|  |  |  |
| --- | --- | --- |
| Boolean Algebra:  AND : x · y, x ∧ y NOT : ¬x, x, x ′ OR : x + y, x ∨ y XOR : x ⊕ y = (x ∧ ¬y) ∨ (¬x ∧ y) NOR : x ↓ y = ¬(x ∨ y) NAND : x ↑ y = ¬(x ∧ y) | Half Adder  Define Half Adders. Draw the circuit diagram and the truth table for a Half  Adder. - Sarthaks eConnect | Largest Online Education Community **NOTE** : Carry Column comes  before Sum Column  Full Adder Full-adder circuit diagram and truth table, where A, B, and C in are... |  Download Scientific Diagram **NOTE** : Carry Column comes  before Sum Column | Character Sets:  ASCII : American Standard Code for Information Interchange  : 7 bit (max)  UTF-8 : Uses upto four 8-bit bytes (32 bits) to represent characters  : Backward Compatible with ASCII |
| Binary Negative Representations:  - **Sign Magnitude**  - Invert only the sign bit of the number  - **One’s Complement:**  - Invert whole binary number (0 becomes 1 and 1 becomes 0)  - **Two’s Complement:**  - Add 1 to the one’s complement of binary number | Boolean Algebra Laws: |
| Decimal Point Values  0.5 0.25 0.125 0.0625 0.03125 | Computer Architecture:  - Memory : Array of bytes used for storing data, with each location having a numeric address. - Accumulator : Temporary Storage for values  - ALU (Arithmetic Logic Unit) : Performs all arithmetic and bitwise operations in a computer. - Program Counter : Contains next address to be fetched. - CIR (Current Instruction Register) : Holds instruction currently being executed / decoded.  - General Purpose Register : Stores both address and data.  - Instruction Decoder : Translates instruction code into the address. |
| Floating Point Format (Drexel)  seeeemmmmmmmmm s : sign bit e : exponent bits m : mantissa bits |
| CARDIAC Instruction Set:   |  |  |  | | --- | --- | --- | | **0** | **INP** | Read card into memory | | **1** | **CLA** | Load from Mem to ACC | | **2** | **ADD** | Add from Mem to ACC | | **3** | **TAC** | Test and JMP | | **4** | **SFT** | Shift Accumulator | | **5** | **OUT** | Write Mem to Output | | **6** | **STO** | Store ACC to Mem | | **7** | **SUB** | Subtract Mem from ACC | | **8** | **JMP** | JMP to address | | **9** | **HRS** | Halt and Reset | | CPU Architecture: |
| Compilation Process:  Lexical Analyzer : Breaks text into tokens using a keyword table and removes white spaces and comments Parser : Checks for structure (Syntax Errors) Symbol Table : Tracks the declaration and use of names Intermediate Code Generator : Creates alternate representation Code Generator : Produces error-free binary code Optimizer : Improves CPU usage or space used |
| Operating System:   - Manages hardware resources among competing programs - Provides common services to applications - Interfaces applications to hardware - Handles buffers and interrupts | Resource Manager:  - Allocates memory to programs - Prevents interrupts and allows buffers - Protects itself and the overall system from both malicious and accidental damage - Programs run at the pleasure of the OS | Process Management:  - Manages CPU in terms of running programs - Allocates memory to processes - Schedules processes to get CPU usage - **Switches between processes, called Context Switching** |
| JPEG:  - Joint Photographic Experts Group - Uses Lossy Compression using Discrete Cosine Transforms - Good Quality Compression for realistic images - Uses (Y, Cb, Cr) or (Y, U, V) color model. | PNG:  - Portable Network Graphics - Uses Lossless compression - Supports Multiple RGB Models - More advanced Compression than GIF | GIF:  - Graphics Interchange Format - Designed for 8-bit color maps - Lossless for 8-bit color maps, otherwise Lossy - Supports primitive animation - LZW Compression |

|  |  |  |
| --- | --- | --- |
| Bresenham’s Algorithm:  bresenham (x0, y0, x1, y1) {  dx = x1 – x0  dy = y1 – y0   y = y0  e = 0   repeat {  plot(x,y)  e = e + dy  if (2 \* e > dy) {  y = y + 1  e = e – dx  }   until x0 = x1  x0 = x0 + 1   } } | Symmetric Encryption:  - Uses a single key (shared secret) - Sender uses key to encrypt and embeds it in the message, and receiver decrypts with same key.  Asymmetric Encryption:   - Both sides generate their own public and private key sets, and send each of their public keys to one another. - Sender encrypts message using public key of receiver. - Receiver decrypts message using their own private key. | RSA (Rivest–Shamir–Adleman) Encryption:  - Choose 2 random numbers p and q - n = p \* q - r = (p – 1) \* (q – 1) - Pick e relatively prime to r - Find d such that e \* d = 1 mod r - Share e and n, d is private key for decryption - E(x) = xe mod n  - D(x) = xd mod n - x = xed mod n  - The message is broken into ASCII numbers separated by spaces, and each number is considered x individually. |
| RGB – 24 Bits Greyscale – 8 Bits Address : B + rW + c   : B – Base Address of Frame Buffer  : r – Pixel Row Number  : c – Pixel Column Number  : W – Width of a row | Digital Signatures:  - The plaintext to be sent is passed through a hashing algorithm, which produces a hexadecimal hash.  - This hash is then encrypted with the sender’s private key, and sent along with the message, which is encrypted in the receiver’s public key.  - The receiver decrypts the encrypted message using his private key, and the received hash with the earlier shared public key of the sender.  - After getting the plaintext (NOT HASH), he passes it through the same hashing algorithm used by the sender.  - The receiver then compares the received hash, as well as the calculated hash, and if both the hashes are the same, the sender is who he says he is. | Digital Certificate:  - Digital Certificate containing the public key and other information of the sender is issued by a certification authority guarantees sender’s identity. - It’s electronic document used to prove identity of a website / individual. |
|  |  |