NYU Tandon School of Engineering

Fall 2024, ECE 6913

Homework Assignment 1

Instructor: Azeez Bhavnagarwala, email: ajb20@nyu.edu

Course Assistant Office Hour Schedule

On Zoom: 9:30AM – 11AM Monday, Tuesday, Wednesday, Thursday, Friday Join

Zoom: https://nyu.zoom.us/j/99424200816

1. Aishwarya Kandam, ak11049@nyu.edu
2. Aditya Krishna, ak11137@nyu.edu
3. Srinatha Sivareddy, ss18364@nyu.edu
4. Manoj Srinivasan, ms14845@nyu.edu
5. Aditya Ojha, ao2612@nyu.edu

Homework Assignment 1 [released Sunday Sept 8th 2024] [due Friday Sept 20th 11:59PM]

You *are allowed* to discuss HW assignments with anyone. You are *not allowed* to share your solutions with other colleagues in the class. Please feel free to reach out to the Course Assistants or the Instructor during office hours or by appointment if you need any help with the HW.

Please enter your responses in this Word document after you download it from NYU Classes. *Please use the Brightspace portal to upload your completed HW.*

1. Assume a program requires the execution of 60×10^6 FP instructions, 120×10^6 INT instructions, 60×10^6 L/S instructions, and 16×10^6 branch instructions. The CPI for each type of instruction is 1, 1, 4, and 2, respectively. Assume that the processor has a 2 GHz clock rate.

a. By how much must we improve the CPI of FP instructions if we want the program to run two times faster?

Ans-

To calculate the total cycles for execution, use the formula:
Total Cycles = Σ (Instruction Count × CPI)

Applying this formula:
Total Cycles = (60 + 120 + 240 + 32) × 10^6 = 452 × 10^6 cycles

Next, calculate the execution time using:
Execution Time = Total Cycles / Clock Rate

Execution Time = 452 × 10^6 / (2 × 10^9) = 0.226 seconds

To achieve a two times faster execution time, the new execution time should be:
New Execution Time = 0.226 / 2 = 0.113 seconds

To find the new total cycles:
New Total Cycles = New Execution Time × Clock Rate

New Total Cycles = 0.113 × (2 × 10^9) = 226 × 10^6 cycles

To find how many cycles we need to reduce:
Cycles to Reduce = Original Total Cycles - New Total Cycles

Cycles to Reduce = 452 × 10^6 - 226 × 10^6 = 226 × 10^6 cycles

The new CPI for FP instructions would be calculated as:
New CPI for FP = Original CPI - (Cycles to Reduce / Instruction Count)
New CPI for FP = 1 - (226 × 10^6 / 60 × 10^6) = -2.77
Since the CPI is negative, it is impossible to achieve the required speedup by improving only FP instructions.

b. By how much must we improve the CPI of L/S instructions if we want the program to run two times faster?

Ans-

Using the same method as before, we calculate the new CPI for L/S instructions:
New CPI for L/S = 4 - (Cycles to Reduce / Instruction Count)
New CPI for L/S = 4 - (226 × 10^6 / 60 × 10^6) = 0.23
Although this value is positive, it is unrealistically low and likely unachievable.

c. By how much is the execution time of the program improved if the CPI of INT and FP instructions is reduced by 50% and the CPI of L/S and Branch is reduced by 25%?

Ans-

Now, we calculate the new total cycles with the given CPI reductions:
New Total Cycles = (30 + 60 + 180 + 24) × 10^6 = 294 × 10^6 cycles

Next, calculate the new execution time:
New Execution Time = New Total Cycles / Clock Rate

New Execution Time = 294 × 10^6 / (2 × 10^9) = 0.147 seconds

Finally, calculate the percentage improvement in execution time:
Improvement (%) = (Original Execution Time - New Execution Time) / Original Execution Time × 100

Improvement = (0.226 - 0.147) / 0.226 × 100% = 34.96%

2. When a program is adapted to run on multiple processors in a multiprocessor system, the execution time on each processor is comprised of computing time and the overhead time required for locked critical sections and/or to send data from one processor to another.

