Assignment 3
CMSC611-101
Advanced Computer Architecture, Fall 2009

Sandipan Dey

Question 1: *(20 Points)*
Consider a branch-target buffer that has penalties of 0, 2, and 2 clock cycles for correct conditional branch prediction, incorrect prediction, and a buffer miss, respectively. Consider a branch-target buffer design that distinguishes conditional and unconditional branches, storing the target address for a conditional branch and the target instruction for an unconditional branch.
A) What is the penalty in clock cycles when an unconditional branch is found in the buffer?
B) Determine the improvement from branch folding for unconditional branches. Assume a 90% hit rate, an unconditional branch frequency of 5%, and a 2-cycle penalty for a buffer miss. How much improvement is gained by this enhancement? How high must the hit rate be for this enhancement to provide a performance gain?

Answer:

A) In case of branch folding the target instruction is stored in BTB instead of the target address. Hence, the target instruction need not be fetched when the unconditional branch is found in the buffer. This is equivalent to (-1) stalls, since the next IF cycle is exempted, hence penalty in clock cycles = -1 (one clock cycle is saved, improvement by 1 clock cycle) when the branch is found in the buffer.

B) For unconditional branches, without branch folding (storing target address for unconditional branch instead),

   $\text{Penalty}_{\text{withoutfolding}}$ = 5%.(90%.0 + 10%.2) = 0.01 clock cycles.
   $\text{Penalty}_{\text{withfolding}}$ = 5%.(90%.(-1) + 10%.2) = -0.035 clock cycles. (-ive =>Improvement!)
   Improvement with folding over without folding =0.01 – (-0.035) = 0.045 clock cycles.

   If we assume that average CPI before this improvement is 1.0,
   after branch folding CPI becomes $\text{CPI}_{\text{folding}}$ =1 –.035 = 0.965.

   Without branch folding (storing target address for unconditional branch instead),
   the CPI becomes $\text{CPI}_{\text{withoutfolding}}$ =1 + 5%.(90%.0 + 10%.2) = 1.01.

   Hence, speedup = 1.01 / 0.965 = 1.04663, performance gain = 4.66%.

   Let's assume the required hit rate to be *x*%. Then
   $\text{Penalty}_{\text{withfolding}}$ = 5%.(*x*%.(-1) + (1-*x*%).2) clock cycles.
   But, in order to be an improvement, $\text{Penalty}_{\text{withfolding}}$ = -5%. (*x*% - 2 (1-*x*%)) < 0
   => 3*x*/100 > 2 => *x* > 66.67%.

Hence, minimum hit rate must be greater than 66.67%.

Question 2: *(20 Points)*
One difference between a write through cache and write-back cache can be in the time it takes to write. During the first cycle, we detect whether a hit will occur, and during the second (assuming a hit) we actually write the data. For this question, assume that the write buffer for a write through will never stall the CPU (no penalty). Assume a read hit takes 1 clock cycle, the cache miss penalty is 50 clock cycles, and a block write from the cache to main memory takes 50 clock cycles. Finally, assume the instruction cache miss rate is 0.5 % and the data cache miss rate is 1%.

Use the instruction frequencies shown in the following table:

Instruction	load	store	add	sub	mul	compare	load imm	cond branch	jump	call	return	shift	and	or	other
Frequency	26%	9%	14.6%	0.5%	0.1%	12.4%	7.9%	11.5%	1.3%	1.1%	1.5%	6.2%	1.6%	4.2%	1.1%

A) Estimate the performance of a write–through cache with a two cycle write.
B) Repeat part (A) assuming the write–through cache pipelines the writes, as discussed in class, so that a write hit takes just one clock cycle.

Answer:

We assume instruction cache is read-only. Hence we assume that the 2-cycle writes occur only at the data cache.

A) load instruction = 26%
store instruction = 9%
load + store instruction = 35%

% instruction cache references = 100 % / (100 + 35)% = 74.07%
%data cache references = 35 % / (100 + 35)% = 25.93%
%data cache references for read (in case of load) = 26 % / (100 + 35)% = 19.26%
%data cache references for write (in case of store) = 9 % / (100 + 35)% = 6.67%

Since, Average Memory Access Time = Hit Time + Miss Rate x Miss Penalty.

$Time_{instruction}$ = 74.04% x (1 + 0.5% x 50) = 74.04% x1.25 =0.93

$Time_{Data\,\mathrm{Re}\,ad}$ = 19.26% x (1 + 1% x 50) = 19.26% x1.5 =0.29

Both the above memory accesses are strictly read and not write, hence will not go to the second write cycle.

