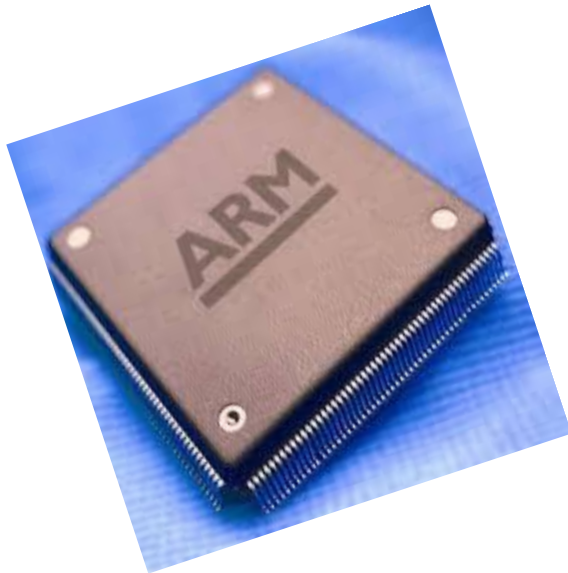


MICROPROCESSOR &

COMPUTER ARCHITECTURE UE17CS253



UNIT-4

Session – 1

Interrupts & Interrupt Handling Mechanism

SOFTWARE METHOD – POLLING

- In this method, all interrupts are serviced by branching to the same service program.
- This program then checks with each device if it is the one generating the interrupt.
- The order of checking is determined by the priority that has to be set.
- The device having the highest priority is checked first and then devices are checked in descending order of priority.
- If the device is checked to be generating the interrupt, another service program is called which works specifically for that particular device.

The structure will look something like this-

```
if (device[0].flag) device[0].service();  
    else if (device[1].flag) device[1].service();  
    .....  
    else //raise error
```



- The daisy-chaining method involves connecting all the devices that can request an interrupt in a serial manner.
- This configuration is governed by the priority of the devices.
- The device with the highest priority is placed first followed by the second highest priority device and so on.

WHAT IS INTERRUPT/EXCEPTION?

- Main ()

- {

- :

Can happen anytime

Depends on types of interrupts

- Doing something

- (e.g.

- browsing)

- :


- } ring



Phone rings

Phone rings



```
_isr() // Interrupt service routine
{
    
    some tasks (e.g. answer
        telephone)

} //when finished,
  //goes back to main
```

- Interrupts
 - Reset
 - (or power up)

- Software
 - Interrupt
 - SWI-XX

- Hardware
 - Interrupt
 - FIQ,IRQ
 - Timer

Triggered by
power_up/
reset_key

Triggered by the
software instruction
SWI x

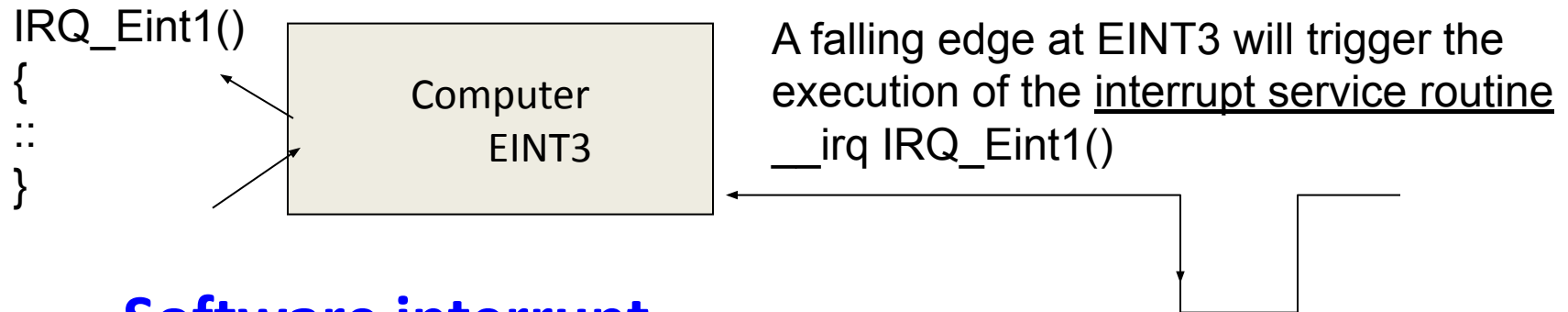
- ADC
- External
- Interrupt
- EINT

Triggered by
hardware sources

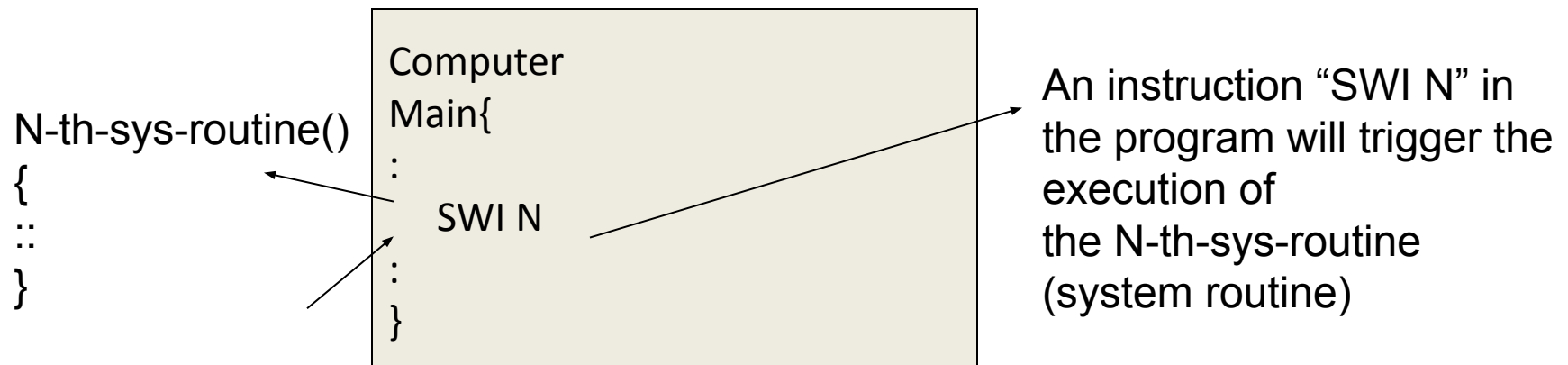
- The terms are used differently by various manufacturers
- Traditionally exception means
 - **The normal operation of a program is interrupted and the processor will execute another piece of software (exception handling) somewhere.**
 - Interrupt (hardware interrupt) is an exception caused by some hardware condition happening outside the processor (e.g. external hard interrupt, IRQ FIQ).
 - Software interrupt (SWI) is an exception caused by an assembly software instruction (SWI 0x?? exception call instruction) written in the software code.
 - Trap is an exception caused by a failure condition of the processor (e.g. abort “pre-fetch , data” , undefined instruction, divided_by_zero, or stack overflow etc)

