Computer Organization and Architecture: Question Bank Solutions

Reference

The solutions provided are based on the concepts and terminology found in **William Stallings, *Computer Organization and Architecture: Designing for Performance* (9th Edition)**.

Group 1: Fundamentals and Core Concepts (Q1 - Q40)

Q1. What is computer architecture?

Computer architecture refers to the **attributes of a system visible to the programmer** [1]. These attributes have a direct impact on the logical execution of a program. Examples include the instruction set, the number of bits used to represent various data types (e.g., numbers, characters), I/O mechanisms, and techniques for addressing memory.

Q2. What is computer Organisation?

Computer organization refers to the **operational units and their interconnections** that realize the architectural specifications [1]. These are the hardware details transparent to the programmer. Examples include control signals, interfaces between the computer and peripherals, and the memory technology used.

Q3. What is the diffrence between computer organisation and computer architecture?

The key difference is that **Architecture** is concerned with *what* the system does (programmer's view), while **Organization** is concerned with *how* it does it (hardware implementation) [1].

Q4. What is pipelining?

Pipelining is a technique that allows a processor to **overlap the execution of multiple instructions** [1]. It is analogous to an assembly line, where an instruction is processed in a series of sequential stages (e.g., Fetch, Decode, Execute). By having different stages process different instructions simultaneously, the overall throughput (rate of instruction execution) is increased, even though the time to execute a single instruction (latency) may not be reduced.

Q5. What is instruction pipeline?

An instruction pipeline is the specific implementation of pipelining within the CPU, where the instruction cycle (Fetch, Decode, Execute, etc.) is broken down into a series of stages, allowing multiple instructions to be in different stages of execution concurrently [1].

Q6. What is instruction register?

The **Instruction Register (IR)** holds the **instruction currently being executed** [1]. The Opcode and operand address of the instruction in the IR are used to determine the operations to be performed.

Q7. What is program Counter?

The **Program Counter (PC)**, also known as the Instruction Pointer, holds the **address of the next instruction to be fetched** [1]. After an instruction is fetched, the PC is typically incremented to point to the next sequential instruction.

Q8. What are registers?

Registers are **small, high-speed storage locations within the CPU** [1]. They are used to temporarily hold data, addresses, and control information that the CPU is actively working with, providing the fastest level of memory access.

Q9. What is Accumulator?

The **Accumulator (AC)** is a general-purpose register used to **store the intermediate results of arithmetic and logic operations** [1]. In many older or simpler CPU designs, one operand is implicitly held in the Accumulator, and the result of the operation is stored back into it.

Q10. What is ALU?

The **Arithmetic Logic Unit (ALU)** is the functional unit of the CPU that **performs arithmetic operations** (addition, subtraction, etc.) **and logical operations** (AND, OR, NOT, etc.) on data [1].

Q11. What is MAR?

The **Memory Address Register (MAR)** holds the **address of the memory location** from which data is to be read or to which data is to be written [1].

Q12. What is MDR?

The **Memory Data Register (MDR)**, also known as the Memory Buffer Register (MBR), contains the **data to be written into memory or the data most recently read from**

memory [1].

Q34. What is Control Unit?

The **Control Unit (CU)** is the functional unit of the CPU that **controls the operation of the CPU and the computer's components** [1]. It interprets instructions and generates the control signals (timing and control) needed to execute them, directing the flow of data between the ALU, registers, and memory.

Q35. What is Instruction Buffer Register?

The **Instruction Buffer Register (IBR)** is a temporary storage register in the CPU used to **hold the right-hand instruction** of a word fetched from memory, while the left-hand instruction is executed [1]. This is a feature of early architectures like the IAS computer to speed up instruction fetching.

Q13. What is Control Lines?

Control lines are part of the control bus, used to **transmit timing and control signals** [1]. These signals are used to synchronize the operations of the system modules and specify the type of operation being performed (e.g., memory read, memory write, I/O request).

Q14. What is Address Lines?

Address lines are part of the address bus, used to **designate the source or destination of the data** on the data bus [1]. The width of the address bus determines the maximum possible memory capacity of the system.

Q15. What is Data Lines?

Data lines are part of the data bus, used to **transfer data between system modules** (CPU, memory, I/O) [1]. The number of data lines determines the width of the data path, which is a key factor in system performance.

Q16. What is bus Arbitration?

Bus arbitration is the **process of determining which device (CPU or I/O module) is granted control of the bus** when multiple devices request access simultaneously [1]. This is necessary to prevent conflicts and ensure orderly data transfer.

Q17. What is Centralized Bus Arbitration ?

