Name :Eslam Ahmed Ali

Day: 02:07_08_2023

Group : ITI . Embedded Adv. Al-Azhar G4 23

Task : 1

1- Explain context switching in ARM cortex M4?

context switching process :

Context switching is a fundamental operation in multitasking operating systems that allows multiple tasks or processes to share a single CPU or processor core. It involves saving the state of a currently running task, known as the "current context," and restoring the saved state of another task, known as the "next context," to continue its execution.

When a context switch occurs, the following steps are typically involved:

1. Saving the current context: The operating system or kernel saves the state of the currently running task, including the values of CPU registers, program counter, stack pointer, and other relevant information. This step ensures that the task's execution can be resumed later from where it left off.

2. Loading the next context: The operating system selects the next task to run based on its scheduling algorithm. It retrieves the saved context of the selected task and restores the state of CPU registers, program counter, stack pointer, and other necessary information from the saved context.

3. Switching memory spaces: In some cases, the context switch may involve switching between different memory spaces or address spaces. For example, when switching between different processes, the virtual memory mappings may need to be updated to reflect the memory layout of the new process.

4. Updating data structures: The operating system updates various data structures to reflect the change in the current running task. This includes updating the task control blocks, scheduling queues, and other bookkeeping data structures used by the operating system for managing tasks.

5. Resuming execution: Once the next context is loaded and the necessary data structures are updated, the processor starts executing instructions from the restored program counter of the new task. The execution continues from the point where the task was preempted during the previous context switch.

context switching process in arm cortext M4:

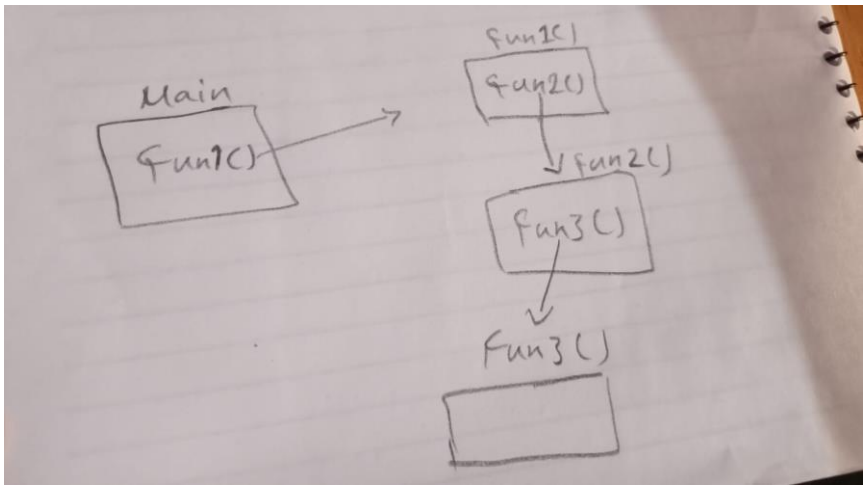
there is a register called a Link Register this register from within general purpose registers this register use to save the last line in main that CPU was executed

and cpu will jump to function which called in main

to handled it after handling the CPU will return to line was there

but if the function call another functions

(this figure will explain)



The first call (fun1()) the CPU will use the linker register and it will be fast call ,but the rest call (fun2) the cpu use stack to save the last line it execute in fun2 in varible in the Stack and jump to fun3 (this process of call is slower than first call)

2- Difference between Autosar and non-autosar projects?

AUTOSAR (Automotive Open System Architecture) is a standardized software architecture framework used in the automotive industry for developing automotive software systems. Non-AUTOSAR projects, on the other hand, refer to software development projects that do not adhere to the AUTOSAR standards. Here are some key differences between AUTOSAR and non-AUTOSAR projects:

1. Architecture and Design Approach:
   - AUTOSAR: AUTOSAR provides a standardized architecture and design approach for automotive software systems. It defines a layered software architecture, standardized interfaces, and a set of software components that can be reused across different automotive applications.
   - Non-AUTOSAR: Non-AUTOSAR projects do not follow the standardized architecture defined by AUTOSAR. They may use different architectural styles, design patterns, and frameworks based on project-specific requirements.
2. Component Reusability:
   - AUTOSAR: AUTOSAR emphasizes component-based development and encourages reusability of software components. AUTOSAR-compliant software components can be developed once and used across different automotive projects, leading to reduced development effort and improved reliability.
   - Non-AUTOSAR: Non-AUTOSAR projects may or may not emphasize component reusability. The level of component reusability depends on the specific development practices and guidelines followed within the project.
3. Communication and Networking:
   - AUTOSAR: AUTOSAR provides standardized communication and networking protocols for in-vehicle communication. It defines a layered communication stack and specifies protocols like CAN, LIN, FlexRay, and Ethernet for inter-component communication.
   - Non-AUTOSAR: In non-AUTOSAR projects, the communication and networking protocols may vary based on project-specific requirements. The choice of protocols and communication mechanisms is typically left to the project development team and may not follow a standardized approach.

4. Tooling and Ecosystem:
- AUTOSAR: AUTOSAR has a well-defined tooling ecosystem with various software development tools that support AUTOSAR-compliant development, including configuration tools, code generators, and validation tools.
- Non-AUTOSAR: Non-AUTOSAR projects may use a wide range of development tools and frameworks that are not specifically tailored for AUTOSAR. The tooling ecosystem may vary based on the preferences and choices of the development team.

5. Industry Compliance:
- AUTOSAR: AUTOSAR is widely adopted in the automotive industry, and many automotive manufacturers and suppliers follow the AUTOSAR standards. Compliance with AUTOSAR enables interoperability and compatibility between different automotive systems and components.
- Non-AUTOSAR: Non-AUTOSAR projects may or may not have a standardized approach to software development. They may follow industry-specific standards or guidelines but are not bound by the AUTOSAR framework.

3- Discuss the difference between AHB and APB?

The AHB (Advanced High-Performance Bus) and APB (Advanced Peripheral Bus) are two bus protocols commonly used in the ARM-based system-on-chip (SoC) designs. They differ in terms of their characteristics, purpose, and the type of peripherals they interface with. Here are the key differences between AHB and APB:

1. Performance and Bandwidth:
   - AHB: The AHB bus is designed for high-performance and high-bandwidth data transfers. It supports pipelining, burst transfers, and multiple data transfers per clock cycle. It is typically used to connect high-bandwidth components such as processors, memory controllers, DMA controllers, and high-speed peripherals.
   - APB: The APB bus is optimized for lower-performance peripherals and slower data transfers. It operates at a lower clock speed compared to AHB, and its design is more focused on minimizing power consumption and area utilization. APB is commonly used to interface with lower-speed peripherals such as UARTs, timers, GPIOs, and other peripherals that do not require high bandwidth.

2. Bus Width:
   - AHB: The AHB bus supports both 32-bit and 64-bit data transfers. It allows for efficient transfer of large amounts of data in a single transaction.
   - APB: The APB bus primarily supports 8-bit and 16-bit data transfers. It is suitable for peripherals with lower data transfer requirements.

3. Arbitration:
   - AHB: The AHB bus uses a priority-based arbitration scheme, where multiple masters contend for bus access based on their assigned priorities. AHB supports multiple masters, allowing for concurrent access to the bus.
   - APB: The APB bus typically uses a simple round-robin arbitration scheme, where each master is granted access to the bus in a sequential order. APB generally supports a single master, allowing only one master to access the bus at a time.

4. Power Consumption:
- AHB: The AHB bus is designed for higher performance and typically consumes more power due to its support for faster and wider data transfers.
- APB: The APB bus is optimized for lower power consumption, making it suitable for power-sensitive designs and peripherals with lower performance requirements.

5. Usage Scenarios:
- AHB: The AHB bus is commonly used to connect high-performance components such as CPUs, memory controllers, and high-speed peripherals that require high bandwidth and efficient data transfers.
- APB: The APB bus is typically used to interface with lower-speed peripherals and control registers of various on-chip peripherals. It is commonly employed for serial communication interfaces, timers, GPIOs, and other peripherals with lower bandwidth requirements.