HW1

1. Consider two different implementations of the same instruction set architecture. The instructions can be divided into four classes according to their CPI (classes A, B, C, and D). P1 with a clock rate of 2.0 GHz and CPIs of 1, 2, 2, and 1, and P2 with a clock rate of 4 GHz and CPIs of 2, 3, 4, and 4.
    - Given a program with a dynamic instruction count of 1.0E6 instructions divided into classes as follows: 10% class A, 20% class B, 50% class C, and 20% class D. Which is faster: P1 or P2 (in total execution time)?

	A(10%)	B(20%)	C(50%)	D(20%)
P1	1	2	2	1
P2	2	3	4	4

      We can get the global CPI for P1 and P2, they are
      $P1_{CPI} = 1 \times 10\% + 2 \times 20\% + 2 \times 50\% + 1 \times 20\% = 1.7$
      $P2_{CPI} = 2 \times 10\% + 3 \times 20\% + 4 \times 50\% + 4 \times 20\% = 3.6$
      We have 10^6 instructions and now we will calculate their execution time
      $P1_{time} = \frac{1.7 \times 10^6}{2 \times 10^9} = 8.5 \times 10^{-4} s$
      $P2_{time} = \frac{3.6 \times 10^6}{4 \times 10^9} = 9 \times 10^{-4} s$
      so, we can see that P1 is faster
    - What is the global CPI for each implementation?

      We can get the global CPI for P1 and P2, they are
      $P1_{CPI} = 1 \times 10\% + 2 \times 20\% + 2 \times 50\% + 1 \times 20\% = 1.7$
      $P2_{CPI} = 2 \times 10\% + 3 \times 20\% + 4 \times 50\% + 4 \times 20\% = 3.6$
    - Find the clock cycles required in both cases.

      clock cycles required = CPI × IC
      $P1_{cycles} = 1.7 \times 1 \times 10^6 = 1.7\text{M cycles}$
      $P2_{cycles} = 3.6 \times 1 \times 10^6 = 3.6\text{M cycles}$
    - Which processor has the highest throughput performance (instructions per second) ?
      $P1_{throughout} = \frac{2B}{1.7} = 1.176 \times 10^9$
      $P2_{throughout} = \frac{4B}{3.6} = 1.11 \times 10^9$
      we can see that $P1$ has the highest throughput performance
    - Which processor do you think is more energy efficient? Why?

      P1, lower CPI is higher energy efficiency.

2. You are designing a system for a real-time application in which specific deadlines must be met. Finishing the computation faster gains nothing. You find that your system can execute the necessary code, in the worst case, twice as fast as necessary.
    - How much energy do you save if you execute at the current speed and turn off the system when the computation is complete?

We know the formula $E = \frac{CV^2}{2}$. So, it is not relative with speed. It does not save the energy

- How much energy do you save if you set the voltage and frequency to be half as much?

  The formula is $E = \frac{CV^2}{2}$. We can see that the energy is not relative to frequency. So, we can get

  $$\frac{E_{new}}{E_{old}} = \frac{C_{old} \times (0.5 \times V_{old})^2}{C_{old} \times V_{old}^2} = \frac{1}{4}$$

  so, for energy, it saves 75%.

  $$\frac{P_{new}}{P_{old}} = \frac{C_{old} \times (0.5 \times V_{old})^2 \times (0.5 \times F_{old})}{C_{old} \times V_{old}^2 \times F_{old}} = 0.125$$

  We can see that $\frac{P_{new}}{P_{old}} = \frac{1}{8}$, so It saved around 87.5%.

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.

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

     we can calculate how many percent of maximum power need to be consumed if we turn off 60% of the servers. We can get

     $$\begin{aligned} P_{old} &= 1 \times 60\% \times V^2 \times F = Max \times 90\% \\ P_{new} &= 1 \times 60\% \times 60\% \times V^2 \times F = Max \times x\% \\ x &= 54 \end{aligned}$$

     And we can know that it saves

     $$\frac{54}{90} = 60\%$$

     so, it saved 60%

   - How much power savings would be achieved by placing 60% of the servers in the “barely alive” state?

     we can know that 40% still need to consume 90% of maximum power. But the other 60% only need 20% of maximum power to maintain barely alive state.

     we can get $0.4 \times 0.9 + 0.6 \times 0.2 = 0.48$

     And we can know that it saves

     $$\frac{0.48}{0.9} = 53\%$$

     So, it reduces to around 53%, so it saves 0.9-0.53=37%.