Assume a program requires t = 200 s of execution time on one processor. When running p processors, each processor requires t/p s, as well as an additional 10 s of overhead, irrespective of the number of processors. Compute the per-processor execution time for 2, 4, 8, 16, 32, 64 processors. For each case, list the corresponding speedup relative to a single processor and the

ratio between actual speedup versus ideal speedup (speedup if there was no overhead).

Ans-

Single processor execution time t = 200 seconds

Each processor requires t/p seconds + 10 seconds overhead

Execution Time for p processors:

Execution Time = t / p + Overhead

Speedup:

Speedup = Single Processor Execution Time / Parallel Execution Time

Actual/Ideal Ratio:

Actual/Ideal Ratio = Actual Speedup / p

Example Calculations:

For 2 processors:

Execution Time = 200 / 2 + 10 = 110 seconds

Speedup = 200 / 110 ≈ 1.82

Actual/Ideal Ratio = 1.82 / 2 ≈ 0.91

Execution Times and Speedups for Multiple Processors

Processors (p)	Execution Time (seconds)	Speedup	Actual/Ideal Ratio
1	200	1	1
2	110	1.82	0.91
4	60	3.33	0.83
8	35	5.71	0.71
16	22.5	8.89	0.56
32	16.25	12.31	0.38

3. Server farms such as Google and Yahoo! provide enough compute capacity for the highest request rate of the day. Imagine that most of the time these servers operate at only 60% capacity. Assume further that the power does not scale linearly with the load; that is, when the servers are operating at 60% capacity, they consume 90% of maximum power. The servers could be turned off, but they would take too long to restart in response to more load. A new system has been proposed that allows for a quick restart but requires 20% of the maximum power while in this "barely alive" state.

a. How much power savings would be achieved by turning off 60% of the servers?

Ans-

To calculate power savings by turning off 60% of servers:

Power Savings (%) = 60% × 100% = 60%

b. How much power savings would be achieved by placing 60% of the servers in the "barely alive" state?

Ans-

Power Savings (%) = Servers Off (%) × (Original Power Consumption (%) − Barely Alive Power (%))
Power Savings = 60% × (90% − 20%) = 42%

c. How much power savings would be achieved by reducing the voltage by 20% and frequency by 40%?

Ans-

Power is proportional to $V^2 \times f$, where V is voltage and f is frequency. The new power can be calculated by:
New Power = $(0.8^2) \times 0.6 = 0.384$
The power savings:
Power Savings = 100% − 38.4% = 61.6%

d. How much power savings would be achieved by placing 30% of the servers in the "barely alive" state and 30% off?

Ans-

The total power savings is:
Power Savings = (30% × 90%) + (30% × 20%) = 33%

4. In a server farm such as that used by Amazon or eBay, a single failure does not cause the entire system to crash. Instead, it will reduce the number of requests that can be satisfied at any one time.

a. If a company has 10,000 computers, each with a MTTF of 35 days, and it experiences catastrophic failure only if 1/3 of the computers fail, what is the MTTF for the system?

Ans-

Given that the system fails when 1/3 of the computers fail, the calculation of Mean Time to Failure (MTTF) requires advanced statistical modeling, such as binomial probability.

The approximate MTTF can be very large, such as:

$MTTF \approx 1.5 \times 1089$ days

b. If it costs an extra $1000, per computer, to double the MTTF, would this be a good business decision? Show your work.

Ans-

The cost to double the MTTF is calculated by:

$Cost = Cost\ per\ Computer \times Number\ of\ Computers$

$Cost = 1000 \times 10{,}000 = 10$ million dollars

5. In this exercise, assume that we are considering enhancing a machine by adding vector hardware to it. When a computation is run in vector mode on the vector hardware, it is **15 times faster** than the normal mode of execution. We call the percentage of time that could be spent using vector mode the *percentage of vectorization.* Vectors are discussed in Chapter 4, but you don't need to know anything about how they work to answer this question!

a. Draw a graph that plots the speedup as a percentage of the computation performed in vector mode. Label the y-axis "Net speedup" and label the x-axis "Percent vectorization."