Now, consider the write,
$Time_{DataWrite}$ =6.67%x(99%x(1 + 1) + 1%x(1+50+1))= 6.67%x(1.98 +0.52)=0.17

Hence, Average Memory Access Time with 2 cycle write
= 0.93 + 0.29 + 0.17 = 1.39

B) When write is pipelined, write hit takes one clock cycle, hence, we have,
$Time_{DataWrite}$ =6.67%x(99%x(1) + 1%x(1+50+1))= 6.67%x(0.99 +0.52)=0.10
Hence, Average Memory Access Time with 2 cycle write pipelined
= 0.93 + 0.29 + 0.10 = 1.28

Question 3: *(60 Points)*
You are building a system around a processor with in-order execution that runs at 1.1 GHz and has a CPI of 0.7 excluding memory accesses. The only instructions that read or write data from memory are loads (20% of all instructions) and stores (5% of all instructions). The memory system for this computer is composed of a direct-mapped LI cache that imposes no penalty on hits. Both the I-cache and D-cache are direct mapped and hold 32 KB each. The I-cache has a 2% miss rate and 32-byte blocks, and the D-cache is write-through with a 5% miss rate and 16-byte blocks. There is a write buffer on the D-cache that eliminates stalls for 95% of all write accesses. The 512 KB write-back, fully associative L2 cache has 64-byte blocks and an access time of 15 ns. It is connected to the LI cache by a 128-bit data bus that runs at 266 MHz and can transfer one 128-bit word per bus cycle. Of all memory references sent to the L2 cache in this system, 80% are satisfied without going to main memory. Also, 50% of all blocks replaced are dirty. The 128-bit-wide main memory has an access latency of 60 ns, after which any number of bus words may be transferred at the rate of one per cycle on the 128-bit-wide 133 MHz main memory bus.

A) What is the average memory access time for instruction accesses?

B) What is the average memory access time for data reads?

C) What is the average memory access time for data writes?

D) What is the overall CPI, including memory accesses?

E) You are considering replacing the 1.1 GHz CPU with one that runs at 2.1 GHz, but is otherwise identical. How much faster does the system run with a faster processor? Assume the LI cache still has no hit penalty, and the speed of the L2 cache, main memory, and buses remains the same in absolute terms (e.g., the L2 cache still has a 15 ns access time and a 266 MHz6 connecting it to the CPU and LI cache).

F) If you want to make your system runs faster, which part of the memory system would you improve? Graph the change in overall system performance holding all parameters fixed except the one that you are improving. Parameters you might consider improving include L2 cache speed, bus speeds, main memory speed, and LI and L2 hit rates. Based on these graphs, how could you best improve overall system performance with minimal cost?

Answer:

A)

$$Time\ to\ send\ an\ instruction\ from\ Main\ Memory\ to\ L_2\ Cache = (64bytes/16bytes) \times \frac{10^{-6}}{133}\sec = 30ns$$

$$Time\ to\ send\ instruction\ from\ L_2\ Cache\ to\ L_1\ instruction\ Cache = (32bytes/16bytes) \times \frac{10^{-6}}{266}\sec = 7.52ns$$

$$L_2\ Cache\ Miss\ Penalty$$
$$= Percentage\ Dirty \times Time\ to\ write\ old\ instruction\ (to\ be\ replaced)\ from\ L2\ Cache\ to\ Main\ Memory$$
$$+ Time\ to\ read\ instruction\ from\ L2\ Cache\ to\ Main\ Memory$$
$$= Percentage\ Dirty \times (Memory\ Access\ Time + Time\ to\ send\ instruction\ from\ L_2\ Cache\ to\ Main\ Memory)$$
$$+ Main\ Memory\ Access\ Time + Time\ to\ send\ instruction\ from\ Main\ Memory\ to\ L_2\ Cache$$
$$= (Percentage\ Dirty + 1) \times (Memory\ Access\ Time + instruction\ send\ time\ between\ L_2\ Cache\ \&\ Memory)$$
$$= (50\% + 1) \times (60 + 30) = 1.5 \times 90 = 135ns$$

$$L_1\ Instruction\ Cache\ Miss\ Penalty$$
$$= L_2\ Cache\ Hit\ Time + Time\ to\ send\ instruction\ from\ L_2\ Cache\ to\ L_1\ Instruction\ Cache$$
$$+ L_2\ Cache\ Miss\ Rate \times L_2\ Cache\ Miss\ Penalty$$
$$= 15 + 7.52 + 20\% \times 135 = 15 + 7.52 + 27 = 49.52ns$$

$$\text{Average Memory Access Time for instruction access} = Time_{Instruction}$$
$$= L_1\ Instruction\ Cache\ Hit\ Time + L_1\ Instruction\ Cache\ Miss\ Rate \times L_1\ Instruction\ Cache\ Miss\ Penalty$$
$$= 0 + 2\% \times 49.52 = 99.14\% = 0.99ns$$