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

     $$\frac{P_{new}}{P_{old}} = \frac{C_{old} \times (0.8 \times V_{old})^2 \times (0.6 \times F_{old})}{C_{old} \times V_{old}^2 \times F_{old}} = 0.384$$

     So, it saves 61.6% power

- How much power savings would be achieved by placing 30% of the servers in the “barely alive” state and 30% off?

  here, we will assume the 40% will still consume 90% of maximum power, we can get that $0.4 \times 0.9 + 0.3 \times 0.2 + 0.3 \times 0 = 0.42$.

  And we can know it saves

  $$\frac{0.42}{0.9} = 0.467$$

  So, it reduces to around 46.7%, so it saves around 43.3%.

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.

   - 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?

     For each computer, the $FIT_{computer}$ is $\frac{1}{MTTF} = \frac{1}{35}$ per day. so if 1/3 computers fail, then the $FIT_{system} = 3 \times \frac{1}{35}$ per day. So the MTTF of the whole system is $1/(3 \times \frac{1}{35}) = 11.67$ days.

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

     when double the MTTF, it will be 70 days, then for each computer, the FIT is $\frac{1}{MTTF} = \frac{1}{70}$. so if 1/3 computers fail, then the MTTF of the whole system is $1/(3 \times \frac{1}{70}) = 23.3$ days. We need to pay 10,000,000 to improve the server and we can double the MTTF. So, if the cost of each computer is less than $1000, we can replace them. If not, we can simply use $1000 to double MTTF.

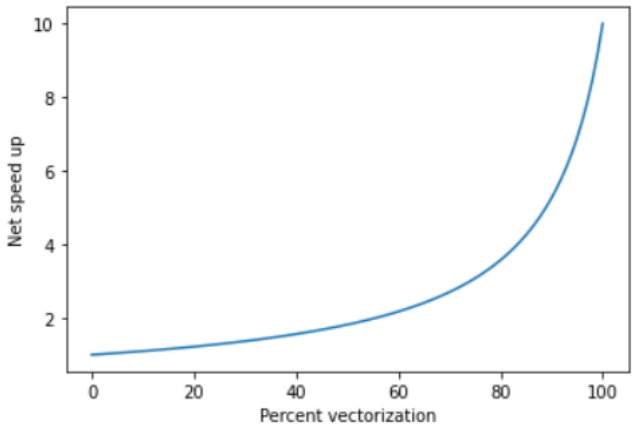
5. In this exercise, assume that we are considering enhancing a machine by adding vector hardware to it. When a computation is run in vector mode on the vector hardware, it is 10 times faster than the normal mode of execution. We call the percentage of time that could be spent using vector mode the *percentage of vectorization.* Vectors are discussed in Chapter 4, but you don’t need to know anything about how they work to answer this question!

   - 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.”

     So, we can set original time is t, after we use vectorization, the time will be reduced to $(100\% - x)t + \frac{x}{10}t$, where x is the percentage of vectorization. so we can get

     $$y = \frac{1}{(1-x) + 0.1x}$$

     The graph is shown below



- What percentage of vectorization is needed to achieve a speedup of 2?

  when y=2, x=0.55. So, it is around 55%

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

  so, we know that it is around 55% vectorization needed to achieved speedup of 2. so the percentage of the computation of runtime is spent in vector is

  $$\frac{\frac{55\%}{10}t}{(100\% - 55\%)t + \frac{55\%}{10}t} = 11\%$$

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

  The maximum speedup is 10, one-half is 5. so, when y=5, x= 88.89%. So, it needs 88.89% vectorization.

- 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 10×)?

  When the vectorization is 70%, then the speed up(y) is y=2.7x.

  it said we need to achieve additional 2x speedup, which means double. so it is 5.4x speedup, the total percentage of vectorization is 91.53%.

