1. In computer languages, what is translation? (5%)

Computer programs are translated from one language to another by the translator, a programming language processor. Machine code is built from source code for an application.
The translator's primary function is to convert high-level language programs into CPU-friendly machine language programs. Additionally, it finds and reports translation problems.

Types of Translators:

There are three different types of translators as follows:

- **Compiler**

  A compiler, which is a translator, is used to convert programs written in high-level programming languages into low-level programming languages. In a single session, the entire program is translated, and any errors discovered after the conversion are noted. A Compiler is a processor-dependent and a platform-dependent.

- **Interpreter**

  Similar to the compiler, the interpreter is a translator that converts high-level programming languages into low-level programming languages. The difference is that it updates the program one line of code at a time and logs errors as they are discovered. An Interpreter is also more portable than compiler as it is processor-independent.
  EX: C,C++, JAVA etc. are compiled languages.

- **Assembler**

  A translator called an assembler is used to convert assembly language code into machine language code.

  It performs the same task as the assembly language compiler while acting more like an interpreter.
  EX : GNU, GAS are the examples of assemblers.

2. In computer languages, what is interpretation? (5%)

A programming language known as a "interpreted language" is one that is typically interpreted rather than being translated into machine instructions. It is

one in which a different program reads and executes the instructions rather than the target computer directly.
Rather than scanning the whole program and translating it into machine code like a compiler dose, the interpreter translates code one statement at a time.
When the first error is encountered, translations are stopped. Therefore, debugging is simple. Java, C++, and C are programming languages that utilize compilers, while Python, JavaScript, and Ruby are programming languages that employ interpreters.

3. **What is a virtual machine? (5%)**

An isolated computing environment known as a virtual machine is produced by removing resources from a physical machine. A virtual machine (VM) is a computational resource that runs apps and runs programs using software rather than a physical computer. A physical "Host" machine hosts one or more virtual "Guest" machines. Even when they are all running on the same host, each virtual machine has its own operating system and operates independently of the others.

Types of Virtual machines:
There are two types of VMs:

- **Process virtual machine**, which separate a single process, and system VMs, which offers a full separation of the operating system and applications from physical computer.
  Example: Java Virtual Machine, the .Net framework and Parrot Virtual Machine.

- **System Virtual Machine**, rely on hypervisors, as a go between giving software access to the hardware resources.
  Big names in the hypervisors space includes VMware, intel/Linux foundation (Xen), Oracle (Oracle VM server for x86), Microsoft (Hyper-V).

4. **What is a data path? (5%)**

Data path is a collection of functional units such as arithmetic logic units and multipliers that perform data processing operations, registers and buses. Along with the control unit it composes the central processing unit. A larger datapath can be made by joining more than one data paths using multipliers.

The data path is the ALU, the set of registers, the CPU's internal buses that allow data to flow between them.

5. **Distinguish bit, Byte, and word. (5%)**

The **Word** length depends upon the number of bits or characters in a word.
The **bit** varies from 4, 8, 12, 16, 32 etc., upto 64 i.e., the word maybe as large as 64 bit or as short as 8 bits.
A **byte** is usually shorter than a word, typically consisting of 8-bits.

6. What are the registers in CPU? (5%)

A CPU includes registers for temporary storage. While certain registers are user-accessible, others are used internally and cannot be accessed from outside the processor. Both kinds of registers are present in most contemporary CPU designs. There are many different types of registers, some of which are often used.
Below are the CPU registers:

- Accumulator (AC)
- Flag Register
- Address Register (AR)
- Data Register (DR)
- Program Counter (PC)
- Instruction Register (IR)
- Stack Control Register (SCR)
- Memory Buffer Register (MBR)
- Index Register (IR)

7. What are the design principles for modern computers? (5%)

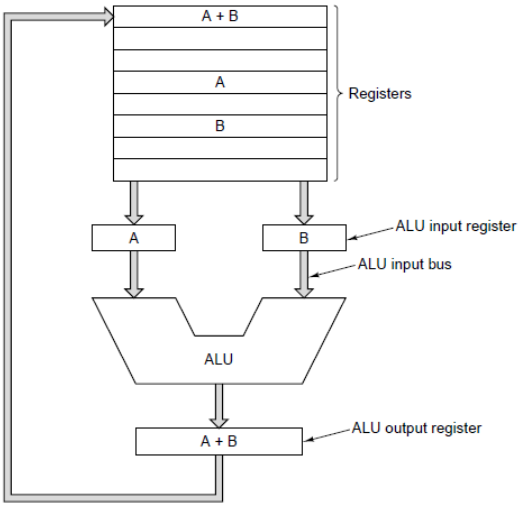
There is a set of design principles, sometimes called the RISC design principles, that architects of general-purpose CPUs do their best to follow:

- All Instructions Are Directly Executed by Hardware
  – eliminates a level of interpretation
- Maximise the Rate at Which Instructions are Issued
  – MIPS = millions of instructions per second
  – MIPS speed related to the number of instructions issued per second
  – Parallelism can play a role
- Instructions should be Easy to Decode
  – a critical limit on the rate of issue of instructions
  – make instructions regular, fixed length, with a small number of fields.
  – The fewer different formats for instructions.
- Only Loads and Stores Should Reference Memory

– Operands for most instructions should come from- and return to-registers.
– Access to memory can take a long time
– Thus, only LOAD and STORE instructions should reference memory.

- Provide Plenty of Registers
  – accessing memory is relatively slow, many registers (at least 32) need to be provided, so that once a word is fetched, it can be kept in a register until it is no longer needed.

8. **Consider the operation of a machine with the data path of the figure below. Suppose that loading the ALU input registers takes 5 nsec, running the ALU takes 10 nsec, storing the result back in the register scratchpad takes 5 nsec, and there is no pipeline. What is the maximum number of MIPS (Million Instructions Per Second) this machine is capable of? (5%)**



CSCI540 Homework Assignment - 1

Poojitha Chennuru - 50307728

8) The loading in the ALU input registers takes the 5nsec and runs the ALU and thus takes the 10nsec and thus stores the result back into the scratch pad registers and thus takes 5nsec. The data cycle in it is 20ns The total time is 20nsec for one cycle to calculate the MIPS divide one sec with 20nsecs

Millions of instructions per second (MIPS) = $\frac{1 \times 10^9}{20}$

$= 5 \times 10^7$

= 50,000,000nsec

Therefore the maximum number of mips this machine is capable of in the absence of pipelining = 50MIPS

9. In a single-core CPU that has a five-stage pipeline, it takes 1 CPU clock cycle to fetch the instruction, 1 CPU clock cycle to decode the instruction, 1 CPU clock cycle to fetch the operands from the registers, 2 CPU clock cycles to execute the instruction, and 1 CPU clock cycles to write the result back to the register. How many CPU clock cycle does it need to execute approximately 60,000 instructions? You do not need to consider any storage other than the registers, and you should assume the instructions are executed in their original order. (10%)

9) As per given information for pipelining:

Total clock cycles = $(n+k-1) * T$

where $n = 60,000$

$k = 5$

$T = 2$

Total clock cycles = $(60000+5-1) * 2$

$= (60004) * 2$

$= 12,0008$

10. **In a 16-bit little endian machine, how is the text "YA" stored in a word? How is the integer number 910 stored in a word? Write down the binary bits. (10%) (Hint: You need to use the ASCII table on Page 139.)**

    Y-01011001
    A-01000001
    Binary
    YA-0101100101000001
    Address
    Lower M has ----> 01000001 (A)
    Higher M+1 has ----->01011001 (Y)

    910 converts to hexadecimal which is 038E
    0000 0011 1000 1110 in binary
    Break in to two bytes so
    MOST SIGNIFICANT BYTE LOWEST SIGNIFICANT BYTE
    00000011 10001110

    Little endian format

    Address
    8E at Lower Address M HAS 10001110
    03 at Higher Address M+1 HAS 00000011

11. **Devise a 7-bit even-parity Hamming code for the numbers 0 to 4. (Hint: 0 is 0000 without parity bits.) (10%)**

Poojitha chennuru
50307728

11) Explanation:

