1. The nor & nand instructions are not part of the RISC-V instruction set because the same functionality can be implemented using existing instructions. Write RISCV code that performs a nor operation on registers x8 and x9 and places the result in register x10. Write RISCV code that performs a nand operation on registers x5 and x6 and places the result in register x7

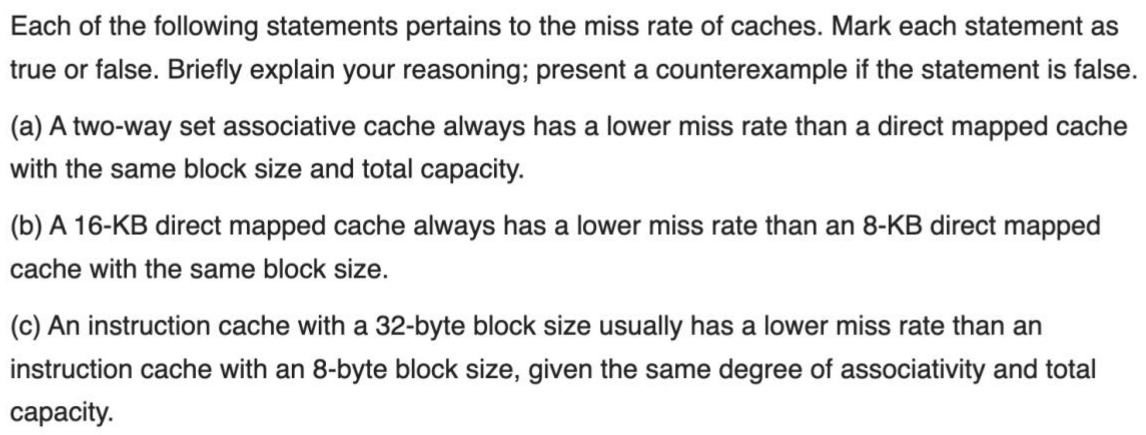
riscv isa dont have any NOR instruction , to acheive the NOR operation OR followed by NOT operation is done
NOR = NOT(OR (A, B))

```
or x9 , x7 , x8 ; //or operation between x7 and x8, results are stored in x9
not x9, x9 ; //1s complement of x9 will get stored in x9
```

hence the register x9 holds the NOR result between the x7 , x8
riscv isa dont have any NAND instruction , to acheive the NAND operation AND followed by NOT operation is done
NAND = NOT(AND (A, B))

```
and x6 , x4 , x5; // and operation between x4 and x5 results are stored in x6
not x6, x6 ; //1s complement of x6 will get stored in x6
```

hence the register x6 holds the NAND result between the x4 , x5



Each of the following statements pertains to the miss rate of caches. Mark each statement as true or false. Briefly explain your reasoning; present a counterexample if the statement is false.

(a) A two-way set associative cache always has a lower miss rate than a direct mapped cache with the same block size and total capacity.

(b) A 16-KB direct mapped cache always has a lower miss rate than an 8-KB direct mapped cache with the same block size.

(c) An instruction cache with a 32-byte block size usually has a lower miss rate than an instruction cache with an 8-byte block size, given the same degree of associativity and total capacity.

Step-by-step solution

Step 1 of 4

Cache is the high speed storage mechanism. Cache can be in the form of reserved part of main memory or may be the independent storage device.

The set associative cache scheme is the hybrid scheme between the two caches that are fully associative cache and direct mapped cache.

The direct mapped cache is defined as the cache location where location is determined from the middle address bits.

Comment

Step 2 of 4

• If each entry in main memory has the option to go to any one of N places in the cache, then it is referred to N-way set associative caching. For a 2 way set associative caching, the entry can go to any of the 2 places in the cache.

• True, the two ways set associative cache has lower miss rate always than the same block size direct mapped cache. Because the two- way set associative caching holds two addresses at the same set index, thus reducing conflicts which results in lower miss rate.

