

## 計算機アーキテクチャ特論 (Advanced Computer Architectures)

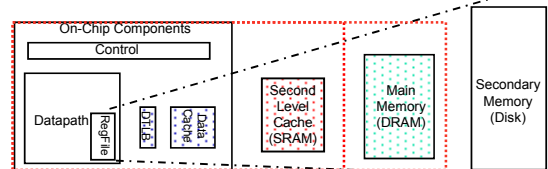
### 4. キャッシュの実装と評価

吉瀬 謙二 計算工学専攻  
kise\_at\_cs.titech.ac.jp www.arch.cs.titech.ac.jp  
W831 講義室 木曜日 15:05 - 16:35

1

## Memory Hierarchy

- By taking advantage of **the principle of locality** (局所性)
  - Present **much memory** in the **cheapest technology**
  - at the **speed of fastest technology**

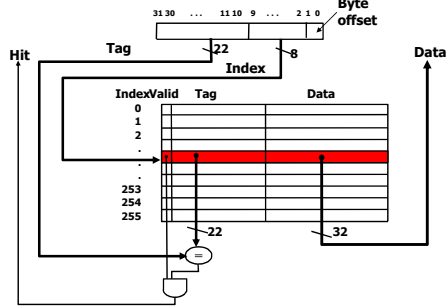


Speed (%cycles): $\frac{1}{2}$ s	1	9	90
Size (bytes): 100's	1KB	4KB	512KB

2

### 1KB L1 Data Cache

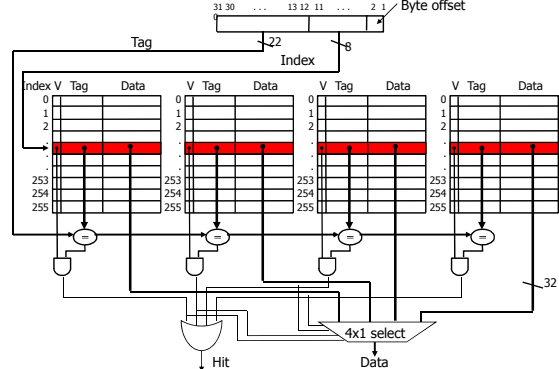
- 1KB cache
- One word/block, cache size = 256K words



3

### 4KB L2 Data Cache: 4-Way Set Associative Cache

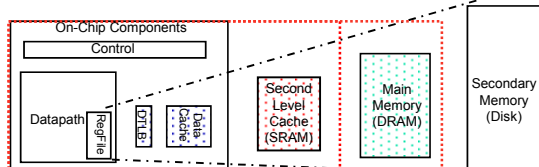
- $2^8 = 256$  sets each with four ways (each with one block)



4

## Memory Hierarchy

- L1 hit, 1 cycle for both LW/SW instruction execution
- L1 miss and L2 hit, 10 cycle for both LW/SW instruction ex.
- L1 miss and L2 miss, 100 cycle for both LW/SW instruction ex.



Speed (%cycles): $\frac{1}{2}$ s	1	9	90
Size (bytes): 100's	1KB	4KB	512KB

5

## Sample code sequence, total of 229 cycles

```

lw    $2, 80($fp)  # L1 hit, 1 cycle
nop                                # 1 cycle
sll   $2, $2, 2    # 1 cycle
move  $3, $2       # 1 cycle
lw    $2, 84($fp)  # L1 miss, L2 miss, 100 cycle
nop                                # 1 cycle
addu  $2, $3, $2   # 1 cycle
lw    $2, 0($2)    # L1 miss, L2 hit, 10 cycle
nop                                # 1 cycle
addiu $2, $2, 100  # 1 cycle
sw    $2, 0($4)    # L1 miss, L2 miss, 100 cycle
lw    $2, 80($fp)  # L1 hit, 1 cycle
sw    $2, 0($4)    # L1 miss, L2 hit, 10 cycle
    
```

