NYU Tandon School of Engineering

Fall 2022, ECE 6913

Homework Assignment 1

Instructor Contact : ajb20@nyu.edu

Course Assistants & Graders: Varadraj Kakodkar (vns2008), Kartikay Kaushik (kk4332), Siddhanth Iyer (si2152), Swarnashri Chandrashekar (sc8781), Karan Sheth (kk4332), Haotian Zheng (hz2687), Haoren Zhang (kk4332), Varun Kumar (vs2411)

Homework Assignment 1 [released: Tuesday Sept 6th] [due Friday Sept 16th by 11:59 PM]

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 Brightspace. *Please use the Brightspace portal to upload your completed HW.*

1. Consider three different processors P1, P2, and P3 executing the same instruction set. P1 has a 3 GHz clock rate and a CPI of 1.5. P2 has a 2.5 GHz clock rate and a CPI of 1.0. P3 has a 4.0 GHz clock rate and has a CPI of 2.2.

a. Which processor has the highest performance expressed in instructions per second?

Single Instruction execution time = CPI * Clocks Cycle Time = CPI / Clock Rate.

Instructions per seconds = 1 / Single Instruction execution time = Clock Rate / CPI

Thus,

P1: (3 * 10^9) /1.5= 2 * 10^9

P2: (2.5 * 10^9) / 1 = 2.5 * 10^9

P3: (4 * 10^9) / 2.2 = 1.818 * 10^9

Thus, Processor "P2" has the highest performance.

b. If the processors each execute a program in 10 seconds, find the number of cycles and the number of instructions.

Number of Cycles:

P1: 10s * 3GHz =3*10^10

P2: 10s * 2.5GHz =2.5*10^10

P3: 10s * 4GHz =4*10^10

Number of Instructions = Number of Cycles / CPI

P1: 3*10^10 / 1.5 = 2 * 10^10

P2: 2.5*10^10 / 1 = 2.5 * 10^10

P3: 4*10^10 / 2.2 = 1.818 * 10^10

c. We are trying to reduce the execution time by 30%, but this leads to an increase of 20% in the CPI. What clock rate should we have to get this time reduction?

The new execution time is 10 * (1 - 30%) = 7s.

Since the Clock Cycles = CPI * Number of Instructions,

Clock Rate = Clock Cycles / Execution time

The new clock rate for each processor should be:

P1: 2 * 10^10 *1.5*(1+20%) / 7 = 5.143 GHz

P2: 2.5 * 10^10 *1*(1+20%) / 7 = 4.286GHz

P3: 1.818 * 10^10 *2.2*(1+20%) / 7 = 6.856 GHz

2. Assume for arithmetic, load/store, and branch instructions, a processor has CPIs of 1, 12, 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 256 million **branch** instructions. Assume that each processor has a 2 GHz 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 (ET) 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.

Assume there is a total n processors.

n = 1:

Total Number of Cycles = 2.56*10^9 * 1 + 1.28 * 10^9 * 12 + 2.56 * 10^8 * 5 = 1.92 * 10^10

Total Execution Time = Total Number of Cycles / Clock Rate = 1.92 * 10^10 / 2GHz = 9.6s

n = 2:

The number of Cycles on single processor = (2.56*10^9 * 1 + 1.28 * 10^9 * 12) / (0.7 * 2) + 2.56 * 10^8 * 5 = 1.408 * 10^10

Thus, the Total Execution Time = Total Number of Cycles / Clock Rate = 1.408 * 10^10 / 2GHz = 7.04s

Compare to the single processor, it is speedup by 9.6s / 7.04s = 1.364 times.

n = 4:

The number of Cycles on single processor = (2.56*10^9 * 1 + 1.28 * 10^9 * 12) / (0.7 * 4) + 2.56 * 10^8 * 5 = 7.68* 10^9

Thus, the Total Execution Time = Total Number of Cycles / Clock Rate = 7.68* 10^9 / 2GHz = 3.84s

Compare to the single processor, it is speedup by 9.6s / 3.84s = 2.5 times.

n = 8:

The number of Cycles on single processor = (2.56*10^9 * 1 + 1.28 * 10^9 * 12) / (0.7 * 8) + 2.56 * 10^8 * 5 = 4.48* 10^9

Thus, the Total Execution Time = Total Number of Cycles / Clock Rate = 4.48* 10^9 / 2GHz = 2.24s

