Quiz 2

• Stats:
  – Total
    • Average: 51.6 points (64.5%)
    • SD: 9.8 points (12.2%)

  – Q1
    • A: 19.7
    • SD: 5.0

  – Q2
    • A: 17.6
    • SD: 4.2

  – Q3
    • A: 12.3
    • SD: 4.9
Agenda

- The Mystery OOO Processor
- Quiz 2 Solutions
- Lab 3
  - branch predictors
DEC Alpha 21264

- 1996/1997
- single-core
  - 4-way
  - out-of-order
  - highly speculative
  - 7-stage
  - up to 80 instructions in flight
  - tournament branch predictor
- 15.2M transistors
  - 6M for logic
  - rest is caching, history tables
- 350 nm
- 600 MHz
- 64KB I$, 64KB D$ (on-chip)
  - 1 to 16MB L2$ (off-chip)
- 314mm$^2$ die (fairly large)
DEC Alpha 21264

- 1996/1997
- single-core
  - 4-way
  - out-of-order
  - highly speculative
  - 7-stage
  - up to 80 instructions in flight
  - tournament branch predictor
- 15.2M transistors
  - 6M for logic
  - rest is caching, history tables
- 350 nm
- 600 MHz
- 64KB I$, 64KB D$ (on-chip)
  - 1 to 16MB L2$ (off-chip)
- 314mm² die (fairly large)
Quiz 2: Question 1.a: Cache Parameters

- 32-bit address
  - 8 index bits
  - 4 block offset bits

- tag size = 32 - index_sz - blk_offset_sz = 20 bits
- number of bytes/blk = 2^{blk_offset_sz} = 16 bytes
- number of sets = 2^{index_sz} = 256 sets

- capacity = (#sets) * (#bytes/line) * (#associativity) = 8KB
Quiz 2 – Question 1.B Critical Paths

Friday, March 11, 2011
Quiz 2 – Question 1.B Critical Paths

tag decoder -> tag read -> comparator -> 2-in AND -> buffer driver -> data output driver

220 + 360 + 450 + 40 + 190 + 280 = 1540 ps

Friday, March 11, 2011
Quiz 2 – Question 1.B Critical Paths
Quiz 2 – Question 1

Data decoder -> data read -> mux -> data output driver

220 + 400 + 300 + 190 = **1110ps**
## Quiz 2 – Question 1.c: Miss Rates

<table>
<thead>
<tr>
<th>C code</th>
<th>Psuedo-assembly of inner-loop</th>
</tr>
</thead>
</table>
| ```c
#define N 4096
int A[N], B[N];
int i;

for(i = 0; i < N; i++)
``` | ```
# rA holds the addr to A[i]
# rB holds the addr to B[i]
LD  r2,0(rB)
LD  r1,0(rA)
ADD r1,r1,r2
ST  r1,0(rA)
``` |

- 8KB 2-way, 4-word lines (16B), 256 lines
- SA,CA caches are identical in miss-rates
- 4 iterations fit in a single cache line
- 3 accesses per iteration
- first two accesses in first iteration miss (i%4==0)
- miss-rate = 2/12, or 16.67%

Friday, March 11, 2011
## Quiz 2 – Question 1.d: Hit Rates

<table>
<thead>
<tr>
<th>C code</th>
<th>Pseudo-assembly of inner-loop</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>#define N 4096</code></td>
<td><code># rA holds the addr to A[i]</code></td>
</tr>
<tr>
<td><code>int A[N], B[N];</code></td>
<td><code># rB holds the addr to B[i]</code></td>
</tr>
<tr>
<td><code>int i;</code></td>
<td><code>LD r2,0(rB)</code></td>
</tr>
<tr>
<td><code>for(i = 0; i &lt; N; i++)</code></td>
<td><code>LD r1,0(rA)</code></td>
</tr>
<tr>
<td><code>A[i] = A[i] + B[i];</code></td>
<td><code>ADD r1,r1,r2</code></td>
</tr>
<tr>
<td></td>
<td><code>ST r1,0(rA)</code></td>
</tr>
</tbody>
</table>