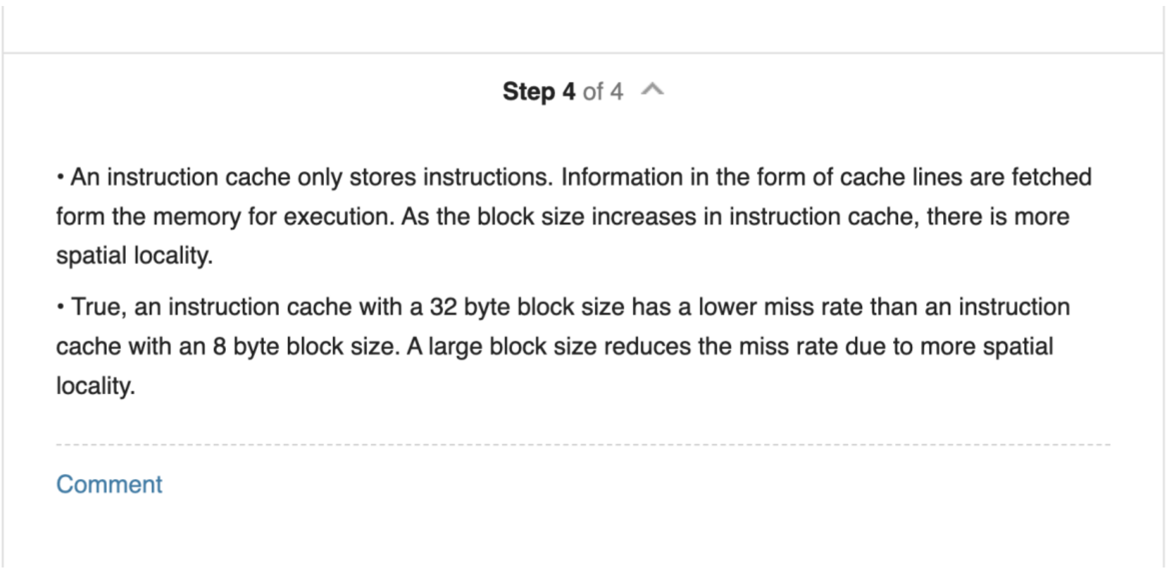
Comment

Step 3 of 4

• It is defined as the cache location where location is determined from the middle address bits. In the direct mapped cache, each entry in main memory has the option to go to only one place.

• True, 16 kb direct mapped cache with same block size has the lower miss rate than the 8 kb direct mapped cache. Because the capacity of a cache affects the miss rate, the more the capacity, the less is the miss rate.

Comment



Step 4 of 4

• An instruction cache only stores instructions. Information in the form of cache lines are fetched form the memory for execution. As the block size increases in instruction cache, there is more spatial locality.

• True, an instruction cache with a 32 byte block size has a lower miss rate than an instruction cache with an 8 byte block size. A large block size reduces the miss rate due to more spatial locality.

Comment

2. Describe the trade-offs of increasing each of the following cache parameters while keeping the others the same: [10 pts]

   (a) block size

   (b) associativity

   (c) cache size

Block size:-

Increasing block size will also increase cache's ability to take benefit of spatial locality. This will decrease the miss rate for applications with spatial locality. It also decrease the number of locations to map an address possibly increasing conflict misses. ~~Aslo~~ Also, the miss penalty i.e, the amount of time it takes to fetch the cache block from memory increases

Associativity:-

Increasing the associativity increases the amount of necessary hardware but in maximum cases decreases the miss rate. Associativities above 8 usually display only incremental decreases in miss rate

Cache size:-

Increases the cache size will decrease capacity misses & could decrease conflict misses. It could also increase access time

1.Students M,O&P are building custom hardware and compilers for their project
The following observations are known.CPI=1
Student O has built his design and it has a known execution time for a new program of 600 seconds.
Student M proposes a single cycle data path,and plans to use a compiler that will generate 300 Billion dynamic instructions per execution.
Student P proposes a pipelined data path,with a clock period of 1.2 ns.His compiler will generate 400 Billion instructions.Of these 400 Billion instructions,25%are loads,and 20%are branches.40%of all loads cause a 2-cycle load-use stall.The penalty for a mis-predicted branch is 5 cycles.The processor has
full bypassing,and does not suffer from any other stalls.
(i)How fast does M need to make his clock cycle in order to make the program run faster than O's?
(ii)How accurate does P's branch predictor need to be in order to make the program run faster than B's?