6

# Benchmark program test40

Nov 08, 12 12:23	code.txt	Page 1/2	Nov 08, 12 12:23	code.txt	Page 2/2
<pre> 1 // Benchmark program test40 2 // 3 // 4 #include &lt;stdio.h&gt; 5 #include &lt;stdlib.h&gt; 6 7 #define NDM 512 // Number of data elements 8 #define NDM2 256 // Number of data elements 9 #define NDM3 128 // Number of data elements 10 #define NDM4 64 // Number of data elements 11 #define NDM5 32 // Number of data elements 12 #define NDM6 16 // Number of data elements 13 #define NDM7 8 // Number of data elements 14 #define NDM8 4 // Number of data elements 15 #define NDM9 2 // Number of data elements 16 #define NDM10 1 // Number of data elements 17 #define NDM11 0.5 // Number of data elements 18 #define NDM12 0.25 // Number of data elements 19 #define NDM13 0.125 // Number of data elements 20 #define NDM14 0.0625 // Number of data elements 21 #define NDM15 0.03125 // Number of data elements 22 #define NDM16 0.015625 // Number of data elements 23 #define NDM17 0.0078125 // Number of data elements 24 #define NDM18 0.00390625 // Number of data elements 25 #define NDM19 0.001953125 // Number of data elements 26 #define NDM20 0.0009765625 // Number of data elements 27 #define NDM21 0.00048828125 // Number of data elements 28 #define NDM22 0.000244140625 // Number of data elements 29 #define NDM23 0.0001220703125 // Number of data elements 30 #define NDM24 0.00006103515625 // Number of data elements 31 #define NDM25 0.000030517578125 // Number of data elements 32 #define NDM26 0.0000152587890625 // Number of data elements 33 #define NDM27 0.00000762939453125 // Number of data elements 34 #define NDM28 0.000003814697265625 // Number of data elements 35 #define NDM29 0.0000019073486328125 // Number of data elements 36 #define NDM30 0.00000095367431640625 // Number of data elements 37 #define NDM31 0.000000476837158203125 // Number of data elements 38 #define NDM32 0.0000002384185791015625 // Number of data elements 39 #define NDM33 0.00000011920928955078125 // Number of data elements 40 #define NDM34 0.000000059604644775390625 // Number of data elements 41 #define NDM35 0.0000000298023223876953125 // Number of data elements 42 #define NDM36 0.00000001490116119384765625 // Number of data elements 43 #define NDM37 0.000000007450580596923828125 // Number of data elements 44 #define NDM38 0.0000000037252902984619140625 // Number of data elements 45 #define NDM39 0.00000000186264514923095703125 // Number of data elements 46 #define NDM40 0.000000000931322574615478515625 // Number of data elements 47 #define NDM41 0.0000000004656612873077392578125 // Number of data elements 48 #define NDM42 0.00000000023283064365386962890625 // Number of data elements 49 #define NDM43 0.000000000116415321826934814453125 // Number of data elements 50 #define NDM44 0.0000000000582076609134674072265625 // Number of data elements 51 #define NDM45 0.00000000002910383045673370361328125 // Number of data elements 52 #define NDM46 0.000000000014551915228366851806640625 // Number of data elements 53 #define NDM47 0.0000000000072759576141834259033203125 // Number of data elements 54 #define NDM48 0.00000000000363797880709171295166015625 // Number of data elements 55 #define NDM49 0.000000000001818989403545856475830078125 // Number of data elements 56 #define NDM50 0.0000000000009094947017729282379150390625 // Number of data elements 57 #define NDM51 0.00000000000045474735088646411895751953125 // Number of data elements 58 #define NDM52 0.000000000000227373675443232059478759765625 // Number of data elements 59 #define NDM53 0.0000000000001136868377216160297393798828125 // Number of data elements 60 #define NDM54 0.00000000000005684341886080801486968994140625 // Number of data elements 61 #define NDM55 0.000000000000028421709430404007434844970703125 // Number of data elements 62 #define NDM56 0.0000000000000142108547152020037174224853515625 // Number of data elements 63 #define NDM57 0.00000000000000710542735760100185871124267578125 // Number of data elements 64 #define NDM58 0.000000000000003552713678800500929355621337890625 // Number of data elements 65 #define NDM59 0.0000000000000017763568394002504646778106689453125 // Number of data elements 66 #define NDM60 0.00000000000000088817841970012523233890533447265625 // Number of data elements 67 #define NDM61 0.000000000000000444089209850062616169452667236328125 // Number of data elements 68 #define NDM62 0.0000000000000002220446049250313080847263336171875 // Number of data elements 69 #define NDM63 0.00000000000000011102230246251565404236316680859375 // Number of data elements 70 #define NDM64 0.00000000000000005551115123125782702118158334029296875 // Number of data elements 71 #define NDM65 0.000000000000000027755575615628913510590791670146484375 // Number of data elements 72 #define NDM66 0.0000000000000000138777878078144567552953958350732421875 // Number of data elements 73 #define NDM67 0.00000000000000000693889390390722837764769791753662109375 // Number of data elements 74 #define NDM68 0.000000000000000003469446951953614188823848958768310546875 // Number of data elements 75 #define NDM69 0.0000000000000000017347234759768070944119244793841552734375 // Number of data elements 76 #define NDM70 0.0000000000000000008673617379884035472055962396920776171875 // Number of data elements 77 #define NDM71 0.00000000000000000043368086899420177360279811984603880859375 // Number of data elements 78 #define NDM72 0.000000000000000000216840434497100886801399059923019404296875 // Number of data elements 79 #define NDM73 0.0000000000000000001084202172485504434006995299615097021484375 // Number of data elements 80 #define NDM74 0.000000000000000000054210108624275221700349764980754851072265625 // Number of data elements 81 #define NDM75 0.000000000000000000027105054312137610850174882490377425361328125 // Number of data elements 82 #define NDM76 0.0000000000000000000135525271560688054250874412451887126806640625 // Number of data elements 83 #define NDM77 0.0000000000000000000067762635780344027125437206225943564033203125 // Number of data elements 84 #define NDM78 0.00000000000000000000338813178901720135627186031129717820166015625 // Number of data elements 85 #define NDM79 0.0000000000000000000016940658945086006781359301556485891008330078125 // Number of data elements 86 #define NDM80 0.00000000000000000000084703294725430033906796507782429455041650390625 // Number of data elements 87 #define NDM81 0.000000000000000000000423516473627150169533982538912147275208251953125 // Number of data elements 88 #define NDM82 0.0000</pre>					