B)

$$Time\ to\ send\ data\ from\ Main\ Memory\ to\ L_2\ Cache = (64bytes/16bytes)\times\frac{10^{-6}}{133}\sec = 4\times 7.5ns = 30ns$$

$$Time\ to\ send\ data\ from\ L_2\ Cache\ to\ L_1\ data\ Cache = (16bytes/16bytes)\times\frac{10^{-6}}{266}\sec = 1\times 3.76ns = 3.76ns$$

$$L_2\ Cache\ Miss\ Penalty$$
$$= Percentage\,Dirty\times Time\ to\ write\ old\ data\ (to\ be\ replaced)\ \ from\ L2\ Cache\ back\ to\ Main\ Memory$$
$$+ Time\ to\ read\ data\ from\ L2\ Cache\ to\ Main\ Memory$$
$$= Percentage\,Dirty\times(Memory\ Access\ Time + Time\ to\ send\ data\ from\ L_2\ Cache\ to\ Main\ Memory)$$
$$+ Main\ Memory\ Access\ Time + Time\ to\ send\ data\ from\ Main\ Memory\ to\ L_2\ Cache$$
$$= (Percentage\,Dirty + 1)\times(Memory\ Access\ Time + data\ send\ time\ between\ L_2\ Cache\,\&\,Memory)$$
$$= (50\% + 1)\times(60+30) = 1.5\times 90 = 135ns$$

$$L_1\ Data\ Cache\ Miss\ Penalty$$
$$= L_2\ Cache\ Hit\ Time + Time\ to\ send\ data\ from\ L_2\ Cache\ to\ L_1\ Data\ Cache$$
$$+ L_2\ Cache\ Miss\ Rate\times L_2\ Cache\ Miss\ Penalty$$
$$= 15 + 3.76 + 20\%\times 135 = 15 + 3.76 + 27 = 45.76ns$$

$$\text{Average Memory Access Time for Data Read} = Time_{Data\,\mathrm{Re}\,ad}$$
$$= L_1\ Data\ Cache\ Hit\ Time + L_1\ Data\ Cache\ Miss\ Rate\times L_1\ Data\ Cache\ Miss\ Penalty$$
$$= 0 + 5\%\times 45.76 = 228.80\% = 2.29ns$$

C)

$$Time\ to\ send\ data\ from\ Main\ Memory\ to\ L_2\ Cache = (64bytes/16bytes)\times\frac{10^{-6}}{133}\sec = 4\times 7.5ns = 30ns$$

$$Time\ to\ send\ data\ from\ L_2\ Cache\ to\ L_1\ data\ Cache = (16bytes/16bytes)\times\frac{10^{-6}}{266}\sec = 1\times 3.76ns = 3.76ns$$

$L_2\ Cache\ Miss\ Penalty$

$= Percentage\ Dirty \times Time\ to\ write\ old\ data\ (to\ be\ replaced)\ from\ L2\ Cache\ back\ to\ Main\ Memory$

$+ Time\ to\ read\ data\ from\ L2\ Cache\ to\ Main\ Memory$

$= Percentage\ Dirty \times (Memory\ Access\ Time + Time\ to\ send\ data\ from\ L_2\ Cache\ to\ Main\ Memory)$

$+ Main\ Memory\ Access\ Time + Time\ to\ send\ data\ from\ Main\ Memory\ to\ L_2\ Cache$

$= (Percentage\ Dirty + 1)\times(Memory\ Access\ Time + data\ send\ time\ between\ L_2\ Cache \,\&\, Memory)$

$= (50\% + 1)\times(60+30) = 1.5\times 90 = 135ns$

$L_1\ Data\ Cache\ Miss\ Penalty$

$= L_2\ Cache\ Hit\ Time + Time\ to\ send\ data\ from\ L_2\ Cache\ to\ L_1\ Data\ Cache$

$+ L_2\ Cache\ Miss\ Rate \times L_2\ Cache\ Miss\ Penalty$

$= 15 + 3.76 + 20\%\times 135 = 15 + 3.76 + 27 = 45.76ns$

$\text{Average Memory Access Time for Data Write} = Time_{DataWrite}$

$= 95\% \times No\ Stall\ Time\ for\ Write\ Buffer + 5\% \times L_1\ Data\ Cache\ Miss\ Penalty$

$= 0 + 5\%\times 45.76 = 228.80\% = 2.29ns$

D)

$Overall\ CPI\ including\ Memory\ Access$

$= CPI_{CPUExecution} + (Time_{instruction} + (\%load)\times Time_{Data\,Read} + (\%store)\times Time_{DataWrite})\times Clock\ Rate$