Compare to the single processor, it is speedup by 9.6s / 2.24s = 4.286 times.

b. If the CPI of the arithmetic instructions were doubled, what would the impact be on the execution time of the program on 1, 2, 4, or 8 processors?

n = 1:

Total Number of Cycles = 2.56*10^9 * 2 + 1.28 * 10^9 * 12 + 2.56 * 10^8 * 5 = 2.176 * 10^10

Total Execution Time = Total Number of Cycles / Clock Rate = 2.176 * 10^10 / 2GHz = 10.88s

n = 2:

The number of Cycles on single processor = (2.56*10^9 * 2 + 1.28 * 10^9 * 12) / (0.7 * 2) + 2.56 * 10^8 * 5 = 1.591 * 10^10

Thus, the Total Execution Time = Total Number of Cycles / Clock Rate = 1.591* 10^10 / 2GHz = 7.955s

n = 4:

The number of Cycles on single processor = (2.56*10^9 * 2 + 1.28 * 10^9 * 12) / (0.7 * 4) + 2.56 * 10^8 * 5 = 8.594* 10^9

Thus, the Total Execution Time = Total Number of Cycles / Clock Rate = 8.594 * 10^9 / 2GHz = 4.297s

n = 8:

The number of Cycles on single processor = (2.56*10^9 * 2 + 1.28 * 10^9 * 12) / (0.7 * 8) + 2.56 * 10^8 * 5 = 4.937* 10^9

Thus, the Total Execution Time = Total Number of Cycles / Clock Rate = 4.937 * 10^9 / 2GHz = 2.469s

We can notice that the execution time on each processor is increased because the CPI of arithmetic instructions is doubled.

c. 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?

Suppose the new CPI of load/ store instructions for single processor is "x".

n = 1:

Total Number of Cycles = 2.56*10^9 * 1 + 1.28 * 10^9 * x + 2.56 * 10^8 * 5 = 1.28 * 10^9 * (x + 3)

n = 4: (Original CPI values)

The number of Cycles on single processor = (2.56*10^9 * 1 + 1.28 * 10^9 * 12) / (0.7 * 4) + 2.56 * 10^8 * 5 = 7.68* 10^9

Thus we have the equation

1.28 * 10^9 * (x + 3) = 7.68* 10^9

and we can calculate the x = 3.

3. Consider the three different processors P1, P2, and P3 executing the same instruction set. P1 has a clock cycle time of 0.33 ns and CPI of 1.5; P2 has a clock cycle time of 0.40 ns and CPI of 1.0; P3 has a clock cycle time of 0.3 ns and CPI of 2.8.

a. Which has the highest clock rate? What is it?

Clock Rate = 1 / Clock Cycle time

P1: 1 / 0.33ns = 3.03GHz

P2: 1 / 0.4ns = 2.5GHz

P3: 1 / 0.3ns =3.333 GHz

So Processor 3 has the highest clock rate that is 3.333 GHz.

b. Which is the fastest computer? If the answer is different than above, explain why. Which is slowest?

To compare the speed of each processor, we can calculate the execution time of a single instruction.

P1: 1.5 * 0.33ns = 0.495ns

P2: 1 * 0.4ns = 0.4 ns

P3: 2.8 * 0.3ns = 0.84ns

So the Processor 2 is fastest and the Processor 3 is slowest.

Although Processor 3 has the highest clock rate, its "CPI" (cycles per instruction) is too high so it becomes the slowest one.

c. How do the answers for a and b reflect the importance of benchmarks?

When we compare the actual speed of a processor, we need to compare the execution time of single instruction, which is equal to "CPI * Clock cycle time", rather than only compare the CPI or the Clock Rate.

4. 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.

a. How much energy do you save if you execute at the current speed and turn off the system when the computation is complete?

0 energy will be saved. Because the formula of energy is E = Capacity * V^2 and there doesn't exist any term that related to the execution time or speed.

b. How much energy do you save if you set the voltage and frequency to be half as much?
The Energy of processor = Capacity load * Vlotage^2

Thus, if we set the voltage and frequency half of the original value, the new energy of the processor is 0.5 * 0.5 = 0.25 times of original energy. Thus, we can save (1 - 0.25) = 75% energy.

5. 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.

a. 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)?

