ命令実行順は次の通り．
  - 分岐命令を実行
  - 分岐命令の次アドレスの命令を実行
  - 分岐成立では，飛び先アドレスの命令を実行
    （不成立では，分岐命令の次の次のアドレスの命令を実行）
- 分岐命令の後続の幾つかの命令を実行した後に分岐．分岐命令によるストールは生じない．

Untaken分岐命令	IF	ID	EX	MEM	WB				
分岐遅延命令 ($i+1$)		IF	ID	EX	MEM	WB			
命令 $i+2$			IF	ID	EX	MEM	WB		
命令 $i+3$				IF	ID	EX	MEM	WB	
命令 $i+4$					IF	ID	EX	MEM	WB

Taken分岐命令	IF	ID	EX	MEM	WB				
分岐遅延命令 ($i+1$)		IF	ID	EX	MEM	WB			
分岐先			IF	ID	EX	MEM	WB		
分岐先 + 1				IF	ID	EX	MEM	WB	
分岐先 + 2					IF	ID	EX	MEM	WB

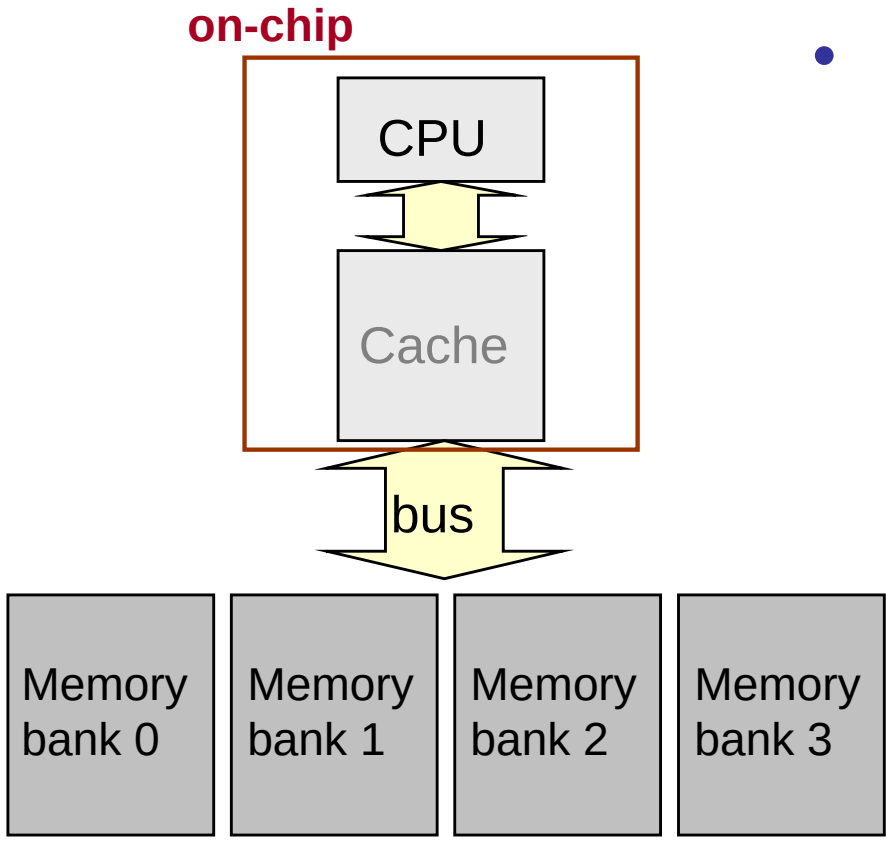
Multiword Block Direct Mapped Cache

- Four words/block, cache size = 1K words



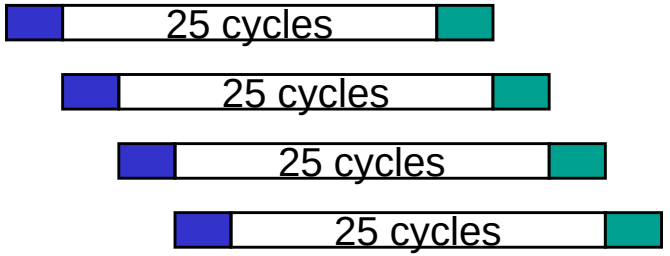
What kind of locality are we taking advantage of?