$= 0.7 + (0.99 + 20\%\times 2.29 + 5\%\times 2.29)\times 1.1GHz = 2.42$

E)

$Overall\ CPI\ including\ Memory\ Access\ with\ new\ CPU$

$= CPI_{CPU} + (Time_{instruction} + (\%load)\times Time_{Data\,Read} + (\%store)\times Time_{DataWrite})\times Clock\ Rate$

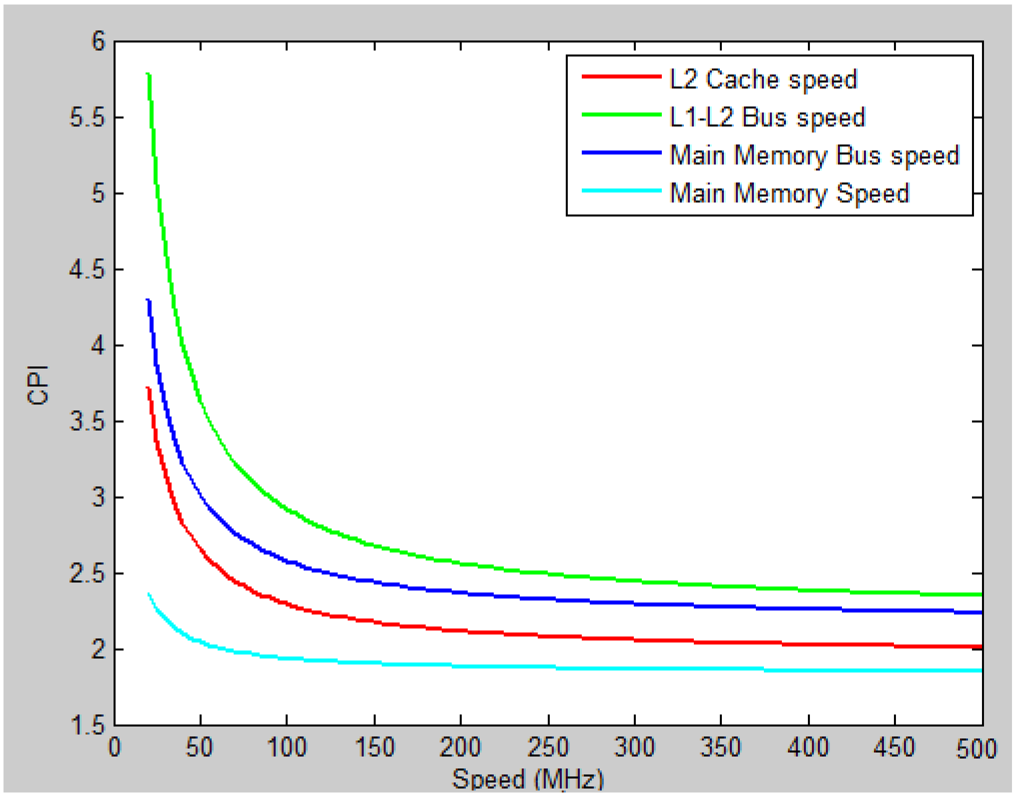
$= 0.7 + (0.99 + 20\%\times 2.29 + 5\%\times 2.29)\times 2.1GHz = 3.98ns$

$$\therefore Speedup = \frac{Performance_{new}}{Performance_{old}} = \frac{(CPU\ time)_{old}}{(CPU\ time)_{new}} = \frac{IC_{old}\times(Overall_{\times}CPI)_{old}\times(Clock\ Cycle\ Time)_{old}}{IC_{new}\times(Overall_{\times}CPI)_{new}\times(Clock\ Cycle\ Time)_{new}}$$