## キャッシュに関する論文

- M.K. Qureshi, D. Thompson, Y.N. Patt.  
**The V-Way Cache: Demand Based Associativity via Global Replacement.**  
In Proc. of Int. Symp. Computer Architecture, ISCA 2005.
- Kaushik Rajan and R.Govindarajan.  
**Emulating optimal replacement with a shepherd cache,**  
In MICRO-40, pp. 445-454, 2007.

どちらか1つを選択して、プロセッサシミュレータ上に実装、性能評価する.

8

# レポート キャッシュの実装

1. SimCore/MIPS functional simulatorにキャッシュを追加して、そのヒット率とプロセッサ性能を評価する。
  1. ベンチマークプログラム 40bench を用いる。
  2. 命令は考慮しない。LW, SW命令で参照するデータキャッシュを対象。
  3. キャッシュの構成は講義スライドを参照

# レポート キャッシュの実装


---

## 1. 評価項目

1. ベンチマークプログラムを実行した際の、ロード命令の実行回数、ストア命令の実行回数を測定してまとめること。メモリアップされたアドレス0へのロード、ストアはカウントしてはならない。
2. 1KBのL1データキャッシュを実装し、そのミス率を測定せよ。また、L1キャッシュを追加することによるプロセッサ性能の向上率を示すこと。
  1. ライトアロケート方式（書き込みでミスした場合も、データをキャッシュに格納）とする。
3. 4KBの標準的な4-way set associative方式のL2データキャッシュを実装し（L1データキャッシュも利用）、そのミス率を測定せよ。また、L2キャッシュを追加することによるプロセッサ性能の向上率を示すこと。
4. 講義で紹介した2つのキャッシュ方式から1つを選択し、その方式をL2キャッシュとして実装すること。ミス率を測定せよ。L2キャッシュを変更することによるプロセッサ性能の向上率を示すこと。標準的なL2より高い性能を実現すること。想定した性能が得られたか議論せよ。

## レポート 提出方法

- **12月13日の講義開始時に印刷したものを提出.**
- **A4サイズで10ページ以内にまとめること.**
  - 実装したキャッシュのコードを掲載すること.
  - 実装した各種の方式のミス率をグラフにして示すこと.
  - 実装したキャッシュの挙動は本当に正しいのか、どうして正しいと主張できるか説明すること.



# 講義計画

---

<http://www.arch.cs.titech.ac.jp/sub5.html>

- 導入：マイクロプロセッサ
- スーパースカラプロセッサの基礎と命令レベル並列性
- **キャッシュ**
- **分岐予測**
- 動的命令スケジューリングと投機処理
- メモリデータフローとデータキャッシュ
- 組込技術，低消費電力技術
- チップマルチプロセッサ
- オンチップネットワーク，メニーコアアーキテクチャ

■ **【成績評価】レポートおよび、期末レポートにより評価する。**

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

12