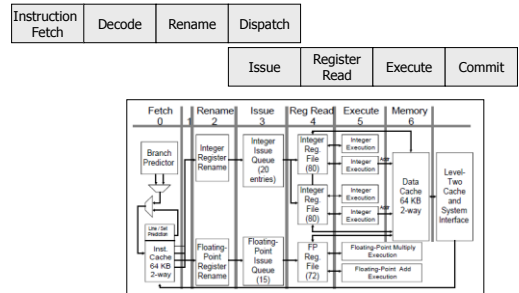


計算機アーキテクチャ 第二 (O)

データ値予測, データフロー実行モデル

1

アウトオブオーダー実行プロセッサの命令パイプライン



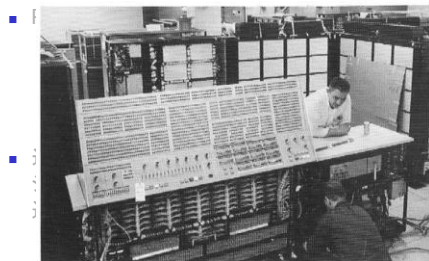
The Alpha 21264 Microprocessor Architecture
R. E. Kessler, E. J. McLellan, and D. A. Webb, Compaq Computer Corporation

Adapted from *Computer Organization and Design*, Patterson & Hennessy, © 2005

2

命令発行機構: Tomasuloのアプローチ

- IBM 360/91 の浮動小数点ユニットでは、アウトオブオーダー実行のための洗練された方式が採用されていた。



Installation of the IBM 360/91 in the Columbia Computer Center machine room in February or March 1969. Photo: AIS archive.
Adapted from *Computer Organization and Design*, Patterson & Hennessy, © 2005

3

命令発行機構: Tomasuloのアプローチ 1967

- IBM 360/91 の浮動小数点ユニットでは、アウトオブオーダー実行のための洗練された方式が採用されていた。
- Robert Tomasulo によって発明されたこの手法では
 - (1) レジスタリネーミングを導入してWAWハザードとWARハザード(偽のデータ依存)を排除
 - (2) 命令が必要とするオペランドがいつ利用できるかを探知し, RAW(Read after Write)ハザードを最小化
- 近年のプロセッサでは, この手法のさまざまなバリエーションが採用されているが, これら2つの重要な概念は共通の特徴

Adapted from *Computer Organization and Design*, Patterson & Hennessy, © 2005

4

マルチレベル・ストライ値予測機構による命令レベル並列性の向上
(JSPP 1999)

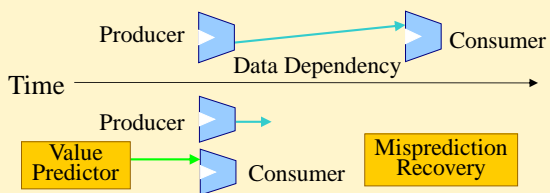
研究の背景

- 真のデータ依存関係が命令レベル並列性を制限
- 生産者から消費者へのデータの流れを解消する技術として値予測

Adapted from *Computer Organization and Design*, Patterson & Hennessy, © 2005

研究の背景

- 真のデータ依存関係が命令レベル並列性を制限
- 生産者から消費者へのデータの流れを解消する技術として値予測



Adapted from Computer Organization and Design, Patterson & Hennessy, © 2005

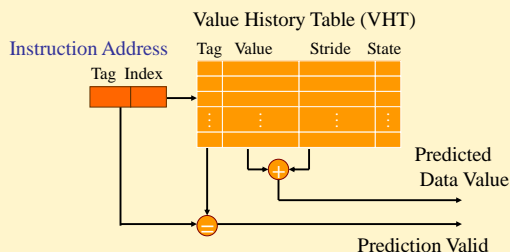
関連研究: 値生成のアルゴリズム

- Last-value予測
 - 最も近い過去に得られた値を予測値
- スライド値予測
 - 最も近い過去に得られた2回の値の差分 Stride と、Last-value の和を予測値
- 2レベル値予測
 - 過去のn個の履歴の中からひとつを選択
- ハイブリッド値予測
 - 複数のアルゴリズムから選択

Adapted from Computer Organization and Design, Patterson & Hennessy, © 2005

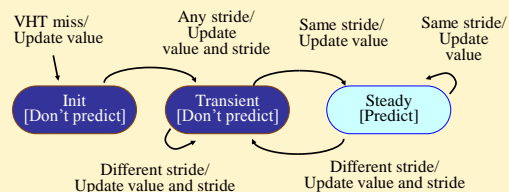
スライド値予測機構

$$\text{Predicted Value} = \text{Last-value} + \text{Stride}$$



Adapted from Computer Organization and Design, Patterson & Hennessy, © 2005

スライド値予測機構 (cont.)