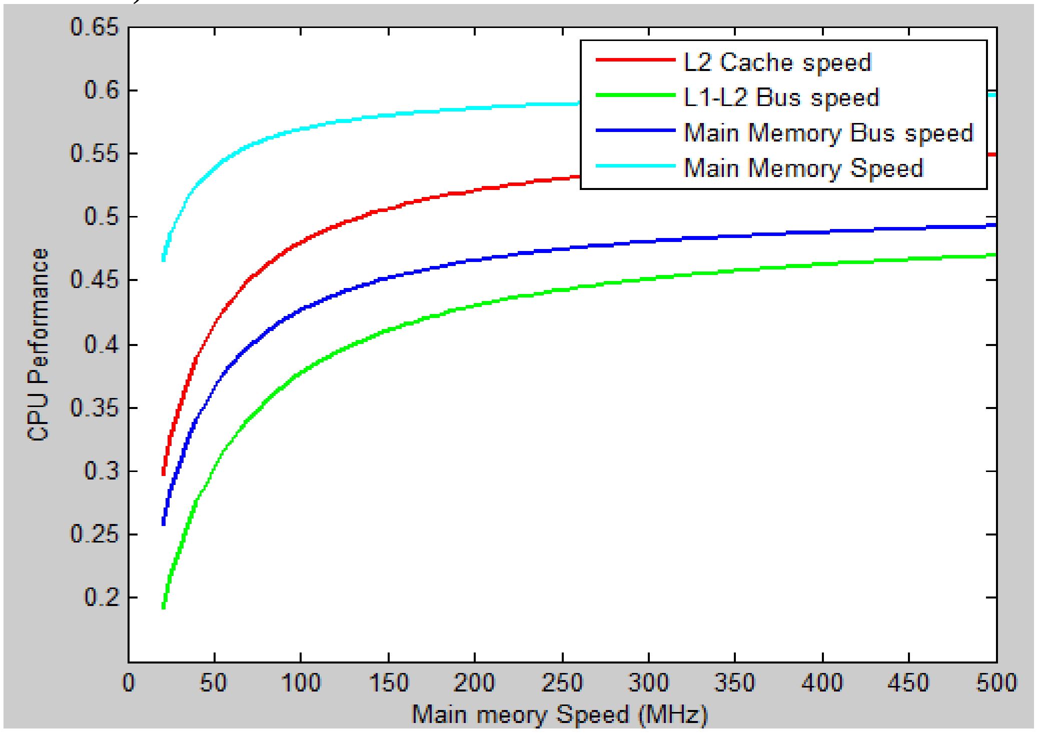
$$= \frac{(Overall_{\times}CPI)_{old}\times(Clock\ Rate)_{new}}{(Overall_{\times}CPI)_{new}\times(Clock\ Rate)_{old}},\quad assu\min g\ IC_{old} = IC_{new}$$

$$= \frac{2.42\times 2.1}{3.98\times 1.1} = 1.16$$

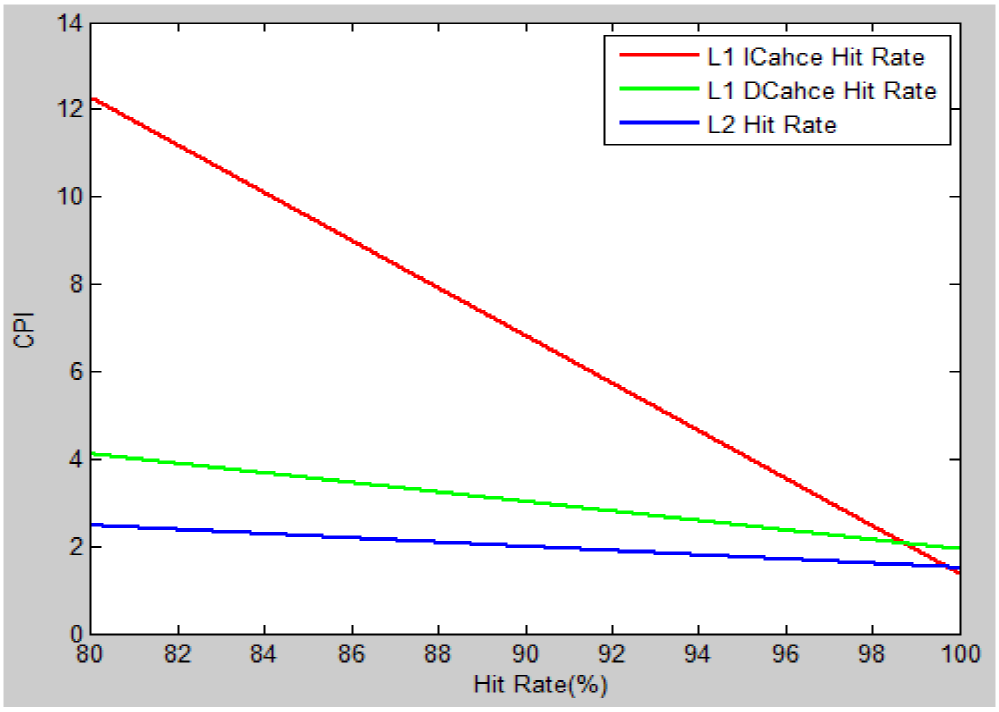
F) To run the system faster (keeping the CPU clock rate unchanged), one needs to decrease the overall CPI.