Consider the 7 bit hamming code has bits to 7 as shown below

Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1

Bit1, Bit2, and Bit4 are the parity bits and all remaining bits are the data bits

d_7	d_6	d_5	P_4	d_3	P_2	P_1

Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1

Bit1, Bit2 and Bit4 are the parity bits and remaining bits are the databits

d_7	d_6	d_5	P_4	P_3	P_2	P_1

parity bit P_1 covers bit1, bit3, bit5, bit7
parity bit P_1 covers bit2, bit3, bit6, bit7
parity bit P_4 covers bit4, bit5, bit6, bit7

Number 0 is 0000 in binary

$d_3=0$, $d_5=0$, $d_6=0$, $d_7=0$

Since $d_3=0$, $d_5=0$, $d_7=0$, so $P_1=0$

Since $d_3=0$, $d_6=0$, $d_7=0$, so $P_2=0$

since $d_5=0$, $d_6=0$, $d_7=0$ so $P_4=0$

Hamming code is 0000111

Number 2 is 0010 in binary

$d_3=0$, $d_5=1$, $d_6=0$, $d_7=0$

Since $d_3=0$, $d_5=1$, $d_7=0$ so $P_1=1$

Since $d_3=0$, $d_6=0$, $d_7=0$ so $P_2=0$
since $d_5=1$, $d_6=0$, $d_7=0$ so $P_4=1$
Hamming code is 0011001
Number 3 is 0011 in binary
$d_3=1$, $d_5=1$, $d_6=0$, $d_7=0$
since $d_3=1$, $d_5=1$, $d_7=0$ so $P_1=0$
since $d_3=1$, $d_6=0$, $d_7=0$ so $P_2=1$
since $d_5=1$, $d_6=0$, $d_7=0$ so $P_4=1$
Hamming code is 0011110
Number 4 is 0100 in binary
$d_3=0$, $d_5=0$, $d_6=1$, $d_7=0$
Since $d_3=0$, $d_5=0$, $d_7=0$ so $P_1=0$
Since $d_3=0$, $d_6=1$, $d_7=0$ so $P_2=1$
Since $d_5=0$, $d_6=1$, $d_7=0$ so $P_4=1$
Hamming code is 0101010

Result:

Number	Binary	Hamming code
0	0000	0000000
1	0001	0000111
2	0010	0011001
3	0011	0011110
4	0100	0101010

12. A computer has a bus with 5-nsec cycle time, during which it can read or write a 32-bit word from memory. The computer has an Ultra4-SCSI disk that uses the bus and runs at 160 MBytes/sec. The CPU normally fetches and executes one 32-bit instruction every 1 nsec. How much does the disk slow down the CPU? Write down the percentage of CPU cycles that are spent on disk access. (10%)

12) consider the data:
cycle time = 5n sec
size of the instruction = 32 bit
speed of the bus = 160 Mbytes/sec
for single 32bit instruction fetches and executes in 1nsec
The computer is made up of an ultra4-scsi disk that runs on a bus at a speed of 160 Mbytes/sec, or it can be assumed to consist of 4 bytes per word.
Thus, the transfer rate of the disk is:

$$\text{Transfer\_rate} = \frac{\text{speed}}{\text{word-size}}$$

$$= \frac{160}{4}$$

$$= 40 \text{ million words/sec}$$

From the above transfer rate, 200 million bus cycles per second the disk rate is

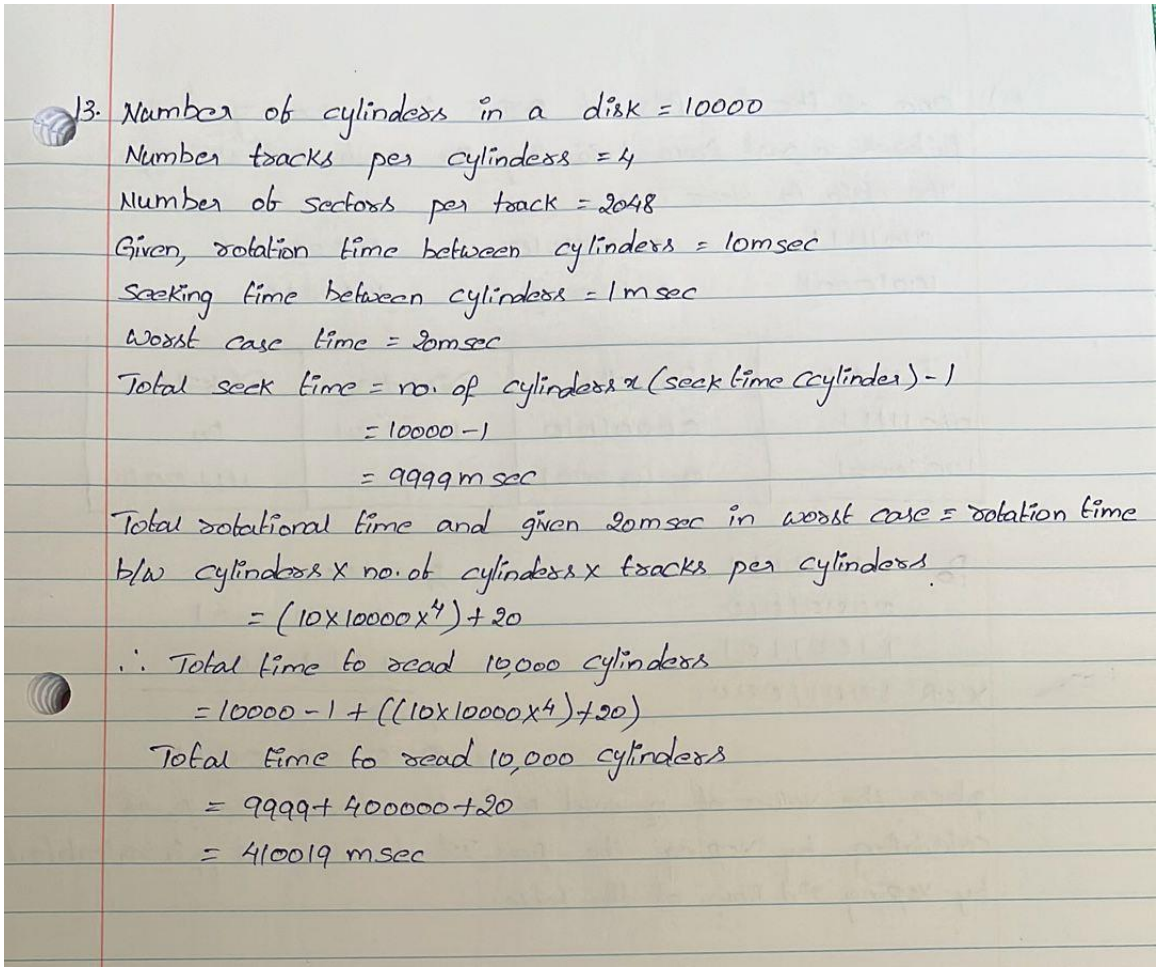
$$= \frac{40}{200}$$

$$= 0.2$$

percentage slowness $= 0.2 \times 100$

$= 20\%$

13. How long does it take to read a disk with 10,000 cylinders, each containing four tracks of 2048 sectors? First, all the sectors of Track 0 are to be read starting at Sector 0, then all the sectors of Track 1 starting at sector 0, and so on. The rotation time is 10 msec, and a seek takes 1 msec between adjacent cylinders and 20 msec for the worst case. Switching between tracks of a cylinder can be done instantaneously. (10%)



14. Let us assume we use RAID level 5 to store the following binary data:
01011111 00001010 11001101
10010001 00100001 11110011
Let us assume we have 4 disk drives, and the size of a strip is 8 bits. Illustrate how the data is stored on the 4 drives. Fill the table below. (10%)
(Note that in practice, a strip is much larger than 8 bits. A strip contains at least 1 sector, and a sector is typically 4096 bits.)

RAID level 5			
Disk drive #1	Disk drive #2	Disk drive #3	Disk drive #4
01011111			

4) One of the benefits of RAID 5 is overcoming disk failures, a part from balancing the reads and writes
The data to store is

01011111	00001010	11001101
10010001	00100001	11110011

Disk#1	Disk#2	Disk#3	Disk#4
01011111	00001010	11001101	P_0
10010001	00100001	P_1	11110011

P_0 = 01011111
00001010
11001101
XOR 10011000

P_1 = 10010001
00100001
11110011
XOR
XOR 0100 0011

place the value of P_0 and P_1 in the above table P_0 is calculating by xoring the row 1st of table and P_1 is calculated by xoring 2nd row of the table.