CS 6810: Assignment 8

Anirudh Narasimhamurthy(u0941400) and Venakata Vineel Yalamarthi(u0881808)

April 15, 2015

1 Question 1

After running the simulations on the four benchmark programs, the observed IPC's were :

Observed IPC

Benchmark Program	IPC
mgrid	0.698
crafty	1.358
mesa	2.070
parser	0.620

Table 1: IPC's of four benchmark programs on initial a.cfg file

The observed Power Consumption values were :

Observed Power Consumption

Benchmark Program	Power Consumed(in Watts)
mgrid	62.375
crafty	54.915
mesa	68.301
parser	47.655

Table 2: Power consumption of four benchmark programs on initial a.cfg file

The overall IPC given by the arithmetic mean of the IPC's is : **1.1865** and the overall Power Consumed is: **58.3115 W**

After increasing the size of LSQ to 80, the observed IPC's were :

Observed IPC after increasing LSQ size to 80

Benchmark Program	IPC
mgrid	0.698
crafty	1.359
mesa	2.073
parser	0.620

Table 3: IPC's of four benchmark programs after increasing the LSQ size to 80

After increasing the size of LSQ to 160, the observed IPC's were :

Observed IPC after increasing LSQ size to 160

Benchmark Program	IPC
mgrid	0.698
crafty	1.359
mesa	2.073
parser	0.620

Table 4: IPC's of four benchmark programs after increasing the LSQ size to 160

Reason why performance does not improve by much:

1. Performance can be dependent on different parameters other than LSQ. Issue queue, instruction fetch queue and ROB size also plays a part in performance. So, LSQ may not only be deciding factor for calculation performance.

2.Load and Store instructions could probably be only a small part of the instruction set in the benchmark programs. So from Amdahl law, the improvement of this small part, even if it is big willstill barely have an impact on the performance improvement of whole system.

3. Even LSQ is the bottleneck of performance, only increasing LSQ size can not guarantee the performance improvment. The out-of-order buffer may have small band width, or the load instruction might stil wait a long time for the store instr to solve the addresses.

2 Question 2

- The IPC's of the four benchmark programs when we increase the size of the structures by 2x is given below:

Benchmark Program	IPC	
mgrid	1.168	70.753
crafty	1.576	61.933
mesa	2.307	76.430
parser	0.62	49.877

Table 5: IPC's of four benchmark programs after increasing the size of structures by 2x

Benchmark Program	IPC	
mgrid	1.307	76.197
crafty	1.605	67.134
mesa	2.321	82.468
parser	0.616	52.652

Table 6: IPC's of four benchmark programs after increasing the size of structures by 4x

- The average IPC when the structures are increased by 2x is: **1.4175** the IPC when the structures are increased by 4x is: **1.46225**

  The average IPC obtained for the file a.cfg is 1.1865.

  Now performance is given by clock speed*IPC.

  Let the initial clock speed be denoted by C_{old}, clock speed for 2x and 4x be denoted by C2 and C4.

  For 2x processor $C_{old}/C2 = IPC_2/IPC_{Old}$
  $C_{old}/C2 = 1.4175/1.18654$
  $C2 = 0.84 * C_old$

  So new clock speed is 84% of old clock speed. Hence reduction that can be tolerated is 100-84 i.e. 16%.

  For 4x processor $C_{old}/C4 = IPC_4/IPC_{Old}$
  $C_{old}/C4 = 1.4622/1.18654$
  $C4 = 0.8113 * C_old$

  So new clock speed is 81.4% of old clock speed. Hence reduction that can be tolerated is 100 -81.4 i.e. 18.6%. !

  Average power for 2x processor = 64.748 W.
  Average power for 4x processor = = 69.612 W.

  Now the average power from part 1 is 58.3115 W.

  Total energy consumed = power*cpu time
  =power*CPI*clock cycle time*instrs
  =(power*instrs)/(IPC*clock speed).

  Since the clock speed is unchanged, that will be a constant term right across the equation. Let that constant be k.The number of instructions will also be constant and let us assume it to be n.

  Energy consumed by the processor in a.cfg = 58.3115*n/(1.18*k)=49.416*n/k
  Energy consumed by the 2x processor = 64.748*n/(1.417*k)=45.693*n/k
  Energy consumed by the 4x processor = 69.61*n/(1.462*k)=47.61*n/k
  Hence from the above equation we see that the 2x processor consumes lesser energy than the energy consumed by a.cfg. Also the 4x processor also consumes lesser energy than the processor in a.cfg. Thus the new processor consumes less energy for a given task.

3 Question 3

- When I increased the widths of the decode,issue and commit pipeline stages by 2x, the IPC of the four benchmark programs were :

Benchmark Program	IPC	Power Consumed
mgrid	0.485	128.605
crafty	1.513	110.743
mesa	2.657	138.090
parser	0.432	97.855

Table 7: IPC's of four benchmark programs after increasing the width of decode, issue and commit pipeline stages

- Note that I had also doubled the functional units while running this simulation.
- Thus the arithmetic mean of IPC or the overall IPC in this case is **1.2715** and the average power consumed is **118.823** W
- We see that IPC of the new processor is lesser than the IPC of the processor obtained by increasing the size of structures by 2x or 4x(1.417 or 1.4625). Also the average power consumed by the processor is almost double as compared to processors in question 2. Hence for improving IPC, it is better to increase the sizes of the structure than to increase the width of the pipelines.

4 Question 4

When we change the memory latency from 300 to 500 cycles, the IPC values which were observed for the four benchmark programs were :

Benchmark Program	IPC
mgrid	0.469
crafty	1.279
mesa	2.002
parser	0.469

Table 8: IPC's of four benchmark programs after changing the memory latency from 300 to 500 cycles

- When compared with the IPC values for the initial a.cfg file which had a IPC of 1.1865, the new IPC value is 1.0547. Hence the percentage decrease in IPC is given by :

  $\%Decrease = \frac{OldValue - NewValue}{OldValue}$

  $\%Decrease = \frac{1.1865 - 1.0547}{1.1865}$

  $\%Decrease = 11.11\%$

- The changes which I made to increase the size of L2 cache were:

  -cache:dl2 ul2:16384:64:8:l
  -cache:dl2lat 32
  -cache:il2lat 30
  And memory access latency is still 500.

  After the changes made, the IPC's which were observed were:

Benchmark Program	**IPC**
mgrid	0.450
crafty	1.056
mesa	1.775
parser	0.432

Table 9: IPC's of four benchmark programs after increasing the size of L2 cache by 4x

- The arithmetic mean of the new IPC's are : 0.9285 This is much lower than the IPC which we had obtained by increasing the memory access latency to 500 which was 1.0547.

  The percentage decrease in IPC is : $\%Decrease = \frac{OldValue - NewValue}{OldValue}$

  $\%Decrease = \frac{1.0547 - 0.9285}{1.0547}$

  $\%Decrease = 11.9\%$

- Thus the notion of trying to make up for the loss in performance due to increase in memory access latency certainly cannot be overcome by increasing or doubling the size of L2 caches.

- The reason for this may be that when we increase the size of the L2 cache there is a chance that the cache miss rate may go down by a small factor like say come down from 10 MPKIs to probably 5MPKIs. But for this to happen a huge number of instructions would/could have to traverse a larger cache which increases the processor time and hence lesser number of instructions are executed per cycle and hence IPC goes down with increased cache size.