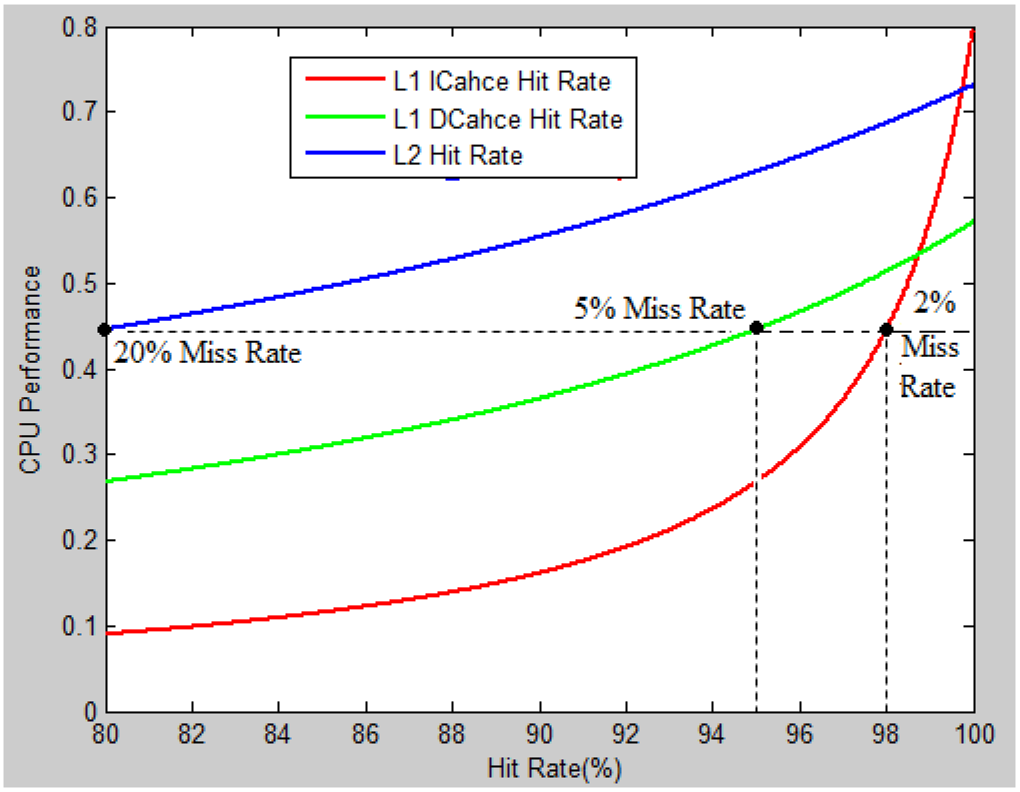
As can be seen from above graph, out of the 4 different speeds (keeping other parameters constant), decreasing Main Memory Speed (having largest negative slope around CPI value 2.42) reduces the CPI most.



Using CPU performance = 1/CPU Time = 1/(IC x CPI x Clock Cycle) = Clock Rate / CPI, assuming a fixed IC value, e.g., IC = 1, we get the above slowly-increasing CPU performance with increase in speed of different parameters considered one at a time keeping other parameters fixed.

