QUESTION 1

Given that,

a)

There are 32-bit microprocessors with 32-bit instruction

From left side first 8 bits are opcode

Then operand = 24 bits

So that maximum directly addressable memory capacity = 2^24

= 16777216

b)the impact on the system speed if the microprocessor bus has

1.) a 32-bit local address bus and a 16-bit local data bus:

As per given local address bus is 32-bit,and we also have 32-bit instruction,the entire address can be decoded once.

And given that data bus is 16-bits, to fetch 32-bit instrcution we need two cycles.

2)

local address bus is 16-bit,and we have 32-bit instruction, sp that to fetch the instruction it takes two cycles.

Local data bus also 16-but, again it takes two cycles.

Totally 4 cycles.

c) No.of operands = 24 bits

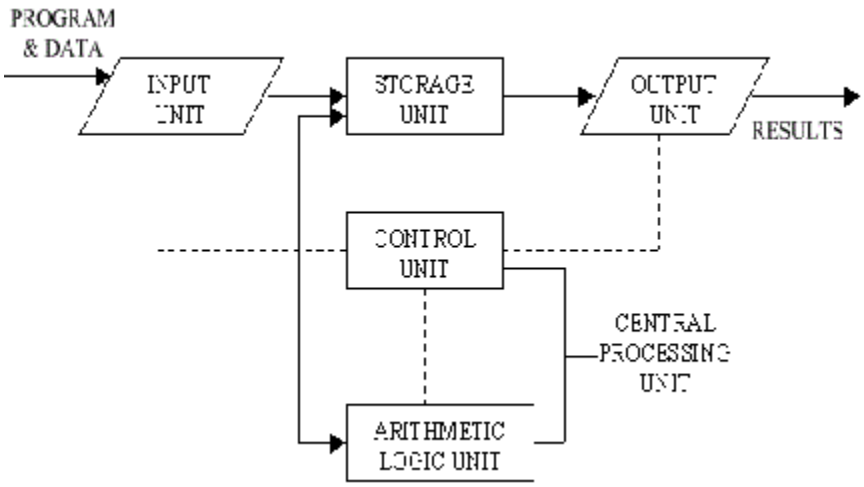
Since operands are 24 bits, program counter needs atlease 24 bits.

Instruction register stores present instruction, hence it needs 32-bits

1. RAID is a set of physical disk drives viewed by the operating system as a single logical drive.

   2. Data are distributed across the physical drives of an array in a scheme known as striping.

   3. Redundant disk capacity is used to store parity information, which guarantees data recoverability in case of a disk failure.



1. Input: This is the process of entering data and programs in to the computer system. You should know that computer is an electronic machine like any other machine which takes as inputs raw data and performs some processing giving out processed data. Therefore, the input unit takes data from us to the computer in an organized manner for processing.

2. Storage: The process of saving data and instructions permanently is known as storage. Data has to be fed into the system before the actual processing starts. It is because the processing speed of Central Processing Unit (CPU) is so fast that the data has to be provided to CPU with the same speed. Therefore the data is first stored in the storage unit for faster access and processing. This storage unit or the primary storage of the computer system is designed to do the above functionality. It provides space for storing data and instructions.

The storage unit performs the following major functions:

• All data and instructions are stored here before and after processing.

• Intermediate results of processing are also stored here.

3. Processing: The task of performing operations like arithmetic and logical operations is called processing. The Central Processing Unit (CPU) takes data and instructions from the storage unit and makes all sorts of calculations based on the instructions given and the type of data provided. It is then sent back to the storage unit.

4. Output: This is the process of producing results from the data for getting useful information. Similarly the output produced by the computer after processing must also be kept somewhere inside the computer before being given to you in human readable form. Again the output is also stored inside the computer for further processing.

5. Control: The manner how instructions are executed and the above operations are performed. Controlling of all operations like input, processing and output are performed by control unit. It takes care of step by step processing of all operations inside the computer.