Stateフィールドの推移と予測アルゴリズム

Adapted from Computer Organization and Design, Patterson & Hennessy, © 2005

スライド値予測機構 (cont.)

1～5の値を繰り返す下の例

Value :	1	2	3	4	5	1	2	3	4	5 ..
Stride:		1	1	1	1	-4	1	1	1	1 ..
Result:	NP	NP	NP	H	H	M	NP	NP	H	H ..
State :	I	T	S	S	S	T	T	S	S	S

NP=No Predict, H=Hit, M=Miss

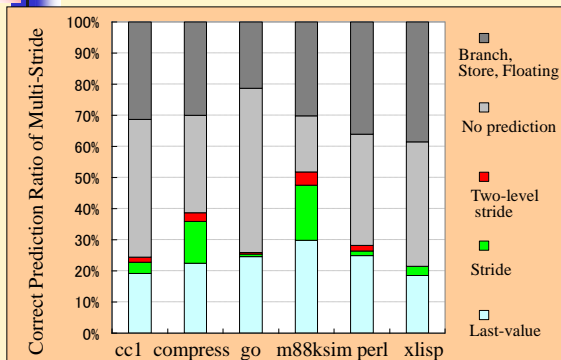
I=Initial, T=Transient, S=Steady

短い間隔でストライドが変化する場合に予測精度が低下、予測成功率40%

Adapted from Computer Organization and Design, Patterson & Hennessy, © 2005

11

Multi-stride値予測の予測成功率



Adapted from Computer Organization and Design, Patterson & Hennessy, © 2005

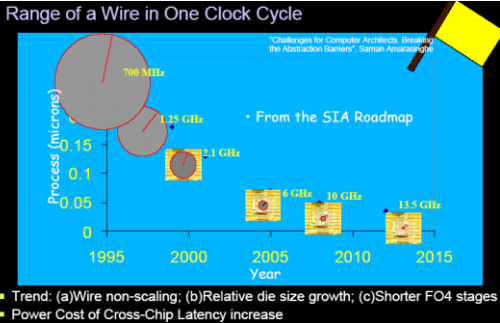
値予測ミスの回数とミス率

Program	Last-value	Stride	2-level Str.
cc1	10734(.05%)	33591(.77%)	13287(.68%)
compress	2679(.02%)	3094(.05%)	1489(.01%)
go	1934(.01%)	4827(.37%)	593(.11%)
m88ksim	16262(.04%)	43832(.20%)	29041(.53%)
perl	1245(.01%)	1544(.11%)	2950(.01%)
xlisp	1904(.02%)	2950(.24%)	9(.05%)

Adapted from Computer Organization and Design, Patterson & Hennessy, © 2005

13

Range of a Wire in One Clock Cycle



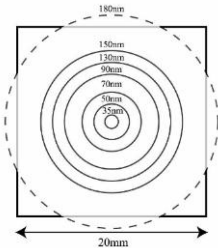
MICRO-36 (2003, San Diego, CA) Keynote Kerry Bernstein Senior Technical Staff Member IBM T.J. Watson Research Center

Lecture Slide, Kenji KISE TokyoTech

14

配線遅延の克服

- クロックサイクルの間に信号が伝わるチップ内の範囲
 - 8 FO4 を1クロック
 - 20 x 20mm のチップ
- 35nm のテクノロジー
 - 1クロックで信号を伝達可能な範囲は全体の1%
 - 距離の離れた2点間では、数十サイクルの遅延
- 配線遅延を考慮して方式を検討



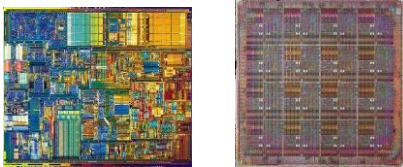
[1] S.W. Keckler, Doug Burger, C.R. Moore, R. Nagarajan, K. Sankaralingam, V. Agarwal, M.S. Hrishikesh, N. Ranganathan, and P. Shivakumar, A Wire-Delay Scalable Microprocessor Architecture for High Performance Systems, International Solid-State Circuits Conference (ISSCC), pp.1068-1069, February 2003.

