Student ID : 1912020106

Subject Code : CSE - 3117

Session : Spring - 2021

Date : 6.15.2021

Student Signature: Emad

Ans to the question no: 4 (a)

. System failure occur

- A programming mistake is a fault.
- The consequence of a fault is an error (or latent error) in the software.
- upon activation, the error become effective
- when this effective error produces erroneous data that affect the delivered service, a failure occurs.

Ans to the question no: 4 (b)

The Sum of the failure rates is - $\text{Failure Rate}_{\text{system}}$

$$= 10 * \frac{1}{1000000} + \frac{1}{500000} + \frac{1}{200000} + \frac{1}{200000} + \frac{1}{1000000} = \frac{10+2+5+5+1}{1000000 \text{ hours}}$$

$$= \frac{23}{1000000 \text{ hours}}$$

The MTTF for the System is just the inverse of the failure rate.

$$\text{MTTF}_{\text{system}} = \frac{1}{\text{Failure Rate}_{\text{system}}} = \frac{100000 \text{ hours}}{23}$$

$$= 43500 \text{ hours}$$

or just under 5 years

Ans to the question no: 4 (c)

Synchronous bus	Asynchronous bus
1) Includes a clock in the control lines and a fixed protocol for sending address and data relative to the clock	1) Self timed and handshaking protocols are used between sender and receiver
2) Little or no logic is needed to decide what to do next	2) Bus length is flexible
3) Fast and inexpensive	3) slower than synchronous bus

Ans to the question no: 4 (d)

There are five levels of program used to evaluate performance of the new machine. They are listed below in decreasing order of accuracy prediction:

1. Real Application:

Have input output and operations that an user can select when running the programm

Ex: compilers for c,

2. Modified (or Scripted) applications

Application are modified for two primary reasons

i) To enhance portability.

ii) To Focus one particular aspect of system performance.

Ex: To create a CPU-oriented bench mark, I/O may be removed or restructed to minimize its impact on execution time.

3. Kernels :

→ Several attemps have been made to extract small, key pieces from real

→ Ex : Livermore loops and Linpack

4. Toy benchmarks

→ Typically between 10 and 100 lines of code.

→ Produce a result the user already known before running the toy programm.

Ex : Puzzle.

Synthetic banchmarks :

→ Similar in philosophy to kernels.

→ Try to match the average frequency of operations and operands of a large set of programs

Ex : whetstone and Dhrystone

Ans to the question no: 1 (a)

Amdahl's Law gives us a quick way to find speedup from some enhancement which depends on two factors:

1. the fraction of computation time in the original computer that can be converted to take advantages of the enhancement-}

For example, if 20 Seconds of the execution time of a program that takes 60 seconds in total can use an enhancement the fraction is 20/60. This value, which we call Fraction enchanced is always less than or equal to 1.

2. The improvement gained by the enhanced execution mode; that is how much faster the task would run if the enhanced mode were used for the entire program - This value is the time of the original mode over the time of the enhanced mode. If the time of the enhanced mode takes 2 seconds for some enhanced mode takes ~~f~~ of the program, while it look portion 5 seconds in the original mode, the improvement is 5/2. we will call this value. which is always greater than 1, $\text{Speedup}_{\text{enhanced}}$

Ans to the question no: 1 (b)

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

$\text{Speedup}_{\text{enhanced}} = 10$

So, $\text{Speedup}_{\text{overall}} = \dfrac{1}{(1-0.4) + \frac{0.4}{10}}$

$= \dfrac{1}{0.6 + \frac{0.4}{10}} = \dfrac{1}{0.64} \cong 1.56$

Ans

Ans to the question no: 1 (c)

- Clock cycle time - Hardware technology and organization

- CPI - Organization and instruction set architecture.

- Instruction Count - Instruction set architecture and compiler technology

- Sometimes it is useful in designing the CPU to calculate the number of total CPU clock cycles as:

$$\text{CPU clock cycles} = \sum_{i=1}^{n} IC_i \times CPI_i$$

we show that 3 sector linked up with each other and each characteristic are inter.dependent, Ans to the Question no: 1 (d)

- Because you use it everyday
- Because you will likely use it for the rest of your life because you are a CS major, studying computer Science.

-> Enable better systems : make computer faster, cheaper, smaller, more reliable

-> By exploiting advances and changes in underlying technology / circuits

-> Enable new applications :

-> virtual reality ?

-> Life like 3D visualization 20 years ago ?

-> Understand why computers work the way they do .