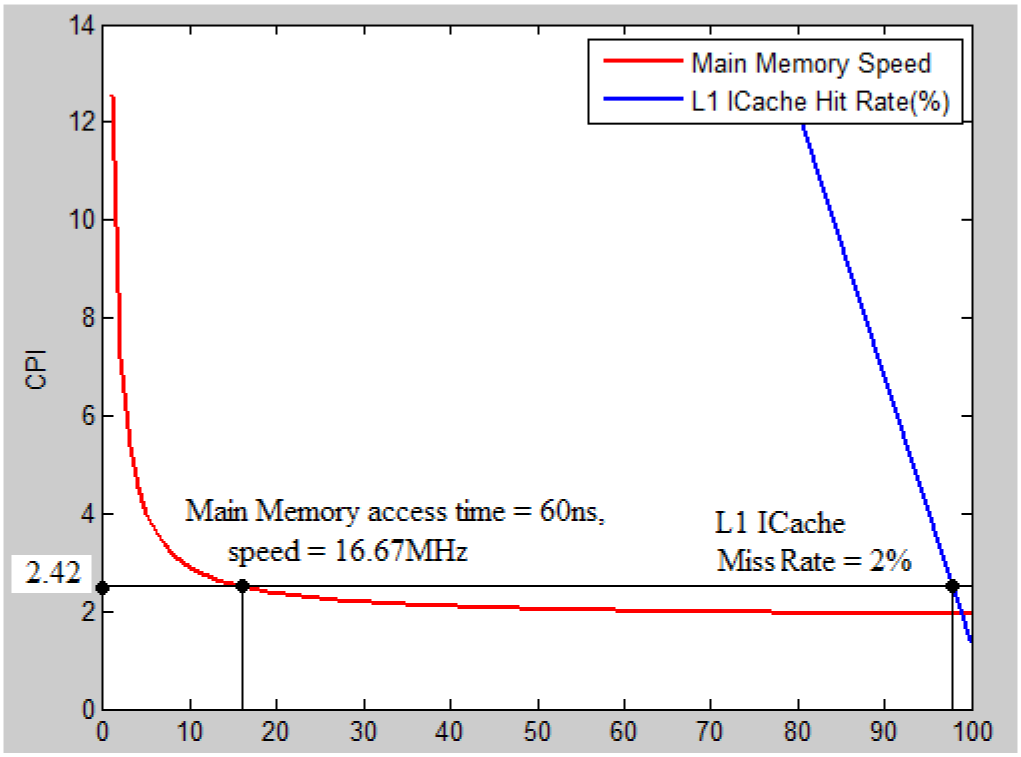


Also, as can be seen from above graph, out of the 3 Hit Rates, increasing the L1 Instruction Cache Hit Rate (having largest negative slope) reduces the CPI most (we keep other parameters constant).



Using CPU performance = 1/CPU Time = 1/(IC x CPI x Clock Cycle) = Clock Rate / CPI, assuming a fixed IC value, e.g., IC = 1, we get the above slowly-increasing CPU performance with increase in speed of different parameters considered one at a time keeping other parameters fixed.

As can be seen from the above figure, the CPU performance is growing rapidly with change of Cache Hit Rates, with gradient of increase being the steepest when increasing L1 ICache Hit Rate.



Again as we can see from the above graph, increasing instruction cache hit rate slightly reduces the CPI more than increasing the Main Memory Speed (or equivalently, decreasing Main Memory access time).

For instance, if we decrease Main Memory access time to 50ns keeping other parameters constant,

With initial

$$CPU\ Performance = \frac{1}{CPU\ Time} = \frac{1}{IC \times CPI \times Clock\ Cycles} = \frac{Clock\ Rate}{IC \times CPI} = \frac{1.1}{IC \times 2.42} = \frac{0.45}{IC} GHz$$

$$L_2\ Cache\ Miss\ Penalty = (50\% + 1) \times (50 + 30) = 1.5 \times 80 = 120ns$$

$$L_1\ Data\ Cache\ Miss\ Penalty = 15 + 3.76 + 20\% \times 120 = 15 + 3.76 + 24 = 42.76ns$$

$$L_1\ Instruction\ Cache\ Miss\ Penalty = 15 + 7.52 + 20\% \times 120 = 15 + 7.52 + 24 = 46.52ns$$

$$Time_{Instruction} = 0 + 2\% \times 46.52 = 93.14\% = 0.93ns$$

$$Time_{Data\,Read} = 0 + 5\% \times 42.76 = 213.80\% = 2.13ns$$

$$Time_{DataWrite} = 0 + 5\% \times 42.76 = 213.80\% = 2.13ns$$

$$Overall\ CPI\ including\ Memory\ Access = 0.7 + (0.93 + 20\% \times 2.13 + 5\% \times 2.13) \times 1.1GHz = 2.31$$

$$CPU\ Performance = \frac{1}{CPU\ Time} = \frac{1}{IC \times CPI \times Clock\ Cycles} = \frac{Clock\ Rate}{IC \times CPI} = \frac{1.1}{IC \times 2.31} = \frac{0.48}{IC} GHz$$

Let’s decrease L1 Instruction Cache miss rate on the contrary to 1%, while keeping other parameters constant again.

$$L_2\ Cache\ Miss\ Penalty = 135ns$$
$$L_1\ Data\ Cache\ Miss\ Penalty = 45.76ns$$
$$L_1\ Instruction\ Cache\ Miss\ Penalty = 49.52ns$$
$$Time_{Instruction} = 0 + 1\% \times 49.52 = 49.52\% = 0.5ns$$
$$Time_{Data\,Read} = 2.29ns$$
$$Time_{DataWrite} = 2.29ns$$
$$Overall\ CPI\ including\ Memory\ Access = 0.7 + (0.5 + 20\% \times 2.29 + 5\% \times 2.29) \times 1.1GHz = 1.88$$
$$CPU\ Performance = \frac{1}{CPU\ Time} = \frac{1}{IC \times CPI \times Clock\ Cycles} = \frac{Clock\ Rate}{IC \times CPI} = \frac{1.1}{IC \times 1.88} = \frac{0.59}{IC} GHz$$

Let’s decrease L1 Data Cache miss rate on the contrary to 4%, while keeping other parameters constant again.