(i)Student O execution time = 600 seconds
Student M
Number of istructions = 300 * 10^9 instructions
Let the cycle time be C
300 * 10^9 * c < 600
C < 2GHz

(ii)clock period = 1.2 ns
Number of instructions = 400 * 10^9
25%are loads
20%are branches.
40% of all load cause 2-cycle stall
Branch miss predition penalty = 5
Let the miss prediction rate be p

Average CPI
1 + 0.25 * 0.4 * 2 + 0.2*p*5
1.2 + p
Total number of clocks
(1.2 + p)*400 * 10^9

Execution time = (1.2 + p)*(400 * 10^9) *1.2 * 10^-9
=> (1.2 + p) * 480 < 600
=> 1.2 + p < 1.25
=> p < (1.25 - 1.2)
=> p < 0.05
miss rate less than 5%

4. Assume that you have a computer with 1 clock cycle per instruction (CPI=1) when all accesses to memory are in cache. The only accesses to data come from load and store instructions. Those accesses account for 25 % of the total number of instructions. Miss penalty is 50 clock cycles and miss rate is 5 %. Determine the speedup obtained when there is no cache miss compared to the case when there are cache misses

⇒ lets look at the formulas first :-

CPU execution time = (CPU clock cycles + Memory stall cycles) × clock cycle time.

where :- CPU clock cycles = IC × CPI

IC = Instruction count

CPI - Cycle per instruction

Memory stall cycles :- It includes the time to handle a cache miss.

$= IC \times \frac{\text{Memory access}}{\text{Instruction}} \times$ Miss Rate × Miss Penalty.

Now, Given , CPI = 1

Miss Penalty = 50

Miss rate = 5%

If all accesses are cache hits :-

CPU execution time =

(CPU clock cycles + Memory stall cycles) × Clock cycle time

⇒ (IC × CPI + 0) × clock cycle

⇒ IC × clock cycle.

[Memory stall cycles = 0 because there is no cache miss)

• with real cache: (i.e with cache miss being present)

Memory Stall cycles = $IC \times \frac{\text{Memory access}}{\text{Instruction}} \times \text{Miss rate} \times \text{Miss penalty}$

$= IC \times (1+0.25) \times 0.05 \times 50$

$= IC \times 1.25 \times 0.05 \times 50$

$= 3.125 \times IC$

where $(1+0.25)$ represents 1 instruction access and 0.25 data ~~per ins~~ per instruction.

∴ CPU execution time(cache) =

~~$(IC \times 1.0 + IC \times 3.125)$~~

$= (IC \times 1.0 + IC \times 3.125) \times$ clock cycle

$= 4.125 \times IC \times$ clock cycle.

∴ Performance ratio $= \frac{\text{CPU execution time cache}}{\text{CPU execution time}}$

$= \frac{4.125 \times IC \times \text{clock cycle}}{IC \times \text{clock cycle}}$

$= 4.125$

So, computer with no cache miss is 4.125 times faster.

b. Suppose that when Program A is run, the user CPU time is 3 seconds, the elapsed wallclock time is 4 seconds, and the system performance is 10 MFLOP/sec. Assume that there are no other processes taking any significant amount of time, and the computer is either doing calculations in the CPU, or doing I/O, but it can't do both at the same time. We now replace the processor with one that runs six times faster, but doesn't affect the I/O speed. What will the user CPU time, the wallclock time, and the MFLOP/sec performance be now? **(10 Points)**

ANSWER:

CPU user time :

The CPU time is the measure of how much time a process is keeping the CPU busy or the amount of time taken by a process in running over the CPU but it does not include the I/O operations or delays, waiting time etc.