- 8KB 2-way, 4-word lines (16B), 256 lines
- SA: hit_rate = 1 - miss_rate = \( \frac{10}{12} \), or 83.33%
- CA:
  - 4 iterations fit in a single cache line
  - 3 accesses per iteration
  - first two accesses in first iteration miss (i%4==0)
  - ALL stores fast hit
  - fast-hit-rate = \( \frac{4}{12} \), or 33.33%
  - slow-hit-rate = \( \frac{6}{12} \), or 50.00%
AMATSa = hit_time + miss_rate * miss_penalty

-or-

AMATSa = hit_rate*hit_time + miss_rate*miss_time

(miss_time = miss_penalty + time_to_detect_miss)

hit_time = 1.54ns (one cycle)
miss_rate = 16.67%
miss_penalty = 100ns
time_to_detect = 1.54ns (one cycle)
hit_rate = 83.33%

AMATsa = (0.83)(1.54ns) + (.1667)(100ns+1.54ns)
= 18.21ns
AMAT_ca =

\[ \text{fast_hit_rate} \times \text{fast_hit_time} + \text{slow_hit_rate} \times \text{slow_hit_time} + \text{miss_rate} \times \text{miss_time} \]

(miss_time = miss\_penalty + time\_to\_detect\_miss)

time\_to\_detect = two cycles!
(one cycle to check Way0, second cycle to check Way1)

AMAT_ca = \((0.333)(1.11\text{ns}) + (0.500)(4 \times 1.11\text{ns}) + (0.1667)(100\text{ns+2} \times 1.11\text{ns})\)

= \textbf{19.63ns}
Quiz 2 – Question 1.e: AMAT

AMAT_ca = 
    fast_hit_rate*fast_hit_time 
    + slow_hit_rate*slow_hit_time 
    + miss_rate*miss_time

final caveats:
Don’t double count Way0 check
Make sure to count Way1 check
Quiz 2 – Question 1.f: Crossover (Performance)

• when is AMAT_sa == AMAT_ca?
  – miss_rates are identical
  – miss_times differ by ~1ns (out of >100ns)
  – so we can ignore considering misses

\[ AMAT_{sa} = AMAT_{ca} \]
\[ \text{hit}_\text{rate} \times \text{hit}_\text{time} = \text{fast}_\text{hit}_\text{rate} \times \text{fast}_\text{hit}_\text{time} + \text{slow}_\text{hit}_\text{rate} \times \text{slow}_\text{hit}_\text{time} \]

let fast_hit_rate = x, slow_hit_rate = y

\[ (x+y)1.54\text{ns} = x\times1.11\text{ns} + y\times4.44\text{ns} \]

\[ x/y = 6.74 \]
Quiz 2 – Question 1.g: Crossover (Energy)

• Read/write to a cache line takes 1nJ

  – fast-hit (CA)?
    • 1 nJ
    • only read Way0
  – hit (SA)?
    • 2 nJ
    • must read both Way0 and Way1 to look for data
  – slow-hit (CA)?
    • 6 nJ
    • read Way0, verify miss (1nJ)
    • read out both ways to swap registers, check Way1’s tag (2nJ)
    • finish swap by writing to both ways (2nJ)
    • read out Way0’s data (1nJ)
Quiz 2 – Question 1.h: Crossover (Energy)

• Energy_sa == Energy_ca?
• miss_rate is small, miss_energy_penalty small
• ignore miss behavior

\[
\text{Energy}_{sa} = \text{Energy}_{ca} \\
\text{hit_rate} \times \text{hit_energy} = \text{fast_hit_rate} \times \text{fast_hit_energy} + \text{slow_hit_rate} \times \text{slow_hit_energy}
\]

let fast_hit_rate = x, slow_hit_rate = y

\[
(x+y)2nJ = x \times 1nJ + y \times 6nJ
\]

\[
x/y = 4
\]
Quiz 2 – Question 1.i: Aliasing

- VA1, VA2 alias PA
- PA has physical tag Pt
- VA1, VA2 map to same set
  - \((\text{index}_{sz} + \text{blk}_{off}_{sz} \leq \text{page}_{off}_{sz})\)