Lecture Slide, Kenji KISE TokyoTech

15

タイルアーキテクチャとは

- 小さいサイズの機能ブロック(タイル)を規則的に敷きつめることで高速なプロセッサを構成する方式
 - タイルのサイズを小さくすることで、タイルの内部で発生する配線遅延の問題を軽減
 - 近くに配置されているタイル間でのみデータの送受信をおこなうことで、タイル間の通信遅延を軽減
 - 同じ構成のタイルを複製して、設計と検証の作業の簡略化



スーパースカラのPentium 4プロセッサと、タイルアーキテクチャのRawプロセッサ

Lecture Slide, Kenji KISE TokyoTech

16

タイル

- 「壁または床などに張る小片状の薄板。陶磁器が一般的」広辞苑
- 「Tiles are flat, square pieces of baked clay, carpet, cork, or other substance, which are fixed as a covering onto a floor or wall.」Collins COBUILD English Dictionary

Lecture Slide, Kenji KISE TokyoTech

17

TRIPSプロセッサ

- テキサス大学におけるタイルプロセッサ
 - 単純な構成の計算ノード
 - 独自の実行モデル
 - 挑戦的



Lecture Slide, Kenji KISE TokyoTech

18

Explicit Dataflow Graph Execution (EDGE)

Explicit Data Graph Execution

- Defined by two key features

- Program graph is broken into sequences of *blocks*
 - Basic blocks, hyperblocks, or something else
 - Blocks commit atomically or not at all - a block never partially executes
- Within a block, ISA support for direct producer-to-consumer communication
 - No shared named registers within a block (point-to-point dataflow edges only)
 - The block's dataflow graph (DFG) is explicit in the architecture
 - Caveat: memory is still a shared namespace (bane of prior dataflow machines)

Lecture Slide, Kenji KISE TokyoTech

Lecture note for CS352 Computer Systems Architecture by Prof. S.W. Keckler

19

Block Compilation

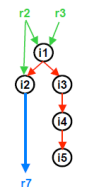


Intermediate Code

```
i1) add r1, r2, r3
i2) add r7, r2, r1
i3) ld r4, (r1)
i4) add r5, r4, #1
i5) beqz r5, 0xdead
```

- Inputs (r2, r3)
- Temporaries (r1, r4, r5)
- Outputs (r7)

Data flow graph



Compiler Transforms

Lecture Slide, Kenji KISE TokyoTech

Lecture note for CS352 Computer Systems Architecture by Prof. S.W. Keckler

20

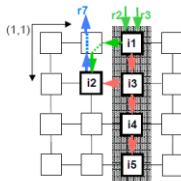
Block Mapping



Data flow graph



Mapping onto array



Scheduler
Scheduling is an optimization, not necessary for correctness

Lecture Slide, Kenji KISE TokyoTech

Lecture note for CS352 Computer Systems Architecture by Prof. S.W. Keckler

21

TRIPS Block Example

RISC code

```
ld R3, 4(R2)
add R4, R1, R3
st R4, 4(R2)
addi R5, R4, #2
beqz R4, Block3
```

TIL (operand format)

```
.bbegin block1
read $t1, $g1
read $t2, $g2
ld $t3, 4($t2)
add $t4, $t1, $t3
st $t4, 4($t2)
addi $t5, $t4, 2
teqz $t6, $t4
b.t<$t6> block3
b.f<$t6> block2
write $g5, $t5
.bend block1

.bbegin block2 ...
```

- Read target format
- Predicated instructions
- LD/ST sequence numbers
- Target fanout with movs
- Block outputs fixed (3 in this example)

TASL (target format)

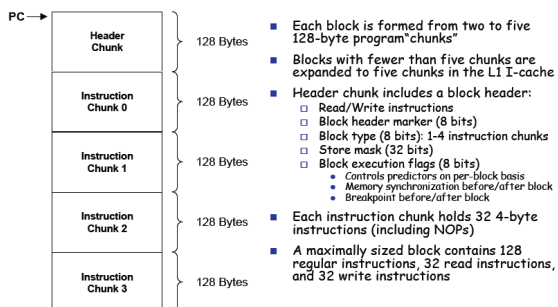
```
[R1] $g1 [2]
[R2] $g2 [1] [4]
[1] ld L:1 4 [2]
[2] add [3] [4]
[3] mov [5] [6]
[4] st S:2 4
[5] addi 2 [W1]
[6] teqz [7] [8]
[7] b.t block3
[8] b.f block2
[W1] $g5
```