Wall Clock Time :

The wall clock time is the actual time on the clock that is taken by any processor to execute a process i.e. it includes delays, I/O operations, waiting times and any other requests that the process may have made during the running time of process from start till the end.

IT is always greater than CPU time.

Wallclock time = CPU user time + I/O time + delays

MFLOP/sec

This is a performance scenario in systems. It is known as Millions of floating point operations per second. It measure the number of floating point operations like addition, subtraction, division etc, that a system can perfom in one second. It does not include I/O.

Formula for MFLOP/sec is :

$$Mflop/sec = \frac{Number of floating points}{10^6 \times CPU user Time}$$

Solution to your question :

Given,

old CPU user time = 3 sec

old wallclock time = 4 sec

old MFLOPS/sec = 10 mflop/sec

New processor is 6 times faster

Performance and CPU time is inversely proportional i.e $Performance_x = \frac{1}{CPUtime_x}$

In question new processor speed is 6 times faster, so the let old performance be 1 the new performance will be 6.

So, $\frac{Performance_{new}}{Performance_{old}} = \frac{CPUtime_{old}}{CPUtime_{new}}$

$\Rightarrow \frac{6}{1} = \frac{3}{CPUtime_{new}}$

$\Rightarrow CPUtime_{new} = \frac{3}{6}$

$\Rightarrow CPUtime_{new} = 0.5sec$

old Wallclock time = CPU time + I/O time

4 = 3 + I/O time

So, **I/O time = 1 sec**

It is given in question that I/O time does not change

new Wallclock time = new CPUtime + I/O time

new Wallclock time= 0.5 + 1 = 1.5sec

$\Rightarrow Wallclocktime_{new} = 1.5sec$

MFlop formula : $Mflop/sec = \frac{Numberoffloatingpoints}{10^6 \times CPUuserTime}$

From old values :

$10 = \frac{Numberoffloatingpoints}{10^6 \times 3}$

$\Rightarrow Numberoffloatingpoints = 30 \times 10^6$

Now new MFLOP for 6 times faster processor :

$Mflop/sec = \frac{30 \times 10^6}{10^6 \times 1.5}$

$$Mflop/sec = \frac{300}{15} = 20$$

So, $Mflop/sec = 20mflop/sec$

So the answers when the processor speed is 6 times faster are :

$\mathbf{CPUtime_{new} = 0.5sec}$

$\mathbf{Wallclocktime_{new} = 1.5sec}$

$\mathbf{Mflop/sec = 20mflop/sec}$

Suppose that when Program A is run, the user CPU time is 3 seconds, the elapsed wallclock time is 4 seconds, and the system performance is 10 MFLOP/sec. Assume that there are no other processes taking any significant amount of time, and the computer is either doing calculations in the CPU, or doing I/O, but it can't do both at the same time. We now replace the processor with one that runs six times faster, but doesn't affect the I/O speed. What will the user CPU time, the wallclock time, and the MFLOP/sec performance be now?

in question new processor speed 6 time faster so the let old Performence be 1 the new peformence will be 6.

so $\frac{\text{Performance}_{new}}{\text{Performance}_{old}} = \frac{\text{CPU time}_{old}}{\text{CPU time}_{~~old~~ new}}$

$\Rightarrow \frac{6}{1} = \frac{3}{\text{CPU time}_{new}}$

$\Rightarrow \text{CPU time}_{new} = \frac{3}{6}$

$\Rightarrow$ CPU time new = 0.5 sec

Old Wall clock = CPU time + I/O time

4 = 3 + I/O time

so I/O time = 1sec

it is given in question that I/O time does not change.

new Wallclock time = new CPU time + I/O time.