→ Enable better Solutions to Problems:

→ Software innovation is built into trends and changes in computer architecture

→ 50% performance improvement per year has enabled this innovation

Ans to the Question no: 3 (a)

A RISC instruction set can be implemended pipelined fashion. Every instruction in the RISC subset can be implemented in at most 5 ~~m~~ clock cycles.

→ The 5 clock cycles are follows:

1. Instruction fetch cycle (IF):

Send the program counter (PC) to memory and fetch the current instruction from memory. update the PC to the next sequential PC by adding 4 (since each instruction 4 bytes) to the PC.

2. Instruction decode / register fetch cycle (ID):

Decode the instruction and read the registers corresponding to register source specifiers from the register file.

3. Execution / effective address cycle (Ex)

The ALU operates on the operands prepared in the prior cycle, performing one of three functions depending on the instruction type

- Memory reference
- Register - Register ALU instruction
- Register - immediate ALU instruction

4. Memory access: If the instruction is a load, the memory does a read using the effective address computed in the previous cycle. If it is a store, then the memory writes the data from the second register read from the register file using the effective address.

5. write-back cycle (WB): write the result into the register file, whether it comes from the memory system (for a load) or from the ALU instruction).

Ans to the question no: 3 (b)

The average instruction execution time on the unpipelined processor is

Average instruction execution time = clock cycle × Average CPI

$= 1ns \times [(40\% + 20\%) \times 4 + 40\% \times 5]$

$= 1ns \times 4.4$

$= 4.4\ ns$

In the pipelined implementation, the clock must run at the speed of the slowest stage plus overhead, which will be 1+0.2 or 1.2 ns: this is the average instruction execution time.

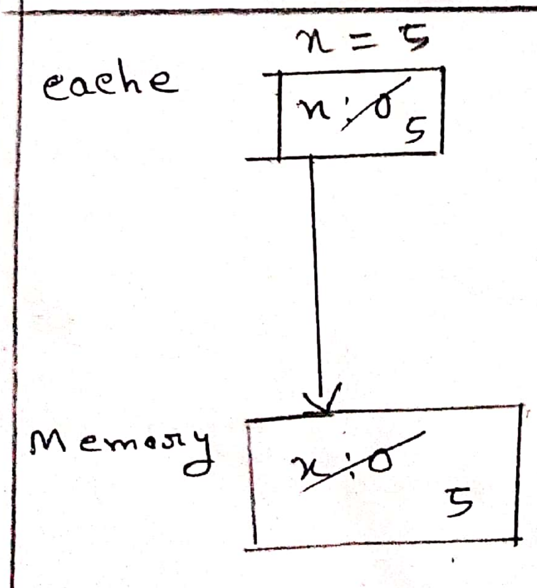
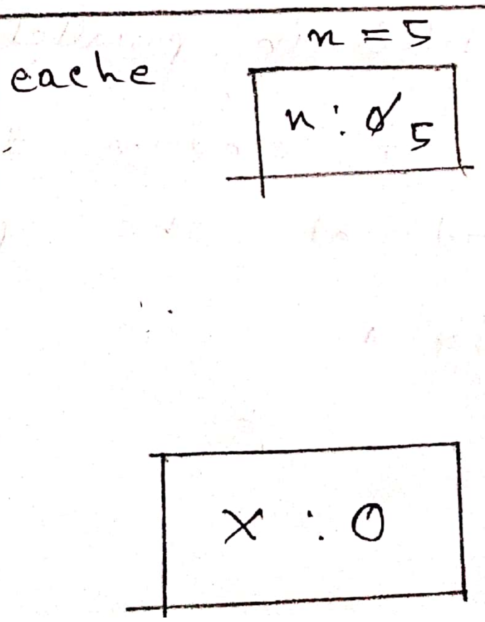
Thus, the speedup from pipelining is

$$\text{Speed up from pipelining} = \frac{\text{Average instruction time unpiplined}}{\text{Average instruction time pipelined}}$$

$$= \frac{4.4\ ns}{1.2\ ns} = 3.7 \text{ times}$$

Ans to the question no: 3 (c)

write - through	write back
1) In this method main memory is updated with every memory write operation as well as cache memory is updated in parallel if it contains the word at the specified address	1) In this method only cache location is updated during write operation

write through	write back
2) Main memory always contains same data as cache	2) Main memory and cache memory may have different data
3) Number of memory write operation in a typical program is more	3) Number of memory write operation in a typical program is less
4) It is a process of writing cache and main memory simultaneously	4) It is a process of writing cache and data is removed from cache first copied to main memory
 cache x = 5 x : ~~0~~ 5 Memory ~~x : 0~~ 5	 cache x = 5 x : ~~0~~ 5 X : 0