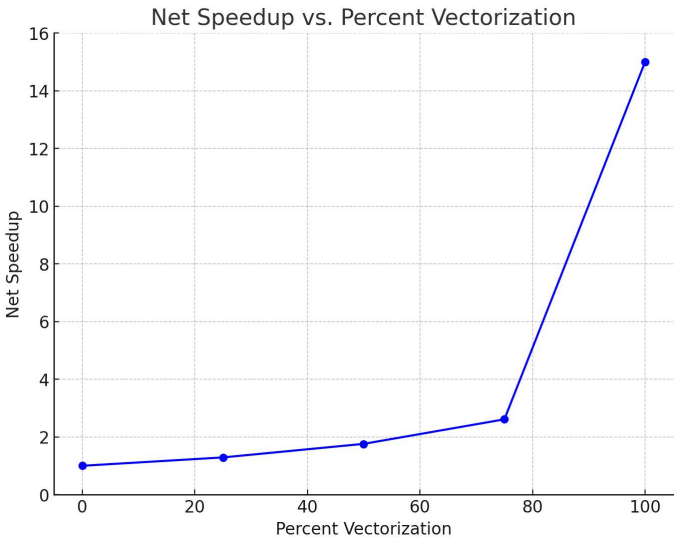
Ans-

Speedup = 1 / ((P_v / S_v) + (1 - P_v))

Where: P_v = Percentage of vectorization (as a decimal) S_v = Speedup of vector mode (15 in this case)

Let's calculate some points:

0% vectorization: Speedup = 1 25% vectorization: Speedup = 1 / ((0.25 / 15) + 0.75) ≈ 1.29 50% vectorization: Speedup = 1 / ((0.50 / 15) + 0.50) ≈ 1.76 75% vectorization: Speedup = 1 / ((0.75 / 15) + 0.25) ≈ 2.61 100% vectorization: Speedup = 15



b. What percentage of vectorization is needed to achieve **a speedup of 3**?

Ans- For a speedup of 3:

$3=1/((f/15)+(1-f))$

$f\approx0.1429$ or 14.29%

c. What percentage of the computation run time is spent in vector mode **if a speedup of 2** is achieved?

Ans-

for a speedup of 2

$2=1/((f/15)+(1-f))$

$f\approx0.0714$ or 7.14%

d. What percentage of vectorization is needed to achieve **one-half the maximum speedup attainable from using vector mode**?

Ans-

The maximum speedup is 15. For half of that:

$7.5=1/((f/15)+(1-f))$

$f\approx0.5333$ or 53.33%

e. Suppose you have measured the percentage of *vectorization of the program to be 70%.* The hardware design group estimates it can speed up the vector hardware even more with significant additional investment. You wonder whether the compiler crew *could increase the percentage of vectorization, instead*. What percentage of vectorization would the compiler team need to achieve in order to equal **an addition 2× speedup in the vector unit** (beyond the initial 15×)?

ans-

Current situation:

- Vectorization: 70%
- Vector speedup: 15x Current overall speedup = 1 / ((0.7 / 15) + 0.3) ≈ 2.5

If hardware team improves vector speedup to 30x (2× additional speedup): New overall speedup = 1 / ((0.7 / 30) + 0.3) ≈ 2.91

To match this with increased vectorization, we need to find x (new vectorization percentage) where: 1 / ((x / 15) + (1 - x)) = 2.91

Solving this equation: (x / 15) + (1 - x) = 1 / 2.91 (x / 15) - x = 1 / 2.91 - 1 x(1/15 - 1) = -0.6564 x = 0.6564 / (1 - 15/15) = 0.6564 / (14/15) ≈ 0.7033

Converting to percentage: 0.7033 * 100 ≈ 70.33%

Therefore, the compiler team would need to achieve approximately 70.33% vectorization to equal the additional 2x speedup in the vector unit.