In centralized bus arbitration, a **single hardware module (the bus controller or arbiter)** is responsible for granting bus access [1]. The arbiter receives requests from all potential masters and grants the bus to one based on a specific priority scheme.

Q37. What is bus structure ?

A bus structure is a **communication pathway connecting two or more devices** [1]. It typically consists of multiple lines (data, address, and control) and is a shared transmission medium, meaning only one device can transmit successfully at a time.

Q40. What is single bus structure ?

A single bus structure is a computer organization where the **CPU, main memory, and I/O devices all share a single common bus** for communication [1]. While simple and cost-effective, it creates a bottleneck as only one transfer can occur at any given time.

Q18. What is Instruction fetch?

Instruction fetch is the **first stage of the instruction cycle**, where the CPU reads the next instruction from the memory location whose address is stored in the Program Counter (PC) [1]. The instruction is then loaded into the Instruction Register (IR).

Q19. What is Operand fetch?

Operand fetch is the process, typically part of the **Execute stage**, where the CPU retrieves the **data required for the instruction** (the operands) from memory or registers [1].

Q20. What is interrupt?

An interrupt is a **mechanism by which other modules (I/O, memory) can interrupt the normal sequence of the processor's execution** [1]. It is a signal that causes the CPU to suspend its current program and execute a special routine (Interrupt Service Routine) to handle the event.

Q21. What is Carry bit?

The carry bit (or carry flag) is a **single bit in the Processor Status Word (PSW) or flag register** that is set to 1 when an arithmetic operation results in an **overflow out of the most significant bit (MSB)** of the result [1]. It is primarily used for multi-word arithmetic.

Q22. What is Borrow bit?

The borrow bit is a flag, similar to the carry bit, that is set during a **subtraction operation when a borrow is required** into the most significant bit [1]. It is often implemented using the carry flag in two's complement arithmetic.

Q23. What is Booth's Algorithm?

Booth's algorithm is a **multiplication algorithm** that allows for faster multiplication of signed binary numbers in two's complement representation [1]. It works by recoding the multiplier to reduce the number of partial products that need to be generated and added.

Q24. What Sign bit 1 represents?

In signed number representations (Sign-Magnitude, One's Complement, Two's Complement), a **sign bit of 1 represents a negative number** [1]. The sign bit is the most significant bit (MSB) of the number.

Q25. What is signed Number?

A signed number is a binary number representation that **includes a mechanism to represent both positive and negative values** [1]. Common methods include Sign-Magnitude, One's Complement, and Two's Complement.

Q26. What is Unsigned Number?

An unsigned number is a binary number representation where **all bits are used to represent the magnitude** of the number, and only non-negative values (zero and positive integers) can be represented [1].

Q27. What is arithmatic right shift?

An arithmetic right shift is a bitwise operation that **shifts all bits to the right by one position**, but the **most significant bit (MSB) is preserved** (copied) to maintain the sign of the number [1]. This operation is equivalent to division by 2 for signed numbers.

Q28. What is arithmatic left shift?

An arithmetic left shift is a bitwise operation that **shifts all bits to the left by one position**, with a **0 being inserted into the least significant bit (LSB)** [1]. This operation is equivalent to multiplication by 2, provided no overflow occurs.

Q29. What is Radix point?

The radix point is the **symbol that separates the integer part from the fractional part** of a number in any base (radix) [1]. In base 10, it is called the decimal point; in base 2, it is called the binary point.

Q30. What is One's complement?

The one's complement of a binary number is obtained by **inverting every bit** (changing all 0s to 1s and all 1s to 0s) [1].

Q31. What is Two's complment?

The two's complement of a binary number is calculated by **taking the one's complement and adding 1** to the least significant bit (LSB) [1]. It is the most common method for representing signed integers in computers.

Q32. How to calculate one's complement?

To calculate the one's complement, simply **invert each bit** of the binary number. For example, the one's complement of 1010 is 0101 [1].

Q33. How to calculate two's complement?

To calculate the two's complement, follow these two steps [1]:

1. Find the one's complement of the number.
2. Add 1 to the least significant bit (LSB) of the one's complement result.

Q36. What is restoring method?

The restoring method (or restoring division algorithm) is a **binary division algorithm** [1]. In this method, if the partial remainder becomes negative after subtracting the divisor, the original partial remainder is **restored** by adding the divisor back. This process is repeated for each bit of the quotient.

Q38. Explain truth table for binary arithmatic addition.

The truth table for binary addition (Full Adder) considers two input bits (A and B) and a Carry-In bit (C_{in}), producing a Sum bit (S) and a Carry-Out bit (C_{out}) [1].

