Computer Organisation and Architecture (EET 2211)

Name → Dinanath Dash

Regd → 2241004161

Sl → 06

Sec → 2241026

Branch → Branch

Q1) What is the difference between DRAM and SRAM in terms of application?

Ans→ DRAM (Dynamic Random Access Memory) and SRAM (Static Random Access Memory) serve as primary memory components in computers but differ in their characteristics & applications.

i) Volatility → DRAM is volatile memory, meaning it requires continuous power to retain data, whereas SRAM is non-volatile, meaning it retains data even when the power is turned off.

ii) Speed → SRAM is faster than DRAM because it doesn't need to be refreshed as frequently.

iii) Density → DRAM is denser than SRAM meaning it can store more data in the same physical space.

iv) Cost → DRAM is generally cheaper per bit compared to SRAM, making it more cost-effective for storing large amount of data.

In summary, DRAM is typically used for main memory in computers due to its high density & lower cost, while SRAM is used for cache memory and registers in CPUs where speed & low latency are essential.

Q2) Explain different types of ROM.

Ans→ ROM (Read Only Memory) comes in various types, each designed for specific applications & functions.

i) Mask ROM (MROM): This type of ROM is programmed during the manufacturing process. The data is "masked" onto the chip using a photolithographic process, making it permanent & unchangeable. Mask ROM is used for firmware and boot code in devices where the data doesn't need to be modified.

ii) Programmable ROM (PROM): PROM allows the user to write data onto the chip once using a special device called PROM programmer. Once programmed, the data is permanent & cannot be changed. PROM is used for applications where the data needs to be written once during manufacturing process but doesn't need to be modified afterward.

iii) Erasable Programmable ROM (EPROM): EPROM allows for multiple write cycles, allowing the user to erase & reprogram the data multiple times using UV light exposure. EPROM chips have a transparent window on the top that allows UV light to penetrate and erase the data. EPROM is used for applications where occasional updates or changes to the stored are required.

iv) Electrically Erasable Programmable ROM (EEPROM): EEPROM allows for multiple write cycles, similar to EPROM, but can be erased and reprogrammed electrically without the need for UV light exposure. EEPROM is used in applications where frequent updates or changes to the data are required, such as BIOS in computers and firmware in embedded systems.

v) Flash Memory: Flash memory is a type of EEPROM that allows for even faster erasing and programming. It is used in a wide range of applications, including USB drives, solid-state drives (SSDs), memory cards & mobile devices.

Q3) How is the syndrome for the Hamming code interpreted?

Ans→ The syndrome for a Hamming code is interpreted to detect and correct errors in transmitted data.

i) Calculate Syndrome → The syndrome is calculated by performing an exclusive-OR (XOR) operation on the recieved codeword and the parity check of the matrix of the Hamming code.

ii) Interpretation → The syndrome vector serves as an indicator of whether an error has occurred, and, if so, where it is located in the codeword.

iii) Error Correction → Once the error position is identified, the erroneous bit is flipped to correct the error.

Q4) Explain the error correction process in the memory system using a suitable diagram.

Ans→ i) Data and ECC Generation →

a) The original data to be stored in memory is fed into an ECC encoder.

b) The ECC encoder generates additional bits (ECC bits) based on the original data using specific algorithms such as Hamming codes or Reed-Solomon codes.

c) The data & ECC bits are combined to form a codeword.

ii) Storage in memory →

a) The codeword, consisting of both the original data and ECC bits, is stored in memory cells.

b) Each memory cell typically stores a portion of the data along with its corresponding ECC bits.

iii) Data Retrieval →

a) When the data is retrieved from memory, it is read along with its associated ECC bits.

iv) Error Detection →

a) The retrieved data, along with ECC bits, is passed through an ECC decoder.

b) The ECC decoder compares the recieved ECC bits with the ECC bits generated from the retrieved data.

c) If errors are detected (discrepancies between recieved and generated ECC bits), the ECC decoder generates a syndrome indicating the location and nature of the errors.

v) Error Correction →

a) Based on the syndrome generated by the ECC decoder, error correction is performed.

b) Single-bit errors can often be corrected by flipping the erroneous bit to its correct state.

c) For multiple-bit errors or uncorrectable errors, error recovery mechanisms such as retransmission may be initiated.

vi) Corrected Data Output →

a) Once errors are corrected (if possible), the corrected data is provided as the output for further processing or transmission.