6. Two questions

- (i) A program (or a program task) takes 150 million instructions to execute on a processor running at 2.7 GHz. Suppose that 50% of the instructions execute in 3 clock cycles, 30% execute in 4 clock cycles, and 20% execute in 5 clock cycles. What is the execution time for the program or task?

  we can calculate how many cycles totally we need to execute the instructions

  $$150M \times 0.5 \times 3 + 150M \times 0.3 \times 4 + 150M \times 0.2 \times 5 = 5.55 \times 10^8 \text{cycles}$$

  and the process is 2.7GHz, so, we need

  $$\frac{5.55 \times 10^8}{2.7 \times 10^9} = 2.056 \times 10^{-1} s$$

- (ii) Suppose the processor in the previous question part is redesigned so that all instructions that initially executed in 5 cycles and all instructions executed in 4 cycles now execute in 2 cycles. Due to changes in the circuitry, the clock rate also must be decreased from 2.7 GHz to 2.1 GHz. What is the overall percentage improvement?

  we can calculate how many cycles totally we need to execute the instructions

  $150M \times 0.5 \times 3 + 150M \times 0.3 \times 2 + 150M \times 0.2 \times 2 = 3.75 \times 10^8 \text{cycles}$

  and the process is 2.1GHz, so, we need

  $$\frac{3.75 \times 10^8}{2.1 \times 10^9} = 1.786 \times 10^{-1} s$$

  so, $\frac{2.056 \times 10^{-1}}{1.786 \times 10^{-1}} = 1.15$. So, we get around 15% improvement.

7. Assume for a given processor the CPI of arithmetic instructions is 1, the CPI of load/store instructions is 10, and the CPI of branch instructions is 3. Assume a program has the following instruction breakdowns: 500 million arithmetic instructions, 300 million load/store instructions, 100 million branch instructions.

   - Suppose that new, more powerful arithmetic instructions are added to the instruction set. On average, through the use of these more powerful arithmetic instructions, we can reduce the number of arithmetic instructions needed to execute a program by 25%, while increasing the clock cycle time by only 10%. Is this a good design choice? Why?

     Execution time = CPI × IC × Cycle

     So, we can get

     $$\frac{new}{old} = \frac{(500M \times 0.75 \times 1 + 300M \times 10 + 100M \times 3) \times cycle \times 1.1}{(500M \times 1 + 300M \times 10 + 100M \times 3) \times cycle} = \frac{4042.5}{3800}$$

     it is not a good design, we can see that the new design performance is worse than old design.

   - Suppose that we find a way to double the performance of arithmetic instructions. What is the overall speedup of our machine? What if we find a way to improve the performance of arithmetic instructions by 10 times?

     Double the performance of arithmetic, the CPI is 1/2, so can get

     $$\frac{new}{old} = \frac{(500M \times 1/2 + 300M \times 10 + 100M \times 3) \times cycle}{(500M \times 1 + 300M \times 10 + 100M \times 3) \times cycle} = \frac{3550}{3800}$$

     It improves around 7%.

     If it improves the performance of arithmetic instructions by 10 times, then we can get

     $$\frac{new}{old} = \frac{(500M \times 1/10 + 300M \times 10 + 100M \times 3) \times cycle}{(500M \times 1 + 300M \times 10 + 100M \times 3) \times cycle} = \frac{3350}{3800}$$

     So, it improves 13.4%

8. After graduating, you are asked to become the lead computer designer at Hyper Computers, Inc. Your study of usage of high-level language constructs suggests that procedure calls are one of the most expensive operations. You have invented a scheme that reduces the loads and stores normally associated with procedure calls and returns. The first thing you do is run some experiments with and without this optimization. Your experiments use the same state-of-the-art optimizing compiler that will be used with either version of the computer. These experiments reveal the following information:

   - The clock rate of the unoptimized version is 5% higher.

- 30% of the instructions in the unoptimized version are loads or stores.
- The optimized version executes 2/3 as many loads and stores as the unoptimized version. For all other instructions the dynamic counts are unchanged.
- All instructions (including load and store) take one clock cycle.

  Which is faster? Justify your decision quantitatively.

  So, we can summarize the information. The clock rate of unoptimized one is 5% higher which means unoptimized one is 5% faster. The CPI=1. We set clock rate of unoptimized one is C, then the optimized one is 0.95C. And according to other information, we can write the formula as below

  $$\frac{opt}{unopt} = \frac{\frac{(\frac{2}{3}\times 30\%\times 1+70\%\times 1)\times IC}{C\times 0.95}}{\frac{(30\%\times 1+70\%\times 1)\times IC}{C}} = \frac{0.9}{0.95}$$

  We can see that optimized one is faster, we get around 5.5% improvement.