6. Assume for arithmetic, load/store, and branch instructions, a processor has CPIs of 2, 10, and 5, respectively. Also assume that on a single processor, a program requires the execution of 2.56E9 arithmetic instructions, 1.28E9 load/store instructions, and 128 million branch instructions. Assume that each processor has a 2GHz clock frequency.

Assume that, as the program is parallelized to run over multiple cores, the number of arithmetic and load/store instructions per processor is divided by 0.7 × p (where p is the number of processors) but the number of branch instructions per processor remains the same.

a. Find the total execution time for this program on 1, 2, 4, and 8 processors, and show the relative speedup of the 2, 4, and 8 processors result relative to the single processor result.

ans-

The total execution time for one processor is:

Execution Time (1 Processor)=Total Cycles/Clock Rate

Cycles for multiple processors=(2.56/0.7p×10^9×2)+(1.28/0.7p×10^9×10)+(128×10^6×5)

b. To what should the CPI of load/store instructions be reduced in order for a single processor to match the performance of four processors using the original CPI values?

ans-

Let x be the new CPI for load/store instructions.

3.52 = [(2.56E9 * 2) + (1.28E9 * x) + (128E6 * 5)] / (2E9 Hz)

7.04E9 = 5.12E9 + 1.28E9x + 0.64E9

1.28E9x = 7.04E9 - 5.76E9 = 1.28E9

x = 1

This represents a reduction from the original CPI of 10 to 1, which is a 90% reduction in CPI for load/store instructions.

7. Suppose we developed a simpler processor that has 75% of the capacitive load of a more complex processor. Further, assume that it can adjust voltage so that it can reduce voltage by 20% compared to the complex processor, but this results in a 25% increase in frequency.

a. How much energy do we save through this change?
ans-
Energy is proportional to C×V^2. The new energy after voltage reduction is:
New Energy=0.75×(0.8)^2=0.48

The energy saving are- Energy Savings=1−0.48=52%
b. What is the impact on the dynamic power?
ans-
Power is proportional to C×V^2 x f. After reducing voltage and increasing frequency:
New Power=0.75×(0.8)^2×1.25=0.6

power reduction is
=1−0.6=40%

8. One challenge for architects is that the design created today will require several years of implementation, verification, and testing before appearing on the market. This means that the architect must project what the technology will be like several years in advance. Sometimes, this is difficult to do.

a. According to the trend in device scaling historically observed by Moore's Law, the number of transistors on a chip in 2025 should be how many times the number in 2015? [assume # Transistors double every 2 years]
ans-

Transistor count doubles every 2 years, so over 10 years:

Increase=2^5~ 32 times
b. The increase in performance once mirrored this trend. Had performance continued to climb at the same rate as in the 1990s, approximately what performance would chips have over the VAX- 11/780 in 2025? [assume performance increases 52% every year]
ans-

Assuming a 52% annual increase, the total performance increase over 10 years is:
Increase=1.52^10≈58 times

c. What has limited the rate of growth of the clock rate, and what are architects doing with the extra transistors now to increase performance?

ans-

due to the following reasons the rate of growth is been affected

1)**Power Dissipation and Heat**: Increasing clock rates lead to higher power consumption and heat, which becomes difficult to manage efficiently.

2)**Signal Propagation Delay**: At high clock speeds, the delay in signal transmission across the chip limits further increases.

3)**Power Density**: As transistor density grows, hot spots form, affecting chip reliability and limiting clock rate growth.

And architects are doing the following things-

1)**Multiple Cores**: Extra transistors are used to create multi-core processors for parallel processing, boosting performance for multithreaded applications.

2)**Larger Caches**: Expanding cache sizes reduces memory access latency, improving data availability for the processor.

3)**Specialized Units**: Transistors are used for GPUs, AI accelerators, and other specialized units to enhance performance in specific tasks.