Q5) If there are m input lines, n output lines for a decoder that is used to uniquely address a byte addressable 1KB RAM, then the minimum value of m+n is ___.

Ans → 1 KB = 1024 B = 2^{10} B

10×2^{10} decoders

So, input lines (m) = 10

No. of output lines (n) = $2^m = 10^{10} = 1024$

Hence, (m+n) = 10 + 1024 = 1034

~~Q6) Suppose an 8~~

Q6) Suppose an 8-bit data word stored in memory is 11000010.

1 1 0 0 0 0 1 0

D_8 D_7 D_6 D_5 D_4 D_3 D_2 D_1

12	11	10	9	8	7	6	5	4	3	2	1
1100	1011	1010	1001	1000	0111	0110	0101	0100	0011	0010	0001
D_8	D_7	D_6	D_5	C_8	D_4	D_3	D_2	C_4	D_1	C_2	C_1
1	1	0	0		0	0	1		0		

$C_1 = D_1 \oplus D_2 \oplus D_4 \oplus D_5 \oplus D_7$

$= 0 \oplus 1 \oplus 0 \oplus 0 \oplus 1$

$= 0$

$C_2 = D_1 \oplus D_3 \oplus D_4 \oplus D_6 \oplus D_7$

$= 0 \oplus 0 \oplus 0 \oplus 0 \oplus 1$

$= 1$

$C_4 = D_2 \oplus D_3 \oplus D_4 \oplus D_8$

$= 1 \oplus 0 \oplus 0 \oplus 1$

$= 0$

$C_8 = D_5 \oplus D_6 \oplus D_7 \oplus D_8$

$= 0 \oplus 0 \oplus 0 \oplus 1$

$= 0$

Check bits → 0010

Data → 1100 0010 0010

Q7) 8 bit word → 00111001 , Check bits stored with it → 0111
$D_8 D_7 D_6 D_5 D_3 D_2 D_1$ $C_8 C_4 C_2 C_1$

Check bits to be calculated → 1101
$C_8 C_4 C_2 C_1$

12	11	10	9	8	7	6	5	4	3	2	1
1100	1011	1010	1001	1000	0111	0110	0101	0100	0011	0010	0001
D_8	D_7	D_6	D_5	C_8	D_4	D_3	D_2	C_4	D_1	C_2	C_1
0	0	1	1	0	1	0	0	1	1	0	1

C_8	C_4	C_2	C_1
0	1	1	1
⊕ 1	1	0	1
1	0	1	0

→ (10 in binary)

Data word that was read from memory → 001111000110

Q8)

RAID Level	Parity Distribution	Fault tolerance	Write Performance	Storage Efficiency
RAID 4	Dedicated Parity Disk	single disk failure	Moderate	Good (except for parity disk)
RAID 5	Distributed parity	Single disk failure	Better	Good
RAID 6	Distributed double parity	Two disk failures	Moderate	Lower than RAID 5 Due to double parity

RAID 4

A1	A2	A3	Parity A
B1	B2	B3	Parity B
C1	C2	C3	Parity C
D1	D2	D3	Parity D
Disk 1	Disk 2	Disk 3	Disk 4

RAID 5

A1	A2	A3	Parity A
B1	B2	Parity B	B3
C1	Parity C	C_2	C3
Parity D	D1	D_2	D3
Disk 1	Disk 2	Disk 3	Disk 4

RAID 6

A1	B1	Parity B1	Parity B1
A2	Parity B2	Parity B2	B2
Parity B3	Parity B3	A3	B3
Parity B4	A4	B4	Parity B4
Disk 1	Disk 2	Disk 3	Disk 4

Q9) Total storage capacity = 4 * 200 = 800 GB

RAID 0 → 4 * 200 = 800 GB

RAID 1 → (4/2) * 200 = 400 GB

RAID 3 → (4-1) * 200 = 600 GB

RAID 4 → (4-1) * 200 = 600 GB

RAID 5 → (4-1) * 200 = 600 GB

RAID 6 → (4-2) * 200 = 400 GB

Q10) The major functions of an I/O module →

i) Data Transfer → I/O modules manage the transfer of data between the CPU & external devices, such as storage devices, network interfaces & peripherals.

ii) Interface conversion → They provide interface conversion between the internal system bus used by the CPU & the various types of external interfaces and by peripheral devices.

iii) Buffering → I/O modules often include buffers to temporarily store data during the transfer process.

iv) Control and Timing → They manage the timing and control signals required for data transfer operations.

v) Error handling → I/O modules are responsible for error detection and handling during data transfers.