Interleaved（インターリーブ）Memory Organization



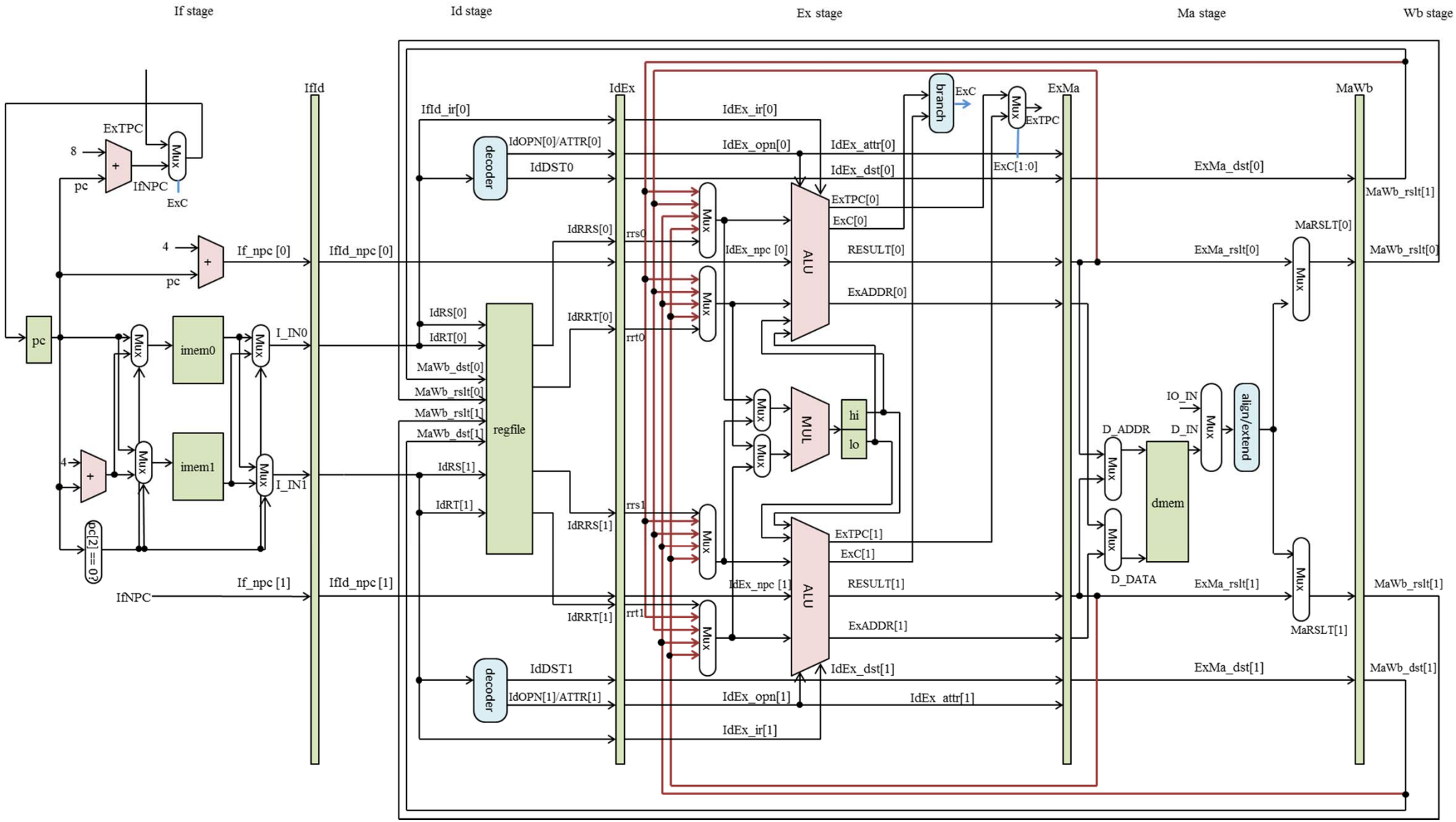
With **parallelism**

- For a block size of **four words** with **interleaved memory (4 banks)**
  - **1** cycle to send 1st address
  - **25 + 3 = 28** cycles to read DRAM
  - **1** cycle to return last data word
  - **30 total clock cycles** miss penalty