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.

   - 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?

     When we calculate the speedup, we are actually calculating the executing time, that is performance. We set two variables, ET_{GPU}, ET_{TPU}. Speedup of TPU over GPU. $\frac{ET_{GPU}}{ET_{TPU}}$. So, we need to check the throughput(the number of task in a fixed time) of workload A and workload B. From Figure 1.28, we know that for workload A, the GPU throughout is 13,461 and the TPU throughput is 225,000. And for workload B, the GPU throughput is 36,465 and the TPU throughput is 280,000.

     From it, we can know that For workload A, TPU is $\frac{225,000}{13,467} = 16.71$ faster than GPU. For workload B, we can know that TPU is $\frac{280,000}{36,456} = 7.68$ faster than GPU. So, we can get the relationship

     $$ET_{TPU} = 16.71 \times ET_{GPU} \text{ (For A)}$$
     $$ET_{TPU} = 7.68 \times ET_{GPU} \text{ (For B)}$$

     When data center spends 70% of its time on workload A and 30% of its time on workload B when running GPUs, we can get

$$ET_{TPU} = (\frac{0.7}{16.71} + \frac{0.3}{7.68}) \times ET_{GPU}$$

we can get

$$\frac{ET_{GPU}}{ET_{TPU}} = \frac{1}{(\frac{0.7}{16.71} + \frac{0.3}{7.68})} = 12.34$$

- 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?

  For general-purpose, we can get $70\% \times 42\% + 30\% \times 100\% = 0.294 + 0.3 = 0.594$

  For GPU, we can get $70\% \times 37\% + 30\% \times 100\% = 0.259 + 0.3 = 0.559$

  For TPU, we can get $70\% \times 80\% + 30\% \times 100\% = 0.56 + 0.3 = 0.86$

- 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?

  $\text{performance per watt} = \frac{IPS}{power}$. We already know the the percentage of Max IPS we can achieve for TPU and GPU from above. So, now, we need to calculate the power. We know that the power scales linearly from idle to busy power. So we can actually get the formula that $power = idle + (\text{max power-idle power}) \times IPS$.

  So for GPU power, we can get $power_{GPU} = 357 + (991 - 357) \times 0.559 = 711w$

  for TPU power, we can get $power_{TPU} = 290 + (384 - 290) \times 0.86 = 371w$

  So, the performance per watt of TPU over GPU, we can get

  $\frac{P_{GPU}}{P_{TPU}} = 1/\frac{\frac{IPS_{GPU}}{power_{GPU}}}{\frac{IPS_{TPU}}{power_{TPU}}} = 1/0.339 = 2.95.$

- 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?

  Speedup of GPU over general-purpose system

  For workload A,B,C, they are $\frac{13,461}{5,482} = 2.456$, $\frac{35,465}{13,194} = 2.76$, $\frac{15,000}{12,000} = 1.25$ respectively.

  So

  $$\frac{ET_{general}}{ET_{GPU}} = \frac{1}{(\frac{0.4}{2.456} + \frac{0.1}{2.76} + \frac{0.5}{1.25})} = 1.669$$

  Speedup of TPU over general-purpose system

  For workload A,B,C, they are $\frac{225,000}{5,482} = 41$, $\frac{280,000}{13,194} = 21.22$, $\frac{2,000}{12,000} = 0.16$ respectively.

  So

  $$\frac{ET_{general}}{ET_{TPU}} = \frac{1}{(\frac{0.4}{41} + \frac{0.1}{21.22} + \frac{0.5}{0.16})} = 0.32$$

- A cooling door for a rack costs $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?

  TDP is thermal design power. We have 14kw.

  For Haswell, we can cool down $\frac{14,000}{504} = 27$

  For GPU, we can cool down $\frac{14,000}{1838} = 7$

For TPU, we can cool down $\frac{14,000}{861} = 16$

- 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?

  We need 11 square feet and 200w per square feet. So, totally, we can dissipate $11 \times 200 = 2,200$w.

  For CPU, we can place $\frac{2,200}{504} = 4$

  For GPU, we can place $\frac{2,200}{1838} = 1$

  For TPU, we can place $\frac{2,200}{861} = 2$.

  From (e). we know that a cooling door can dissipate 14KW, and we only need 2.2KW, so one cooling door is enough.

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