Q11) External or peripheral devices can be broadly classified into three categories based on their function & interaction with the computer system:

i) Input devices →

a) These are used to input data or commands into the computer system.

b) Examples includes keyboards, mice, touchscreens, trackpads, scanners barcode readers, joysticks, microphones and webcams.

ii) Output devices →

a) These are used to recieve processed data from the computer system & present it to the user in a perciable form.

b) Examples include monitors (displays), printers, speakers, projectors, headphones, and tactile feedback devices.

iii) Storage devices →

a) These are used to used to store data and programs persistently for future retrieval purpose.

b) Examples include hard disk drives (HDDs), SSDs, USB drives, memory cards, optical discs (e.g. CDs, DVDs, Blu-ray discs), & network attached storage (NAS) devices.

Q12) The 8255A is a popular programmable peripheral interface chip used in microprocessor-based systems to provide parallel I/O capabilities. It consists of three 8-bit bidirectional I/O ports (Port A, Port B & Port C), which can be configured & controlled using different modes of operation.

i) Mode 0 (Basic Input/Output Mode) : a) In mode 0, each of the 3 ports can be programmed as either input or output.

b) The direction of each port is controlled by setting or clearing bits in the Control Word Register (CW) of the 8255A.

ii) Mode 1 (Strobed Input/Output Mode):

a) Mode 1 is similar to Mode 0 but includes an additional feature for input operations called "strobe" or "hand shake" mode.

b) In this mode, input data is latched when a strobe signal (STB) is received by the 8255A.

iii) Mode 2 (Bidirectional Bus Mode):

a) Mode 2 configures Port A as an input/output port and Port B & C as control ports.

b) Port A operates in bidirectional mode while Port B & C are used to control the direction of data transfer on Port A.

13) To configure the 8255A as described:

a) Port A as input

b) Port B as output

c) All bits of Port C as output

The control register for the 8255A (usually labeled as "Control Word") is an 8-bit register. The configuration bits for each port are as follows:

a) Bit 0 & bit 1: Control bits for Port A

b) Bit 2 & bit 3: Control bits for Port B

c) Bit 4, Bit 5 and Bit 6: Control bits for Port C
d) Bit 7: Mode Set bit

To configure as described:

a) Port A as input: Set control bits 0 and 1 to 0b00.
b) Port B as output: Set control bits 2 & 3 to 0b01.
c) All bits of Port C as output: Set control bits 4, 5, & 6 to 0b01.

So, the control register bits for this configuration would be:

a) Port A: 00
b) Port B: 01
c) Port C: 01
d) Mode set bit: Not specified, but typically set based on whether you're using Mode 0, Mode 1 or Mode 2.

15) i) 82C59A Interrupt Controller →

a) The 82C59A is a programmable interrupt controller (PIC) that manages interrupt requests from external devices.
b) It consists of multiple interrupt request lines (IRQ0 to IRQ7), each corresponding to a specific external device or I/O Port.

ii) Interrupt Request Lines:

a) External devices are connected to the IRQ lines of the 82C59A
b) Each device generates an interrupt request (IRQ) signal when it needs attention from the CPU.

iii) Interrupt Handler:

a) The interrupt handler is a software routine executed by the CPU in response to an interrupt request.
b) When an interrupt occurs, the CPU suspends its current execution and jumps to the appropriate interrupt handler based on the priority of the interrupt.

iv) Priority Resolver:

a) The 82C59A includes a priority resolver that determines the highest priority interrupt request pending at any given time.
b) When multiple interrupt requests are pending simultaneously, the priority resolver ~~interrupt request pending at any given time~~. selects the IRQ line with the highest priority & forwards it to the CPU.

v) CPU Execution →

a) Upon receiving the highest priority interrupt request from the 82C59A, the CPU suspends its current execution and jumps to the corresponding interrupt handler.

vi) Interrupt Acknowledge →

a) Once the CPU completes processing the interrupt, it sends an interrupt acknowledge (INTA) signal to the 82C59A to acknowledge the interrupt & reset the interrupt request line.

b) This allows other pending interrupts to be processed in subsequent cycles.