A	B	C_in	S	C_out
0	0	0	0	0
0	0	1	1	0
0	1	0	1	0
0	1	1	0	1
1	0	0	1	0
1	0	1	0	1

1	1	0	0	1
1	1	1	1	1

Q39. Explain truth table for binary arithmatic subtraction.

The truth table for binary subtraction (Full Subtractor) considers two input bits (A and B) and a Borrow-In bit (B_{in}), producing a Difference bit (D) and a Borrow-Out bit (B_{out}) [1].

A	B	B_in	D	B_out
0	0	0	0	0
0	0	1	1	1
0	1	0	1	1
0	1	1	0	1
1	0	0	1	0
1	0	1	0	0
1	1	0	0	0
1	1	1	1	1

Group 2: Numerical Problems (Q41 - Q50)

Q41. Calculate one's complement of 1011.

Binary: 1011 **One's Complement:** 0100

Q42. Calculate one's complement of 10101001.

Binary: 10101001 **One's Complement:** 01010110

Q43. Calculate one's complement of 10111101.

Binary: 10111101 **One's Complement:** 01000010

Q44. Calculate two's complement of 1100.

Binary: 1100 **One's Complement:** 0011 **Two's Complement:** 0100

Q45. Calculate two's complement of 11100101.

Binary: 11100101 **One's Complement:** 00011010 **Two's Complement:** 00011011

Q46. Calculate two's complement of 11001000.

Binary: 11001000 **One's Complement:** 00110111 **Two's Complement:** 00111000

Q47. Solve 3 * -4 by booth's algorithm.

Multiplicand (M): 3 (00000011) **Multiplier (Q):** -4 (11110100) **-M (2's Complement):** 11111101

Step	A	Q	Q-1	Operation
0	00000000	11111100	0	Initial State
1	00000000	01111110	0	Arithmetic Shift -> ASR
2	00000000	00111111	0	Arithmetic Shift -> ASR
3	11111110	10011111	1	A = A - M -> ASR
4	11111111	01001111	1	Arithmetic Shift -> ASR
5	11111111	10100111	1	Arithmetic Shift -> ASR
6	11111111	11010011	1	Arithmetic Shift -> ASR
7	11111111	11101001	1	Arithmetic Shift -> ASR
8	11111111	11110100	1	Arithmetic Shift -> ASR

Final Product (A Q): 1111111111110100 **Decimal Result:** -12

Q49. Solve 5 * 4 by booth's algorithm.

Multiplicand (M): 5 (00000101) **Multiplier (Q):** 4 (00010100) **-M (2's Complement):** 11111011

Step	A	Q	Q-1	Operation
0	00000000	00000100	0	Initial State
1	00000000	00000010	0	Arithmetic Shift -> ASR
2	00000000	00000001	0	Arithmetic Shift -> ASR
3	11111101	10000000	1	A = A - M -> ASR
4	00000001	01000000	0	A = A + M -> ASR
5	00000000	10100000	0	Arithmetic Shift -> ASR
6	00000000	01010000	0	Arithmetic Shift -> ASR
7	00000000	00101000	0	Arithmetic Shift -> ASR
8	00000000	00010100	0	Arithmetic Shift -> ASR

Final Product (A Q): 0000000000010100 **Decimal Result:** 20

Q48. Solve 10/ 3 by division restoring algorithm.

Dividend (Q): 10 (0000) **Divisor (M):** 3 (0011) **-M (2's Complement):** 1101

Step	A	Q	Operation
0	0000	1010	Initial State
1	0000	0100	Shift Left, A = A - M (Negative), Restore A, Q0 = 0
2	0000	1000	Shift Left, A = A - M (Negative), Restore A, Q0 = 0

3	0000	0000	Shift Left, A = A - M (Negative), Restore A, Q0 = 0
4	0000	0000	Shift Left, A = A - M (Negative), Restore A, Q0 = 0

Final Quotient (Q): 0000 (0) **Final Remainder (A):** 0000 (0)

Q50. Solve 8/ 2 by division restoring algorithm.

Dividend (Q): 8 (0000) **Divisor (M):** 2 (0010) **-M (2's Complement):** 1110

Step	A	Q	Operation
0	0000	1000	Initial State
1	0000	0000	Shift Left, A = A - M (Negative), Restore A, Q0 = 0
2	0000	0000	Shift Left, A = A - M (Negative), Restore A, Q0 = 0
3	0000	0000	Shift Left, A = A - M (Negative), Restore A, Q0 = 0
4	0000	0000	Shift Left, A = A - M (Negative), Restore A, Q0 = 0

Final Quotient (Q): 0000 (0) **Final Remainder (A):** 0000 (0)

References

[1]: William Stallings, *Computer Organization and Architecture: Designing for Performance* (9th Edition). (Source for all definitions and algorithms.)