new Wallclock time = 0.5 + 1 = 1.5 sec

Mflop formula -

$$\text{Mflop/sec} = \frac{\text{Number of floating points}}{10^6 \times \text{CPU User time}}$$

from old values

$$10 = \frac{\text{Number of floating points}}{10^6 \times 3}$$

$\Rightarrow$ Number of floating points $= 30 \times 10^6$

Now new MFLOP for 6 times faster processor -

$$\text{Mflop/sec} = \frac{30 \times 10^6}{10^6 \times 1.5}$$

$$\text{Mflop/sec} = \frac{300}{15} = 20$$

so Mflop/sec = 20 mflop/sec

so the answer when the processor speed is 6 times faster are -

CPU time new = 0.5 sec.

Wallclock time new = 1.5 sec.

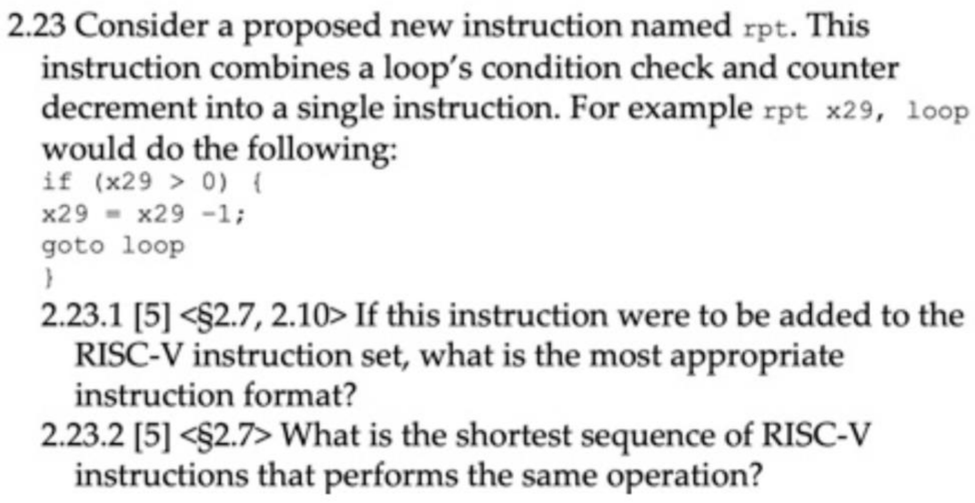
Mflop/sec = 20 mflop/sec.

Question 1: Three C's of Cache Misses (18 points)

Mark whether the following modifications to cache parameters will cause each of the categories to **increase, decrease**, or whether the modification will have **no effect**. You can assume the baseline cache is set associative. **Explain your reasoning** to receive credit.

Assume that in each case the other cache parameters (number of sets, number of ways, number of bytes/line) and the rest of the machine design remain the same.

	compulsory misses	**conflict misses**	**capacity misses**
increasing number of sets	no effect block size is constant	decreases more sets are available for data, so there is less of a chance for two ops to collide and evict one another	decreases capacity increases
increasing number of ways	no effect block size is constant	decreases there are more ways available for data to be placed into	decreases capacity increases
increasing number of bytes per line	decreases more data is brought in on a given cache miss	no effect associativity and number of sets is constant	decreases capacity increases