Number of class A instruction: 10% * 10^6 = 1 * 10^5

Number of class B instruction: 20% * 10^6 = 2 * 10^5

Number of class C instruction: 50% * 10^6 = 5 * 10^5

Number of class D instruction: 20% * 10^6 = 2 * 10^5

Total Clock Cycles:

P1:

10^5 * 1 + 2 * 10^5 * 2+ 5 * 10^5 * 2+ 2 * 10^5 * 1= 1.7 * 10^6

P2:

10^5 * 2 + 2 * 10^5 * 3+ 5 * 10^5 * 4 + 2 * 10^5 * 4 = 3.6 * 10^6

Total execution time = Total Clock Cycles / Clock Rate

P1:

1.7 * 10^6 / (2 * 10^9) = 850 us

P2:

3.6 * 10^6 / (4 * 10^9) = 900 us

Thus, P1 is faster.

b. What is the global CPI for each implementation?

Global CPI

P1: 10% * 1 + 20% * 2 + 50% * 2 + 20% * 1 = 1.7

P2: 10% * 2 + 20% * 3 + 50% * 4 + 20% * 4 = 3.6

c. Find the clock cycles required in both cases.

Total Clock Cycles:

P1:

10^5 * 1 + 2 * 10^5 * 2+ 5 * 10^5 * 2+ 2 * 10^5 * 1= 1.7 * 10^6

P2:

10^5 * 2 + 2 * 10^5 * 3+ 5 * 10^5 * 4 + 2 * 10^5 * 4 = 3.6 * 10^6

d. Which processor has the highest throughput performance (instructions per second) ?

Instructions per seconds = 1 / Single Instruction execution time = Clock Rate / CPI

P1: 2GHz / 1.7 = 1.176 * 10^9

P2: 4GHz / 3.6 = 1.111 * 10^9

e. Which processor do you think is more energy efficient? Why?

I think Processor 1 is more energy efficient. Because its clock rate is lower than processor 2 which means during a same amout of time, processor 1 executes less clock cycles. Meanwhile, P1 also has lower CPI and better throughput performance.

6. **a.** What is the difference between CISC and RISC architectures? Give some examples wherein you think CISC architectures are better suited for than RISC architectures and vice versa.

The differecens between CISC and RISC architectures are:
(1) CISC has a large number types of instructions while RISC has a limited number types of instructions.
(2) For CISC arithmetic and logical operations can access the register and the memory, while RISC arithmetic and logical operations only access registers
(3) RISC allows fewer CPI than CISC.
(4) The instructions in CISC have multiple addressing formats and variable length. However, RISC's addressing formats is simple and the length of instructions is fixed.

Since RISC has less types of instructions, it is more friendly for someone to start to learn.
If we want to write more compact and more efficient program, probably we need to use CISC.

b. Describe in your own words why you think RISC V would be a better alternative compared to ARM or x86 architectures?

(1) Because RISC is a open-source ISA which means the developer and client will be worried about the licensing fee and potential commercial conflict.
(2) The flexibility and reconfigurabilty makes RISC V stand out of other competitors. Developers and companies could easily make customized processors using minimal ISA and different kinds of extensions.

c. What do you think are the challenges faced by RISC V architecture going forward?