QUESTION 2

clock rate = 200 MHz

Machine A

Total instruction = (8 + 4 + 2 + 4) = 18 millions

total clock = (8*1 + 4*3 + 2*4 + 4*3) = (8 + 12 + 8 + 12) = 40 millon Cycles

CPI = 40/18 = 2.22

MIPS = 200*10^6/(40/18) = 90 Millons instructions per second

Execution time = Total clock/clock rate

=> 40*10^6/(200 * 10^6) = 0.2 seconds

Machine B

Total instruction = (10 + 8 + 2 + 4) = 24 millions

total clock = (10*1 + 8*2 + 2*4 + 4*3) = (10 + 16 + 8 + 12) = 46 millon Cycles

CPI = 46/24 = 1.916

MIPS = 200*10^6/(46/24) = 104.34 Millons instructions per second

Execution time = Total clock/clock rate

=> (46*10^6)/(200 * 10^6) = 0.23 seconds

(b)Machine A is faster than machine B

B

Direct Memory Access can be abbreviated to DMA, which is a feature of computer systems. It allows input/output (I/O) devices to access the main system memory (random-access memory), independent of the central processing unit (CPU), which speeds up memory operations.

Cycle Stealing Mode

The cycle stealing mode is used in a system where the CPU cannot be disabled for the length of time required for the burst transfer mode. In the cycle stealing mode, the DMA controller obtains the access to the system bus by using the BR (Bus Request) and BG (Bus Grant) signals, which are the same as the burst mode. These two signals control the interface between the CPU and the DMA controller.

QUESTION 3

Stack condition after each instruction.

empty stack -> _____

(PUSH 4) -> __4__ (As Push will add element on top of stack)

(PUSH 7) - > 7

__4__ (As Push will add element on top of stack)

(PUSH 8) - > 8

7

__4__ (As Push will add element on top of stack)

ADD -> 15

__7__ (As Add will Add top two elements i.e 7 + 8 and Push in stack)

(PUSH 10) - > 10

15

__7__ (As Push will add element on top of stack)

SUB -> 5

__7__ (As SUB will subtract top two elements i.e 15 - 10 and Push in stack)

MUL -> __35__ (As MUL will multiply top two elements i.e 7 * 5 and Push in stack.)

So, Only emenent left in stack is 35.

3B

Condition codes are bits set by the CPU hardware as the result of operations.

Condition Codes

Advantages	Disadvantages
1. Because condition codes are set by normal arithmetic and data movement instructions, they should reduce the number of COMPARE and TEST instructions needed.	1. Condition codes add complexity, both to the hardware and software. Condition code bits are often modified in different ways by different instructions, making life more difficult for both the microprogrammer and compiler writer.
2. Conditional instructions, such as BRANCH are simplified relative to composite instructions, such as TEST AND BRANCH.	2. Condition codes are irregular; they are typically not part of the main data path, so they require extra hardware connections.
3. Condition codes facilitate multiway branches. For example, a TEST instruction can be followed by two branches, one on less than or equal to zero and one on greater than zero.	3. Often condition code machines must add special non-condition-code instructions for special situations anyway, such as bit checking, loop control, and atomic semaphore operations.
4. Condition codes can be saved on the stack during subroutine calls along with other register information.	4. In a pipelined implementation, condition codes require special synchronization to avoid conflicts.

3C

Characteristics of RISC:

- It has simpler instructions and thus simple instruction decoding.
- More general-purpose registers.
- The instruction takes one clock cycle in order to get executed.
- The instruction comes under the size of a single word.
- Pipeline can be easily achieved.
- Few data types.
- Simpler addressing modes.