- Load VA1 into cache

- Then Load VA2. What happens?
  - VA2 maps to the same physical tag as VA1 (Pt)
  - VA2 maps to same set and sees tag Pt
  - VA2 sees a hit!
  - access to VA2 returns the correct data pointed to by PA
Changing from SA to CA cache

- compulsory misses
  - no change
  - everything is identical (except for fast-hit/slow-hit timing)

- conflict misses
  - no change
  - same reason

- capacity misses
  - no change
  - same reason
• add sub-blocking (line size constant)
  – compulsory misses
    • increases
    • less data brought, effectively smaller lines
  – conflict misses
    • no change
    • associativity constant
  – capacity misses
    • no change
    • capacity constant
• double line size, but add sub-blocking
• (capacity constant)
  – compulsory misses
    • no change
    • block size brought in is the same
  – conflict misses
    • increases
    • less sets, so more addresses will conflict
  – capacity misses
    • no change
    • capacity constant


• adding prefetching
  – compulsory misses
    • decreases
    • good prefetcher brings in data before we need it
  – conflict misses
    • increase
    • prefetched data could pollute the cache
  – capacity misses
    • increase
    • prefetched data could pollute the cache
Quiz 2 – Question 3

• definitions:
  – TLB contribution to the CPI
    • imagine measuring the CPI for a given program
    • how will the TLB’s contribution affect the CPI?
    • i.e., on a TLB miss how many cycles get burned handling the miss
  – TLB reach
    • how much memory is accessible from the TLB
    • # of TLB entries * Page Size
  – TLB capacity misses
    • a memory reference that misses in the TLB that would not occur if the TLB had more entries
Quiz 2 – Question 3.i

• increase page size
  – TLB reach
    • increases
    • pages are bigger, so for a fixed amount of entries the TLB can directly access more memory
  – TLB contribution to the CPI
    • decreases
    • more TLB reach, means more memory can be accessed with fewer misses (both capacity and compulsory)
  – TLB capacity misses
    • decreases
    • viewed over an entire program, fewer memory accesses will miss because more memory can fit inside of a given page (bigger reach)
• increase number of TLB entries
  – TLB reach
    • increases
    • more entries are stored, provides access to more memory
  – TLB capacity misses
    • decreases
    • stores more entries
  – TLB contribution to the CPI
    • decreases
    • more TLB reach, means more memory can be accessed with fewer capacity and fewer conflict misses
• increase number of levels in VM hierarchy

• ambiguous if this meant:
  – #levels in page table or
  – #levels in TLB cache hierarchy
Quiz 2 – Question 3.iii

• increase number of levels in VM hierarchy (# of page table levels)
  – TLB reach
    • unchanged
    • (# of TLB entries*page_size is unaffected)
  – TLB capacity misses
    • unchanged
  – TLB contribution to the CPI
    • increases
    • takes longer to walk the page tables
    • (more memory references required)
• increase number of levels in VM hierarchy (# of levels in TLB hierarchy)
  – TLB reach
    • increases
    • assuming outer levels of TLB provide more capacity
  – TLB capacity misses
    • decreases
    • assuming you said that outer levels of TLB hierarchy catch the inner level misses
  – TLB contribution to the CPI
    • decreases
    • because of more caching of page table entries
• increase virtual address from 32b to 64b
  – must increase number of levels in the VM hierarchy!
    • assuming 4KB page sizes, it’d otherwise require
    • $2^{64-12} \times 4B = 2^{54}B = 16PB$ devoted just to the page table!!!
  – answers are the same as the last question!
  – TLB reach
    • unchanged
    • no effect on page size or # of entries
  – TLB capacity misses
    • unchanged
    • because TLB reach is unchanged
  – TLB contribution to the CPI
    • increases since
    • because of more caching of page table entries
Branch Predictors