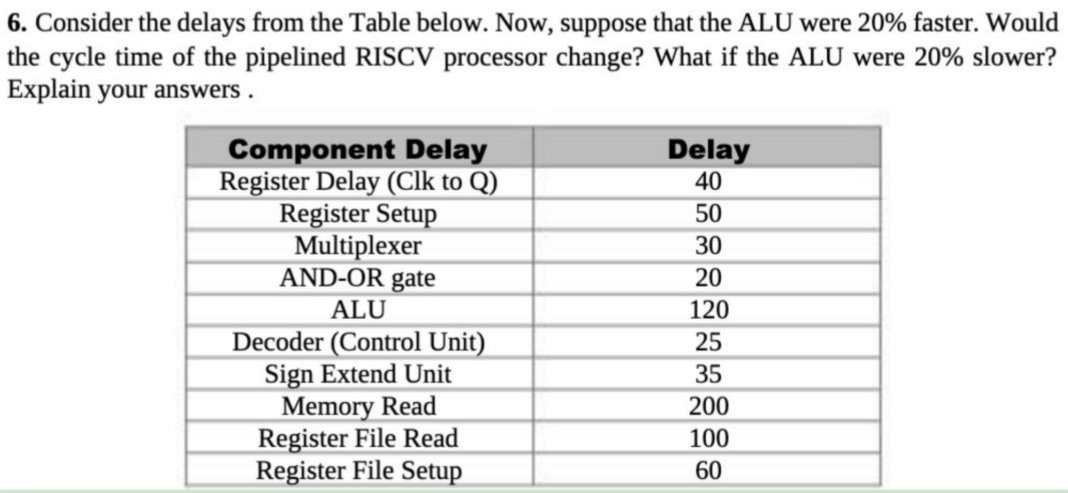

2.23 Consider a proposed new instruction named `rpt`. This instruction combines a loop's condition check and counter decrement into a single instruction. For example `rpt x29, loop` would do the following:

```
if (x29 > 0) {
x29 = x29 -1;
goto loop
}
```

2.23.1 [5] <§2.7, 2.10> If this instruction were to be added to the RISC-V instruction set, what is the most appropriate instruction format?

2.23.2 [5] <§2.7> What is the shortest sequence of RISC-V instructions that performs the same operation?

https://www.chegg.com/homework-help/questions-and-answers/223-consider-proposed-new-instruction-named-rpt-instruction-combines-loop-s-condition-chec-q44477783


6. Consider the delays from the Table below. Now, suppose that the ALU were 20% faster. Would the cycle time of the pipelined RISCV processor change? What if the ALU were 20% slower? Explain your answers .

Component Delay	Delay
Register Delay (Clk to Q)	40
Register Setup	50
Multiplexer	30
AND-OR gate	20
ALU	120
Decoder (Control Unit)	25
Sign Extend Unit	35
Memory Read	200
Register File Read	100
Register File Setup	60

Explanation:
In the pipelined RISC-V processor, the cycle time is determined by the maximum of all the delays offered by any of its stages. All the stages are made to operate with the same delay as the stage with maximum delay.
From the given tabulated data, the maximum delay is offered by the Memory read stage which is 200. Therefore, the cycle time is constrained by the Memory read stage and thus, the cycle time will be 200.

Now, even if the ALU is made 20% faster leading to its delay being 120(1-0.2) = 96, the cycle time will still be 200.

Similarly, if ALU is slowed by 20% leading to its delay being 120(1+0.2) = 144, the maximum delay is still by the Memory read operation which is 200.

Hence, in both the cases, the cycle time will still remain as the 200 even if ALU is speed up or slowed by 20%.

https://www.coursehero.com/qa/wait/47562231/?question_id=47562231

Question:
Assume that you have a computer with 1 clock cycle per instruction (CPI=1) when all accesses to memory are in cache. The only accesses to data come from load and store instructions. Those accesses account for 25 % of the total number of instructions. Miss penalty is 50 clock cycles and miss rate is 5 %. Determine the speedup obtained when there is no cache miss compared to the case when there are cache misses

Answer

Solution:-

let's look at the formalitieslist :-

CPU execution time = (CPU clock cycles + memory stall cycles) x Clock cycle time

where :- CPU clock cycles = IC x CPI

IC = Instrulation count

CPI = cycle per Instruction

memory stall cycles :- It includes time to handle a cache miss

$$= IC \times \frac{\text{memory access}}{\text{Instruction}} \times \text{miss rate} \times \text{miss penalty}$$

Now Given , CPI = 1

miss penalty = 50

miss rate = 5%

If all access are cache hits:-

CPU exicition time =

(CPU clock cycles + memory stall cycles) x clock cycle time

= (IC x CPI + 0) x clock cycle

= IC x clock cycle

[memory stall cycles = 0, because there is no chache miss)

withread chache : (i.e with cache being present)

memory stall cycles = $IC \times \frac{\text{memory access}}{\text{Instruction}} \times$ miss rate $\times$ miss penalty

$= IC \times (1+0.25) \times 0.05 \times 50$

$= IC \times 1.25 \times 0.05 \times 50$

$= 3.125 \times IC$

where (1+0.25) represents 1 instruction access and 0.25 data per Instruction.