QUESTION 4

Answer to 3(a)

Clock Time (CT) is the reciprocal of the **clock** frequency. For a 4 GHz processor has a **cycle** time of

$$CT = \frac{1}{10 \times 10^9} = 0.1 \times 10^{-9} \text{ sec} = 0.1\ nsec$$

Time required for Initial Fetching and Decoding = 10 CT = 10×0.1nsex=1nsec
Time required to transfer 1 byte = 15 CT = 15 = 15×0.1nsec=1.5 nsec
Time required to transfer 64 byte = 64×1.5nsec=1.5 nsec = 96 nsec
The total instruction cycle time = 1 nsec + 96 nsec = 97 nsec

Answer to 3(b)

The worst case delay of acknowledging an interrupt request in case of
1+The size of the partition allocated to the program in bytes × 1.5 nsec

Answer to 3(c)

If the block data transfer instruction can be interrupted at the beginning of each byte, then worst case delay in acknowledging an interrupt request will be equal to the time required to transfer 1 byte, that is, 1.5 nsec

Answer to 4(a)

Clock Time (CT) non-pipelined process = 2.5 GHz processor = 0.4 nsec
Time required to execute one instruction = 4 × 0.4 nsec = 1.6 nsec

Clock Time (CT) pipelined process = 2 GHz processor = 0.5 nsec
Time required to execute one instruction in a 5-line pipeline = $\frac{1}{5} \times 4 \times 0.5\ nsec$ = 0.4 nsec

Increase in speed due to pipelining = $\frac{(1.6-0.4)}{1.6} = 0.75 = 75\%$

Answer to 4(b)

Time required for executing an instruction in a non-pipelined processor = 1.6 nsec
Number instructions executed in 1 Sec= $\frac{1}{1.6} \times 10^9 = 0.625 \times 10^9$
MIPS of the non-pipelined process = 625 MIPS

Time required for executing an instruction in a non-pipelined processor = 0.4 nsec
Number instructions executed in 1 Sec= $\frac{1}{0.4} \times 10^9 = 2.5 \times 10^9$
MIPS of the non-pipelined process = 2500 MIPS

4B

Logic shift is defined as a shift of bits either left or right with vacant bits filled up with zeros. **Arithmetic shift** is defined as shift of bits either left or right with sign bit preserved if possible. Arithmetic shift diffs from logic shift only when negative numbers are involved.

4C

Big-endian is an order in which the "big end" (most significant value in the sequence) is stored first, at the lowest storage address**. Little-endian** is an order in which the "little end" (least significant value in the sequence) is stored first.

4D

Answer→ Given that, **X = (A + B x C) / (D - E x F).**

0 Address	1 Address	2 Address	3 Address
1.) PUSH A → A 2.) PUSH B → B,A 3.) PUSH C → C,B,A 4.) MUL → (C x B), A 5.) ADD → A + (C x B) 6.) PUSH D → D, A + (C x B) 7.) PUSH E	1.) LOAD E → AC ⇐ E 2.) MUL F → AC⇐ AC * F 3.) STORE T → T ⇐ AC 4.) LOAD D → AC ⇐ D 5.) SUB T → AC ⇐ AC-T	1.) MOVE R0,E → R0 ⇐ E 2.) MUL R0,F → R0 ⇐ R0 * F 3.) MOVE R1,D → R1 ⇐ D 4.) SUB R1,R0 → R1 ⇐ R1 - R0 5.) MOVE R0,B → R0 ⇐ B	1.) MUL R0,E,F → R0 ⇐ E x F 2.) SUB R0,D,R0 → R0 ⇐ D - R0 3.) MUL R1,B,C → R1 ⇐ B x C 4.) ADD R1,A,R1 → R1 ⇐ A + R1

→ E,D, A + (C x B) 8.) PUSH F → F,E,D, A+(BxC) 9.) MUL → (ExF),D, A+(BxC) 10.) SUB → D - (ExF), A+(BxC) 11.) DIV → (A+B*C) / (D-E*F) 12.) POP X → X = (A+B*C) / (D-E*F)	6.) STORE T → T ⇐ AC 7.) LOAD B → AC ⇐ B 8.) MUL C → AC⇐ AC * C 9.) ADD A → AC ⇐ AC + A 10.) DIV T → AC ⇐ AC/T 11.) STORE X → X ⇐ AC	6.) MOVE R0, C → R0 ⇐ C 7.) ADD R0, A → R0 ⇐ R0 + A 8.) DIV R0,R1 → R0 ⇐ R0/R1 9.) MOVE X, R0 → X ⇐ R0	5.) DIV X,R0,R1 → X ⇐ R0/R1