Lecture Slide, Kenji KISE TokyoTech

Lecture note for CS352 Computer Systems Architecture by Prof. S.W. Keckler

22

TRIPS Block Format

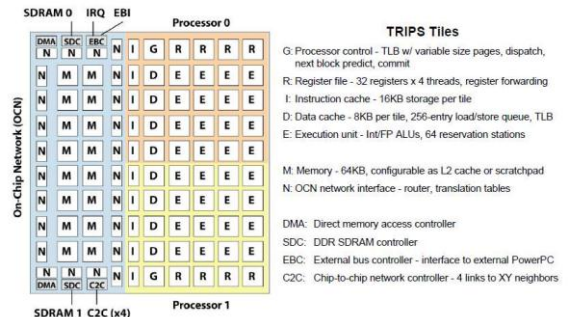


Lecture Slide, Kenji KISE TokyoTech

Lecture note for CS352 Computer Systems Architecture by Prof. S.W. Keckler

23

TRIPS Tile-level Microarchitecture



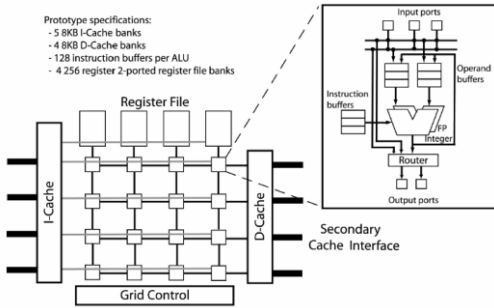
Lecture Slide, Kenji KISE TokyoTech

Lecture note for CS352 Computer Systems Architecture by Prof. S.W. Keckler

24

4×4の計算ノードを持つTRIPSプロセッサ, 計算ノード構成

Prototype specifications:
 - 5 8KB I-Cache banks
 - 4 8KB D-Cache banks
 - 128 instruction buffers per ALU
 - 4 256 register 2-ported register file banks



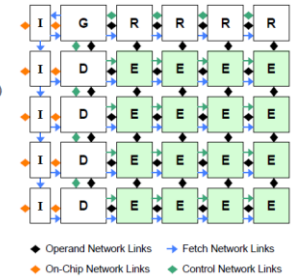
Lecture Slide, Kenji KISE TokyoTech

Lecture note for CS352 Computer Systems Architecture by Prof. S.W. Keckler

25

TRIPS Processor Tiles

- Partition all major structures into banks, distribute, and interconnect
- Execution Tile (E)
 - 64-entry Instruction Queue bank
 - Single-issue execute pipeline
- Register Tile (R)
 - 32-entry Register bank (per thread)
- Data Tile (D)
 - 8KB Data Cache bank
 - LSQ and MHU banks
- Instruction Tile (I)
 - 16KB Instruction Cache bank
- Global Control Tile (G)
 - Tracks up to 8 blocks of insts
 - Branch prediction & resolution logic



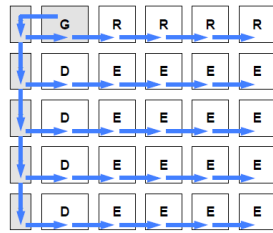
Lecture Slide, Kenji KISE TokyoTech

Lecture note for CS352 Computer Systems Architecture by Prof. S.W. Keckler

26

Block Fetch

- Fetch commands sent to each Instruction Cache bank
- The fetch pipeline is from 4 to 11 stages deep
- A new block fetch can be initiated every 8 cycles
- Instructions are fetched into Instruction Queue banks (chosen by the compiler)
- EDGE ISA allows instructions to be fetched out-of-order

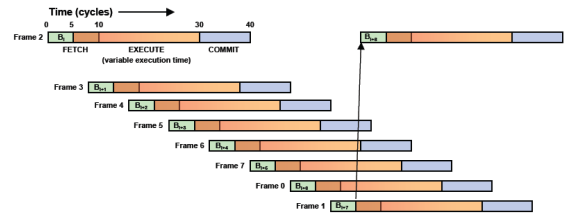


Lecture Slide, Kenji KISE TokyoTech

Lecture note for CS352 Computer Systems Architecture by Prof. S.W. Keckler

27

Block Execution Timeline



- Execute/commit overlapped across multiple blocks
- G-tile manages frames as a circular buffer
 - D-morph: 1 thread, 8 frames
 - T-morph: up to 4 threads, 2 frames each

