**HITEC UNIVERSITY**

**Department of Computer Science**

**BS Computer Science Program**

|  |  |
| --- | --- |
| **Course Title:** | **CS-206 Computer Organization and assembly language 4(3+1)** |
| **Batch / Semester:** | Batch 2020 / 4th Semester |
| **Lab Instructor:** | Fatima Rauf |

**CS-206 Computer Organization and assembly language 4(3+1)**

**Project Ideas**

**Project Submission Due date: Last Week of Semester (Spring-24)**

**Students can choose any project from list of following projects.**

1. **Real-Time Operating System (RTOS) Kernel Development**: Design and implement a real-time operating system kernel in assembly language for a specific microcontroller architecture (e.g., ARM Cortex-M). The RTOS should support multitasking, task scheduling, synchronization primitives (such as semaphores or mutexes), and inter-process communication mechanisms.
2. **Audio Processing Effects Module:** Create an audio processing effects module in assembly language for a digital signal processor (DSP) platform. Implement various audio effects such as reverb, echo, chorus, and equalization, optimizing them for real-time performance and low memory footprint.
3. **RTOS-Based Embedded System Project:** Develop an embedded system project using a real-time operating system kernel written in assembly language. For example, build a smart thermostat system that monitors temperature, controls heating/cooling equipment, and provides a user interface through an LCD display and buttons.
4. **Custom Instruction Set Architecture (ISA) Simulator:** Design and implement a simulator for a custom instruction set architecture (ISA) in assembly language. Define an instruction set, memory model, and execution environment, and create a simulator that can execute programs written in the custom assembly language and provide insights into performance characteristics.
5. **Low-Level Networking Protocol Stack:** Implement a low-level networking protocol stack in assembly language, including layers for Ethernet, IP, TCP, and UDP. Develop drivers for network interface controllers and implement the necessary protocols for packet transmission, reception, and routing.
6. **Game Engine Development:** Build a simple game engine in assembly language that supports sprite-based graphics rendering, collision detection, input handling, and game logic. Create a demo game (e.g., a platformer or shoot 'em up) using the game engine to showcase its capabilities.
7. **Cryptographic Library:** Develop a cryptographic library in assembly language that provides implementations of various cryptographic algorithms, including symmetric encryption (e.g., AES), asymmetric encryption (e.g., RSA), hash functions (e.g., SHA-256), and digital signatures (e.g., ECDSA). Ensure the library is optimized for performance and security.
8. **High-Performance Computing (HPC) Simulation:** Create a simulation of a high-performance computing (HPC) system in assembly language, including multiple processing units (e.g., CPU cores or GPU shaders), memory hierarchy (registers, caches, main memory), and interconnect architecture (e.g., shared memory or message passing). Implement parallel algorithms and benchmarks to evaluate the performance of the simulated HPC system.
9. **Compiler Backend Optimization:** Extend an existing compiler backend (e.g., LLVM or GCC) by implementing new optimization passes in assembly language. Focus on optimizations such as loop unrolling, instruction scheduling, register allocation, and vectorization to improve the performance of generated machine code.
10. **Embedded System Firmware Development**: Develop firmware for an embedded system application (e.g., IoT device or robotics platform) using assembly language. Implement device drivers for sensors, actuators, and communication interfaces, and integrate them into a cohesive firmware that meets real-time performance and resource constraints.
11. **Gate Simulator:** Develop a simulator that models the behavior of basic logic gates (AND, OR, NOT, XOR, etc.). Allow users to create and connect gates to build complex digital circuits, and simulate their behavior.
12. **Arithmetic Logic Unit (ALU) Design:** Design and simulate an ALU that can perform various arithmetic and logic operations (addition, subtraction, AND, OR, etc.) on binary numbers.
13. **File Management System:** Create an assembly language program that implements a simple file management system. Include functionalities like file creation, reading, writing, deletion, and directory management.
14. **String Manipulation Library:** Implement a library of assembly language routines that perform operations on strings, such as concatenation, comparison, searching, and sorting.
15. **Binary Adder Design:** Design and simulate a binary adder circuit that can add two binary numbers. Implement it in assembly language and demonstrate its functionality with test cases.
16. **Multi-Purpose Utility Application**: Binary Converter, Arithmetic Calculator, Logical Operator, and Student Registration System with Real-Time System Time Calculation.
17. **File Encryption/Decryption:** Develop an assembly language program that can encrypt and decrypt files using symmetric encryption algorithms like XOR or Caesar cipher.
18. **Arithmetic Operations Library:** Create an assembly language library that provides routines for performing arithmetic operations on numbers, such as addition, subtraction, multiplication, and division.
19. **File Compression/Decompression:** Implement an assembly language program that can compress and decompress files using algorithms like Huffman coding or Run-Length Encoding (RLE).
20. **String Searching Algorithms:** Implement string searching algorithms (e.g., Naive, Knuth-Morris-Pratt) in assembly language to search for a pattern within a text.
21. **Binary Calculator:** Develop an assembly language program that acts as a calculator, allowing users to perform arithmetic operations on binary numbers (addition, subtraction, multiplication, division).
22. **File Sorting:** Create an assembly language program that can sort the contents of a file using sorting algorithms like Bubble Sort, Insertion Sort, or Quick Sort.
23. **String Encryption/Decryption:** Design an assembly language program that encrypts and decrypts strings using symmetric encryption algorithms like XOR or Transposition cipher.
24. **Logic Gate Circuit Optimizer:** Develop an assembly language program that optimizes a given logic gate circuit by minimizing the number of gates or reducing propagation delay.
25. **File Comparison:** Implement an assembly language program that compares the contents of two files and determines if they are identical or different.