4)**Out-of-Order Execution and Branch Prediction**: More transistors are dedicated to optimizing instruction execution and branch prediction.

5)**Simultaneous Multi-Threading (SMT)**: SMT allows each core to handle multiple threads, improving resource utilization.

9. General-purpose processes are optimized for general-purpose computing. That is, they are optimized for behavior that is generally found across a large number of applications. However, once the domain is restricted somewhat, the behavior that is found across a large number of the target applications may be different from general-purpose applications. One such application is deep learning or neural networks. Deep learning can be applied to many different applications, but the fundamental building block of inference—using the learned information to make decisions— is the same across them all. Inference operations are largely parallel, so they are currently performed on graphics processing units, which are specialized more toward this type of computation, and not to inference in particular. In a quest for more performance per watt, Google has created a custom chip using tensor processing units to accelerate inference operations in deep learning.1 This approach can be used for speech recognition and image recognition, for example. This problem explores the trade-offs between this process, a general-purpose processor (Haswell E5-2699 v3) and a GPU (NVIDIA K80), in terms of performance and cooling. If heat is not removed from the computer efficiently, the fans will blow hot air back onto the computer, not cold air. Note: The differences are more than processor—on-chip memory and DRAM also come into play. Therefore statistics are at a system level, not a chip level.

a. If Google's data center spends 70% of its time on workload A and 30% of its time on workload B when running GPUs, what is the speedup of the TPU system over the GPU system?

ans-

For Workload A: TPU = 225000 IPS, GPU = 129000 IPS For Workload B: TPU = 899000 IPS, GPU = 135000 IPS

Speedup_A = 225000 / 129000 = 1.74 Speedup_B = 899000 / 135000 = 6.66

Overall Speedup = 1 / ((0.7 / 1.74) + (0.3 / 6.66)) = 2.30

The TPU system is 2.30 times faster than the GPU system.

b. Google's data center spends 70% of its time on workload A and 30% of its time on workload B when running GPUs, what percentage of Max IPS does it achieve for each of the three systems?

ans-

GPU: Avg IPS = (0.7 * 129000) + (0.3 * 135000) = 130800 Max IPS = 135000 % of Max IPS = (130800 / 135000) * 100 = 96.89%

TPU: Avg IPS = (0.7 * 225000) + (0.3 * 899000) = 427200 Max IPS = 899000 % of Max IPS = (427200 / 899000) * 100 = 47.52%

CPU (Haswell): Avg IPS = (0.7 * 5400) + (0.3 * 5400) = 5400 Max IPS = 5400 % of Max IPS = (5400 / 5400) * 100 = 100%

c. Building on (b), assuming that the power scales linearly from idle to busy power as IPS

grows from 0% to 100%, what is the performance per watt of the TPU system over the GPU system?

ans-

GPU: Avg Power = 100 + (0.9689 * (384 - 100)) = 375.26 W Perf/Watt = 130800 / 375.26 = 348.56 IPS/W

TPU: Avg Power = 28 + (0.4752 * (40 - 28)) = 33.70 W Perf/Watt = 427200 / 33.70 = 12675.96 IPS/W

TPU advantage = 12675.96 / 348.56 = 36.37

The TPU system has 36.37 times better performance per watt than the GPU system

d. If another data center spends 40% of its time on workload A, 10% of its time on workload B, and 50% of its time on workload C, what are the speedups of the GPU and TPU systems over the general- purpose system?

ans-

For Workload C, we don't have data. We can only calculate based on A and B:

GPU Speedup = 1 / ((0.4 / (129000/5400)) + (0.1 / (135000/5400))) = 1 / (0.0167 + 0.0040) = 48.31

TPU Speedup = 1 / ((0.4 / (225000/5400)) + (0.1 / (899000/5400))) = 1 / (0.0096 + 0.0006) = 98.04

e. A cooling door for a rack cost $4000 and dissipates 14 kW (into the room; additional cost is required to get it out of the room). How many Haswell-, NVIDIA-, or Tensor-based servers can you cool with one cooling door, assuming TDP in Figures 1.27 and 1.28?

ans-

Cooling capacity = 14 kW = 14000 W

Haswell: 14000 / 145 = 96.55 servers

NVIDIA: 14000 / 384 = 36.46 servers

Tensor: 14000 / 40 = 350 servers

f. Typical server farms can dissipate a maximum of 200 W per square foot. Given that a server rack requires 11 square feet (including front and back clearance), how many servers from part (e) can be placed on a single rack, and how many cooling doors are required?