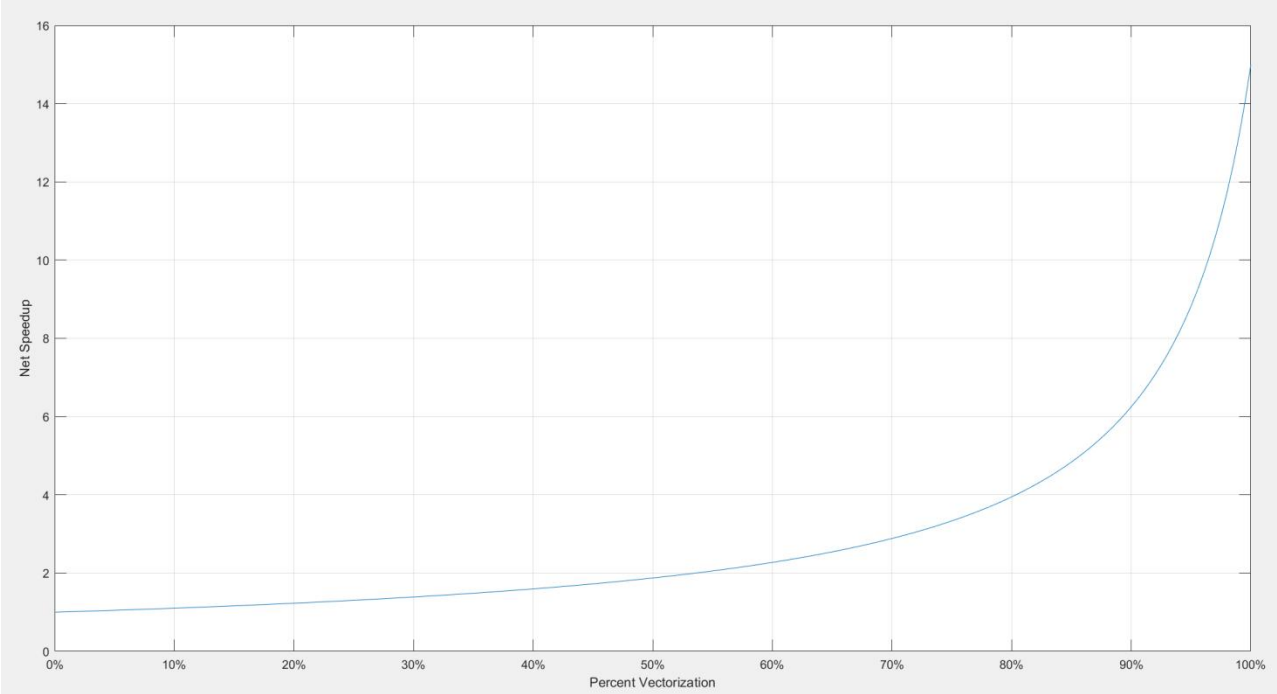
(1) As an opposite side of flexibility and highlt-customized design, if there is any problem occured in the produt or the programming development, it's hard to get help from others because they may have no idea with your personal style of work. Thus, it is better to propose an appropriate scope of standard for the long-term development of RISC V.

(2) Since RISC V has fewer types of instructions than CISC, the efficiency of the program will be significantly affected by the compiler. If compiler working with RISC V can become much more efficient, then the RISC V will definitely become much more popular.

(3) Currently, there are not too many hardware support the use of RISC V. Sometimes the customers prefer the more widely used ISAs.

7. 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."
Assume x% of time was spent in vectorization.
The old execution time is T. After vectorization, the execution time is 1 / 15 * T.
The new total execution time is: 1 / 15 * T * x% + (1 - x%) * T
Thus, the speed up is T / (1 / 15 * T * x% + (1 - x%) * T).
The plot is attached below:



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

When speedup is 2, which is:
T / (1 / 15 * T * x% + (1 - x%) * T) = 2

We can calculate that x = 53.571.
Thus 53.571% of vectorization could achieve a speedup of 2.

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

From last question we know that if we reach a speeup of 2, 53.571% of work is done with vectorization. Thus the percentage of its computation time is:
(1 / 15 * T * 53.571%) / (1 / 15 * T * 53.571% + (1 - 53.571%) * T) = 0.07143 = 7.143%

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

The maximum speed up is 15 times.
When speedup is half of the maximum speedup, which is 7.5:
T / (1 / 15 * T * x% + (1 - x%) * T) = 7.5

We can know that x = 92.857.
Thus to achieve half of the maximum speed up, we need 92.857% of vectorization.

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×)?

When the vectorization is 70%, the speedup is 2.8846. To reach an additional 2x speedup which is 5.7692 times speedup.

T / (1 / 15 * T * x% + (1 - x%) * T) = 5.7692
We can have x = 88.571. Thus 88.571% vectorization could achieve an additional 2x speedup.

8. 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?
The FIT of a single computer = 1 / MTTF = 1 / 35 per day = 0.0286 per day
Failure rate of the 10,000 computers cluster is 1 / 35 * 10000 = 285.714 computers perday
MTTF of System = 10000 * 1 / 3 / 285.715 = 11.667 days

b. If it costs an extra $1000, per computer, to double the MTTF, would this be a good business decision? Show your work.
If the MTTF per computer is doubled,
The FIT of a single computer = 1 / MTTF = 1 / 70 per day = 0.0143 per day
Failure rate of the 10,000 computers cluster is 1 / 70 * 10000 = 142.857 computers perday
MTTF of System = 10000 * 1 / 3 / 142.857 = 23.333 days

We can notice that the MTTF of the System is doubled too. I think it is a worthy improvement because it can make the whole system more reliable and decrease the loss of system failure. It will also improve the experience of developers and customers which is beneficial for the long term goal.