∴ CPU execution time =

$= (IC \times 1.0 + IC \times 3.125) \times$ clock cycle

$= 4.125 \times IC \times$ clock cycle

∴ Performance ratio $= \frac{\text{CPU execution time cache}}{\text{CPU execution time}}$

$= \frac{4.125 \times IC \times \text{clock cycle}}{IC \times \text{clock cycle}}$

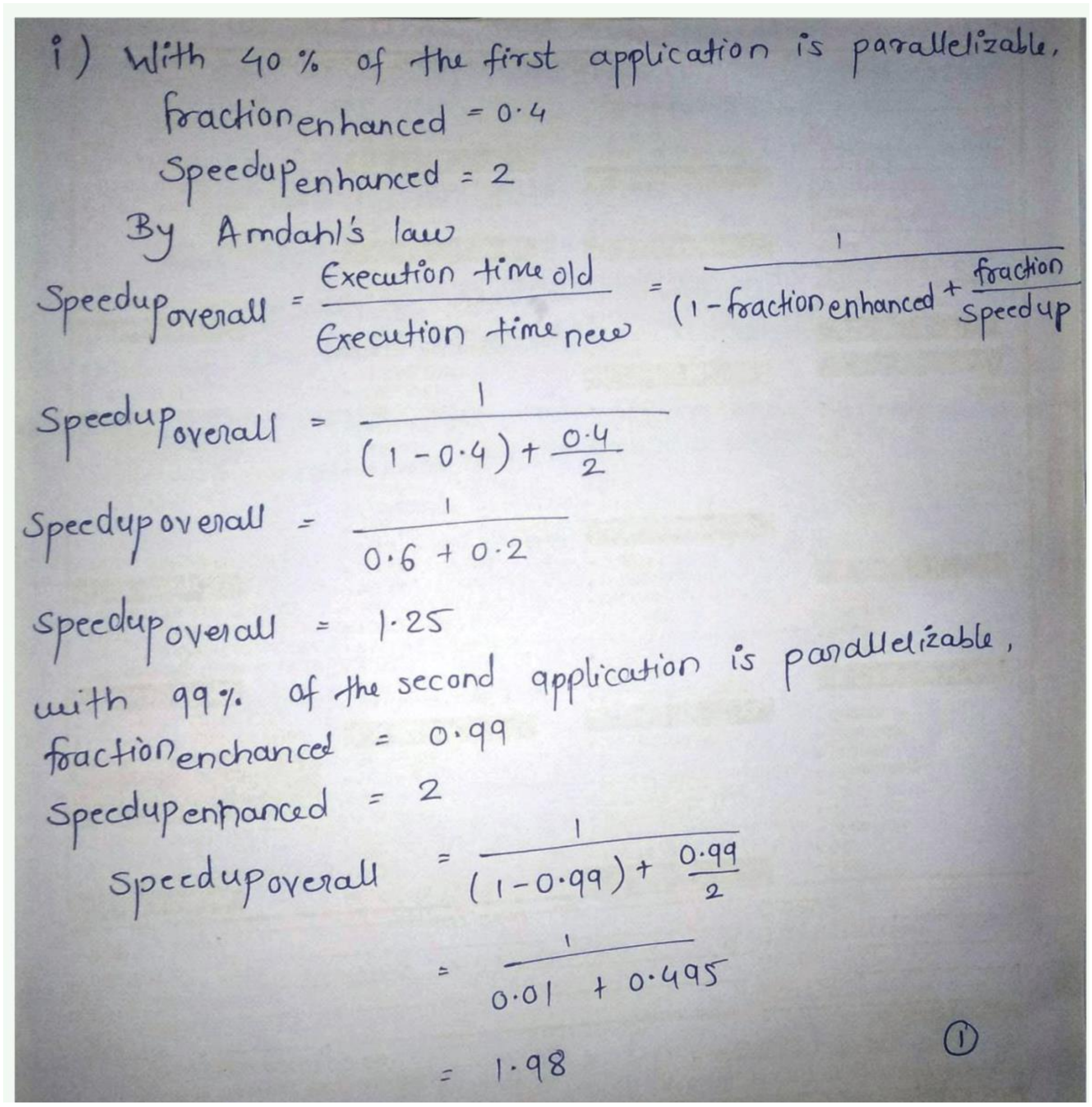
$= 4.125$

so, computer with no cache miss is 4.125 times faster.

3. Hypersonic Computers, your employer, has just bought a new dual Core processor, and you have been tasked with optimizing your software for this processor - for two applications on this dual Core processor, but the resource requirements are not equal. The first one needs 80% of the resources, and the second only 20%.

(i) Assume that 40% of the first application is parallelizable, how much speedup would you achieve with that application if run in isolation? Assume that 99% of the second application is parallelizable, how much speedup would this application observe if run in isolation?

(iii) Given that 40% of the first application is parallelizable, how much overall system speedup would you observe if you parallelized it? Given that 99% of the second application is parallelizable, how much overall system speedup would you observe if you parallelized it?



i) With 40 % of the first application is parallelizable,

$\text{fraction}_{\text{enhanced}} = 0.4$

$\text{Speedup}_{\text{enhanced}} = 2$

By Amdahl's law