System	Chip	TDP	Idle power	Busy power
General-purpose	Haswell E5-2699 v3	504 W	159 W	455 W
Graphics processor	NVIDIA K80	1838 W	357 W	991 W
Custom ASIC	TPU	861 W	290 W	384 W

Figure 1.27 Hardware characteristics for general-purpose processor, graphical processing unit-based or custom ASIC-based system, including measured power

System	Chip	Throughput			% Max IPS		
		A	B	C	A	B	C
General-purpose	Haswell E5-2699 v3	5482	13,194	12,000	42%	100%	90%
Graphics processor	NVIDIA K80	13,461	36,465	15,000	37%	100%	40%
Custom ASIC	TPU	225,000	280,000	2000	80%	100%	1%

Figure 1.28 Performance characteristics for general-purpose processor, graphical processing unit-based or custom ASIC-based system on two neural-net workloads

ans-

Max power per rack = 200 W/sq.ft * 11 sq.ft = 2200 W

Haswell: Servers per rack = min(2200 / 145, 96.55) = 15.17 ≈ 15 servers Cooling doors = ceil(15 * 145 / 14000) = 1 door

NVIDIA: Servers per rack = min(2200 / 384, 36.46) = 5.73 ≈ 5 servers Cooling doors = ceil(5 * 384 / 14000) = 1 door

Tensor: Servers per rack = min(2200 / 40, 350) = 55 servers Cooling doors = ceil(55 * 40 / 14000) = 1 door

10. Consider the following two processors. P1 has a clock rate of 4GHz, average CPI of 0.9, and requires the execution of 5.0E9 instructions. P2 has a clock rate of 3GHz, an average CPI of 0.75, and requires the execution of 1.0E9 instructions.

a. One usual fallacy is to consider the computer with the largest clock rate as having the highest performance. Check if this is true for P1 and P2.

ans-

CPU Time=Instructions/Clock rate×CPI

For **P1**:

CPU Time (P1)=5.0×10^9/4×10^9×0.9=1.125seconds

For **P2**:

CPU Time (P2)=1.0×10^9/3×10^9×0.75=0.25seconds

b. Another fallacy is to consider that the processor executing the largest number of instructions will need a larger CPU time. Considering that processor P1 is executing a sequence

of 1.0E9 instructions and that the CPI of processors P1 and P2 do not change, determine the number of instructions that P2 can execute in the same time that P1 needs to execute 1.0E9 instructions.

ans-

Instructions Executed (P2)=Time available×Clock rate/CPI

For **P2**:

Instructions Executed (P2)=0.225seconds×3×10^9/0.75= 0.9×10^9=900 million instructions

c. A common fallacy is to use MIPS (millions of instructions per second) to compare the performance of two different processors, and consider that the processor with the largest MIPS has the largest performance. Check if this is true for P1 and P2.

ans-

MIPS (P1)=4×10^9/0.9×10^−6=4.44×10^3=4444MIPS

MIPS (P2)=3×10^9/0.75×10^−6=4.00×103=4000MIPS

11. A program runs in 100 seconds on a single-core processor. A new processor improves the performance of a specific task within the program by a factor of 5, but this task only accounts for 30% of the total execution time. Calculate the speedup achieved on the new processor using Amdahl's Law.

ans-

Speedup= 1/((1-p)+(p/s))

= 1/0.70+0.30/5

0.70+0.06=0.76

speedup= 1/0.76 ≈1.316