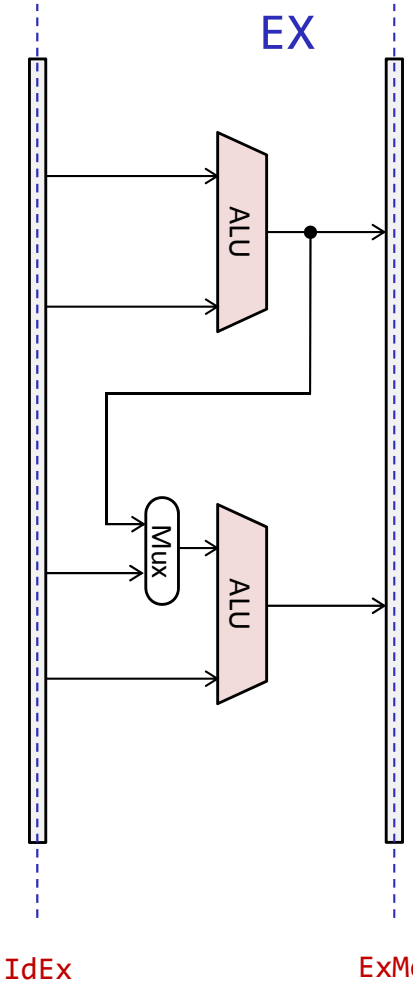


- Number of bytes transferred per clock cycle (bandwidth) for a single miss
  - **(4 x 4) / 30 = 0.533 bytes per clock**

スーパースカラプロセッサ（インタリーブ命令メモリ版）



Cascade ALU Architecture



IEICE TRANS. ELECTRON., VOL.E84–C, NO.2 FEBRUARY 2001

PAPER Special Issue on Low-Power High-Performance VLSI Processors and Technologies

A Cascade ALU Architecture for Asynchronous Super-Scalar Processors

Motokazu OZAWA[†a)], *Student Member*, Masashi IMAI[†], *Nonmember*, Yoichiro UENO[††], Hiroshi NAKAMURA[†], *and* Takashi NANYA[†], *Regular Members*

SUMMARY Wire delays, instead of gate delays, are moving into dominance in modern VLSI design. Current synchronous processors have the critical path not in the ALU function but in the cache access. Since the cache performance enhancement is limited by the memory access delay which mainly consists of wire delays, a reduction in gate delays may no longer imply any enhancement in processor performance. To solve this problem, this paper presents a novel architecture, called the Cascade ALU. The Cascade ALU allows super-scalar processors with future technologies to move the critical path into the ALU part. Therefore the Cascade ALU can enjoy the expected progress in future device speed. Since the delay of the Cascade ALU varies depending on the executed instructions, an asynchronous system is shown to be suitable for implementing the Cascade ALU. However an asynchronous system may have a large handshake overhead, this paper also presents an asynchronous Fine Grain Pipeline technique that hides the handshake overhead. Finally, this paper bring about a situation where the ALU latency instead of the cache access latency assumes the critical path. Thus, a decrease in the ALU latency thanks to a future decrease in gate delays can directly cause a decrease in the cycle time.

This paper presents a novel architecture, called the Cascade ALU architecture. The Cascade ALU architecture allows super-scalar processors with future technologies to have the critical path for performance in the ALU part. Therefore, the Cascade ALU architecture can enjoy any further progress in device speed for an enhancement in processor performance.

In the proposed Cascade ALU architecture, the instruction execution latency may vary depending on

A Cascade ALU Architecture for Asynchronous Super-Scalar Processors, IEICE transactions on electronics
IEICE transactions on electronics 84(2), 229-237, 2001-02-01