Implementation Summary

## Implemented Features

In this project, we implemented key system-level c functions using assembly. Our focus was on startup routines, memory management, system calls, and timers, ensuring efficient low-level control. Here’s a breakdown of what we built:

### Startup Code (startup\_tm4c129.s)

* **Reset\_Handler**: Handles the system reset event and initializes the processor state.
* **SVC\_Handler**: Manages system service calls (supervisor calls).
* **SysTick\_Handler**: Implements the system tick interrupt handler for periodic tasks.

### Standard Library Functions (stdlib.s)

* **\_bzero( )**: Clears memory by setting bytes to zero.
* **\_strncpy( )**: Copies a specified number of characters from one string to another.
* **\_malloc( )**: Allocates memory dynamically from the heap.
* **\_free( )**: Releases allocated memory back to the heap.
* **\_alarm( )**: Implements a timer-based function to trigger events after a set duration.
* **\_signal( )**: Handles software signals and asynchronous event notifications.

### System Calls (svc.s)

* **\_systemcall\_table\_init( )**: Initializes the system call table with function pointers.
* **\_systemcall\_table\_jump( )**: Handles system call execution by jumping to the appropriate function.

### Heap Management (heap.s)

* **\_heap\_init( )**: Initializes the heap and prepares memory for allocation.
* **\_kalloc( )**: Allocates memory from the kernel heap.
* **\_kfree( )**: Frees memory allocated by \_kalloc( ).

### Timer Implementation (timer.s)

* **\_timer\_init( )**: Sets up the hardware timer for time-based events.
* **\_timer\_start( )**: Starts a timer with a given duration.
* **\_timer\_update( )**: Updates the timer state and manages expired timers.
* **\_signal\_handler( )**: Handles timer-generated signals and executes callback functions.

### Missing Components

There were no missing features in this implementation.

### Diagram

A diagram of a computer program

AI-generated content may be incorrect.

This diagram showcases how different components interact with each other in the implementation.