$$L_2\ Cache\ Miss\ Penalty = 135ns$$
$$L_1\ Data\ Cache\ Miss\ Penalty = 45.76ns$$
$$L_1\ Instruction\ Cache\ Miss\ Penalty = 49.52ns$$
$$Time_{Instruction} = 0 + 1\% \times 49.52 = 49.52\% = 0.5n$$
$$Time_{Data\,Read} = 0 + 4\% \times 45.76 = 1.83ns$$
$$Time_{DataWrite} = 0 + 4\% \times 45.76 = 1.83ns$$
$$Overall\ CPI\ including\ Memory\ Access = 0.7 + (0.93 + 20\% \times 1.83 + 5\% \times 1.83) \times 1.1GHz = 2.26$$
$$CPU\ Performance = \frac{1}{CPU\ Time} = \frac{1}{IC \times CPI \times Clock\ Cycles} = \frac{Clock\ Rate}{IC \times CPI} = \frac{1.1}{IC \times 1.75} = \frac{0.40}{IC} GHz$$

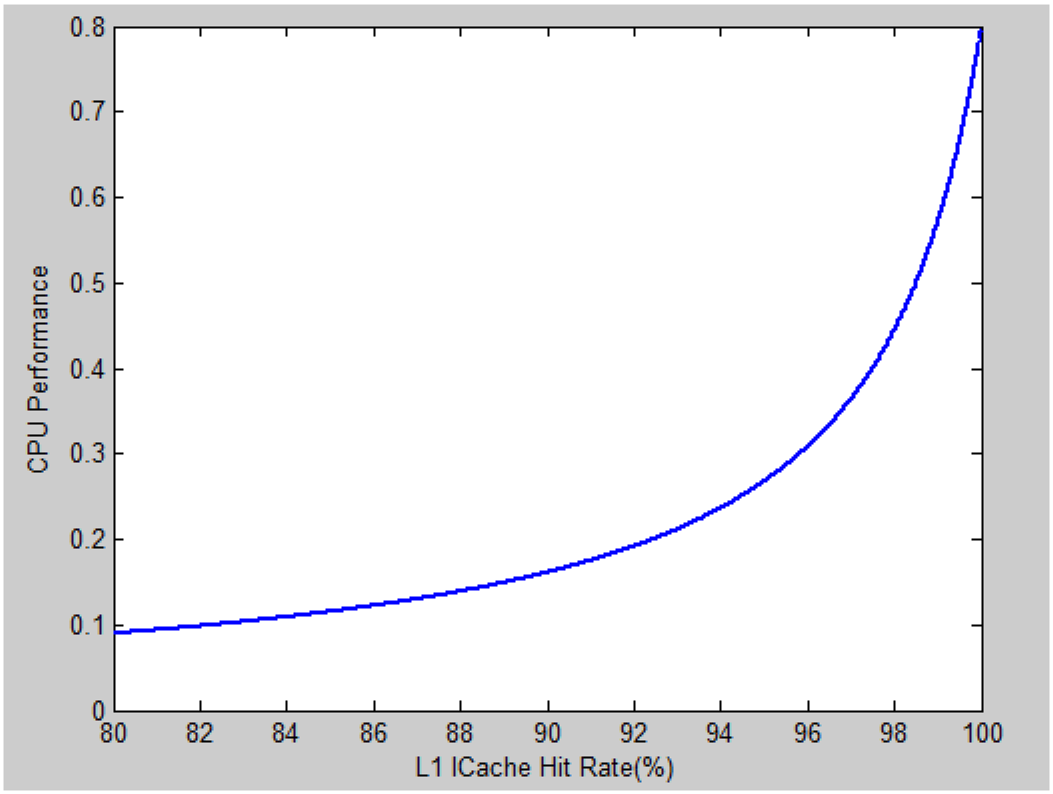
We can see that the decrease in CPI (and increase in performance) is highest when we increase L1 instruction cache hit rate. Hence, increasing the hit rate of L1 Instruction Cache makes the system fastest.

Also, from the equation,

Average Memory Access Time = Hit Time + Miss Rate x Miss Penalty.
For L1 cache, Hit Time = 0 => Average Memory Access Time = Miss Rate x Miss Penalty

⇨ Average Memory Access Time ∞ Miss Rate, with slope as Miss Penalty. If we decrease miss rate (i.e., increase L1 hit rate), the memory access time will decrease that will reduce the overall CPI.

The following graph shows how the performance of CPU varies with L1 cache hit rate.



The above figure shows the exponential boost in CPU performance due to increase in CPU performance = 1/CPU Time = 1/(IC x CPI x Clock Cycle) = Clock Rate / CPI, assuming a fixed IC value, e.g., IC = 1.