Lecture Slide, Kenji KISE TokyoTech

Lecture note for CS352 Computer Systems Architecture by Prof. S.W. Keckler

28

Processor Performance

Name	TRIPS Speedup	Alpha IPC	TRIPS IPC	TRIPS Inst/Block	Description
a2time	5.05	0.81	4.05	77	Control, integer math
bezier	3.30	1.05	3.20	76	Bezier curve, fixed-point math
dct8x8	2.66	1.70	4.70	90	2D discrete cosine transform
matrix	3.30	1.68	4.05	72	Matrix multiply
sha	0.92	2.28	2.10	80	Secure hash (mostly sequential algorithm)
vadd	1.92	3.04	6.51	74	Vector add (limited by load/store bandwidth)

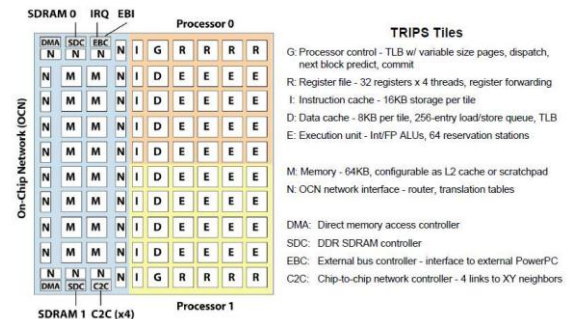
Simulated on TRIPS and Alpha 21264 cycle simulators
 Alpha compilation with GEM compiler and maximum opts (O4 and tuned for 21264)
 TRIPS compilation with in-development compiler plus some hand-tuning
 Speedup measured by comparing Alpha cycles to TRIPS cycles

Lecture Slide, Kenji KISE TokyoTech

Lecture note for CS352 Computer Systems Architecture by Prof. S.W. Keckler

29

TRIPS Tile-level Microarchitecture



Lecture Slide, Kenji KISE TokyoTech

Lecture note for CS352 Computer Systems Architecture by Prof. S.W. Keckler

30

計算機アーキテクチャ 第二 (O)

マルチコア、メニーコアへ

31

Growth in clock rate of microprocessors

From CAQA 5th edition

32

Adapted from Computer Organization and Design, Patterson & Hennessy, © 2005

Growth in processor performance

From CAQA 5th edition

33

Adapted from Computer Organization and Design, Patterson & Hennessy, © 2005

ムーアの法則によるトランジスタ数の増加

ムーアの法則

チップで利用できるトランジスタの数は2年間で2倍に増加する。

プロセッサ	出荷年	トランジスタ数
4004	1971	2,250
8008	1972	2,500
8080	1974	5,000
8086	1978	29,000
286	1982	120,000
386™ processor	1985	275,000
486™ DX processor	1989	1,180,000
Pentium® processor	1993	3,100,000
Pentium II processor	1997	7,500,000
Pentium III processor	1999	24,000,000
Pentium 4 processor	2000	42,000,000



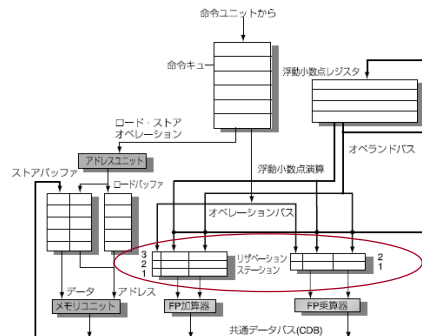
ムーアの法則に従ってトランジスタ数が増加してきた。今後も同様の増加が見込まれる。

出典: Intel社, <http://www.intel.com/research/silicon/mooreslaw.htm>

Adapted from Computer Organization and Design, Patterson & Hennessy, © 2005

34

Tomasuloのアプローチ



Adapted from Computer Organization and Design, Patterson & Hennessy, © 2005

35

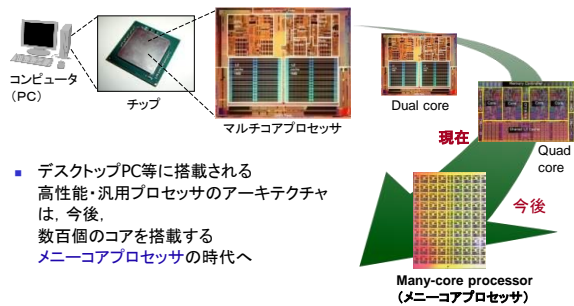
ボラックの法則

- プロセッサの性能は、複雑性の平方根に比例する

Adapted from Computer Organization and Design, Patterson & Hennessy, © 2005

36

マルチコア(2個～数10個)からメニーコアへ



Adapted from *Computer Organization and Design*, Patterson & Hennessy, © 2005

37

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

Adapted from *Computer Organization and Design*, Patterson & Hennessy, © 2005

38