$$\text{Speedup}_{\text{overall}} = \frac{\text{Execution time}_{\text{old}}}{\text{Execution time}_{\text{new}}} = \frac{1}{(1-\text{fraction}_{\text{enhanced}}+\frac{\text{fraction}}{\text{Speedup}}}$$

$$\text{Speedup}_{\text{overall}} = \frac{1}{(1-0.4)+\frac{0.4}{2}}$$

$$\text{Speedup}_{\text{overall}} = \frac{1}{0.6+0.2}$$

$$\text{Speedup}_{\text{overall}} = 1.25$$

with 99% of the second application is parallelizable,

$\text{fraction}_{\text{enchanced}} = 0.99$

$\text{Speedup}_{\text{enhanced}} = 2$

$$\text{Speedup}_{\text{overall}} = \frac{1}{(1-0.99)+\frac{0.99}{2}}$$

$$= \frac{1}{0.01+0.495}$$

$$= 1.98$$

①

iii) Here we assume that each application completely "owns" the whole system that is both processor cores during the time it is running. that is, there is no overlapping of the two applications in time - sharing the two processor crores.

Now. the resuorces needed by first = 0.8

the resources needed by second = 0.2

if 40% of the first application is parallelizable

$$\text{Speedup}_{\text{overall}} = \frac{1}{\text{Resources needed first} + \text{Resources needed Second} \times \left[\left(1 - \text{fraction}_{\text{chanced}} \right) + \frac{\text{fraction}_{\text{enhanced}}}{\text{Speedup}_{\text{enhanced}}} \right]}$$

$$= \frac{1}{0.2 + 0.8 \times \left[(1 - 0.4) + \frac{0.4}{2} \right]}$$

$$= \frac{1}{0.2 + 0.8 \times [0.6 + 0.2]}$$

$$= 1.19$$

if 99% of the second application is parallelizable

$$\text{speedup}_{\text{overall}} = \frac{1}{0.8 + 0.2 \times \left[(1 - 0.99) + \frac{0.99}{2} \right]}$$

②

$$= \frac{1}{0.8 + 0.2 \times [0.01 + 0.495]}$$

$$= 1.11$$