9. a. A program (or a program task) takes 150 million instructions to execute on a processor running at 2.7 GHz. Suppose that 70% of the instructions execute in 3 clock cycles, 20% execute in 4 clock cycles, and 10% execute in 5 clock cycles. What is the execution time for the program or task?

Total Number of clock cycles = 1.5 * 10^8 * 70% * 3 + 1.5 * 10^8 * 20% * 4 + 1.5 * 10^8 * 10% * 5 = 5.1 * 10^8

Execution Time = Total Number of clock cycles / Clock Rate = 5.1 * 10^8 / 2.7 GHz = 0.1889s

b. 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 1.5 GHz. What is the overall percentage improvement?

Total Number of clock cycles = 1.5 * 10^8 * 70% * 3 + 1.5 * 10^8 * 30% * 2 = 4.05 * 10^8

Execution Time = Total Number of clock cycles / Clock Rate = 4.05 * 10^8 / 1.5 GHz = 0.27s

0.1889 / 0.27 = 0.7.

Thus, the redesigned processor is 30% slower than the original processor.

10. Availability is the most important consideration for designing servers, followed closely by scalability and throughput.

a. We have a single processor with a failure in time (FIT) of 100. What is the mean time to failure (MTTF) for this system?

MTTF = 1 / FIT = 1 / (100 / 10^9) = 10^7 hours

b. If it takes one day to get the system running again, what is the availability of the system?

MTTR = 24 hours

MTBF = MTTR + MTTF = 10000024 hours

Availability = MTTF / MTBF = 10000000 / 10000024 = 99.9997%

c. Imagine that the government, to cut costs, is going to build a supercomputer out of inexpensive computers rather than expensive, reliable computers. What is the MTTF for a system with 1000 processors? Assume that if one fails, they all fail.

Failure rate of the supercomputer = sum of failure rate of each processor =

FIT = 100 / 10^9 = 10^(-7)

FIT of the system = FIT * 1000 = 10^(-4)

Thus, the MTTF of the system = 1 / FIT of the system = 10^4 hours

11. 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?
By turning off 60% of the servers, we now only have 40% servers working. Thus we save 60% power.

b. How much power savings would be achieved by placing 60% of the servers in the "barely alive" state?
Assume there are total N servers and maximum power of each server is P.

The original total power consumption is N* 0.9P = 0.9 PN.
After placing 60% of servers to "barely alive" state, the total power is:
0.4N * 0.9P + 0.6N * 0.2P = 0.48PN

Thus, the power saving is (0.9 - 0.48) / 0.9 = 0.4667 = 46.67%

c. How much power savings would be achieved by reducing the voltage by 20% and frequency by 40%?
Power = Capacity load * Frequency * Vlotage^2

New Power = Capacity load * (1 - 40%) * Frequency * ((1 - 20%) * Vlotage)^2 = Capacity load * 0.6 Frequency * (0.8 Vlotage)^2 = 0.384 * Original Power.

Thus, it will save (1 - 0.384) = 0.616 = 61.6% power.

d. How much power savings would be achieved by placing 30% of the servers in the "barely alive" state and 30% off?
New Power = 0.4N * 0.9 P + 0.3N * 0.2 P = 0.42PN
Thus, the power saving is (0.9 - 0.42) / 0.9 = 0.5333 = 53.33%

12. 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: 100 million arithmetic instructions, 20 million load/store instructions, 20 million branch instructions.

a. 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?
   Assume the original clock cycle time is 1.
   The original execution time = (1 * 100M + 10 * 20M + 3 * 20M) * 1 = 360M
   The updated execution time = (1 * 100M * (1 - 25%) + 10 * 20M + 3 * 20M) * 1.1 = 368.5M

   Thus it is a bad design since it will increase the execution time.

b. 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?
   (1) If the performance of arithmetice instructions is doubled:
   The new execution time = (1 * 100M * 0.5 + 10 * 20M + 3 * 20M) * 1 = 310M
   The speed up is:
   (Old execution time - New execution time) / Old execution time = (360M - 310M) / 360M = 13.89%

   (2) If the performance of arithmetice instructions is 10 times faster:
   The new execution time = (1 * 100M * 0.1 + 10 * 20M + 3 * 20M) * 1 = 270M
   The speed up is:
   (Old execution time - New execution time) / Old execution time = (360M - 270M) / 360M = 25%