- Reset, a special interrupt to start the system— happens at power up , or reset button depressed)
- Software interrupt SWI :
Similar to subroutine – happens when “SWI 0x??” is written in the program
- Hardware interrupt
 - FIQ (fast interrupt) or IRQ (external interrupt), when
 - the external interrupt request pin is pulled low, or
 - an analogue to digital conversion is completed, or
 - A timer/counter has made a regular request

– Hardware interrupt, e.g.



– Software interrupt

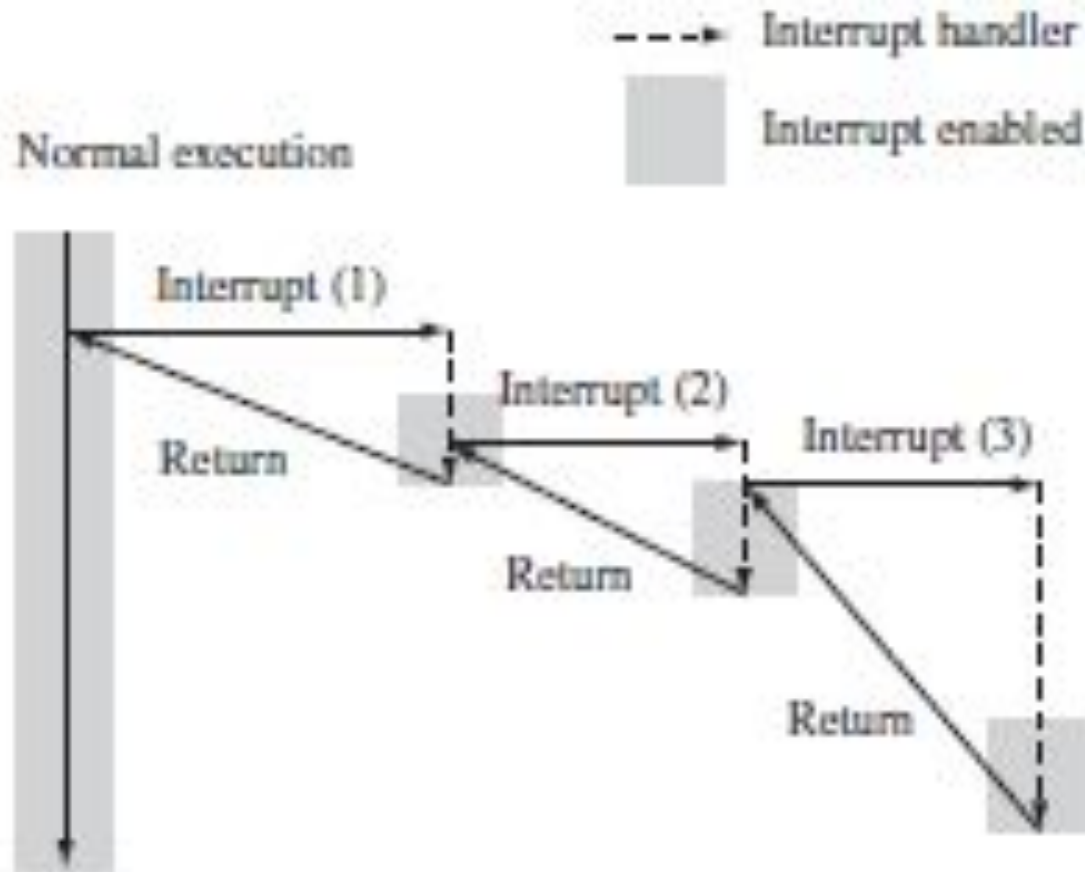


EXCEPTION (INTERRUPT) MODES

- ARM supports 7 types of exceptions and has a privileged processor mode for each type of exception.
- ARM Exception (interrupt) vectors
- Interrupt Vector Table

	Address	Exception	Mode in Entry
1	0x00000000	Reset	Supervisor
2	0x00000004	Undefined instruction	Undefined
3	0x00000008	Software Interrupt	Supervisor
4	0x0000000C	Abort (prefetch)	Abort
5	0x00000010	Abort (data)	Abort
x	0x00000014	Reserved	Reserved
6	0x00000018	IRQ (external interrupt)	IRQ
7	0x0000001C	FIQ (fast interrupt)	FIQ

EXCEPTION (INTERRUPT) MODES



Nested Interrupt call

- Interrupt Service Routine
- Status Registers
- Saving register contents on to the stack.
- Restore while return to the interrupted program.

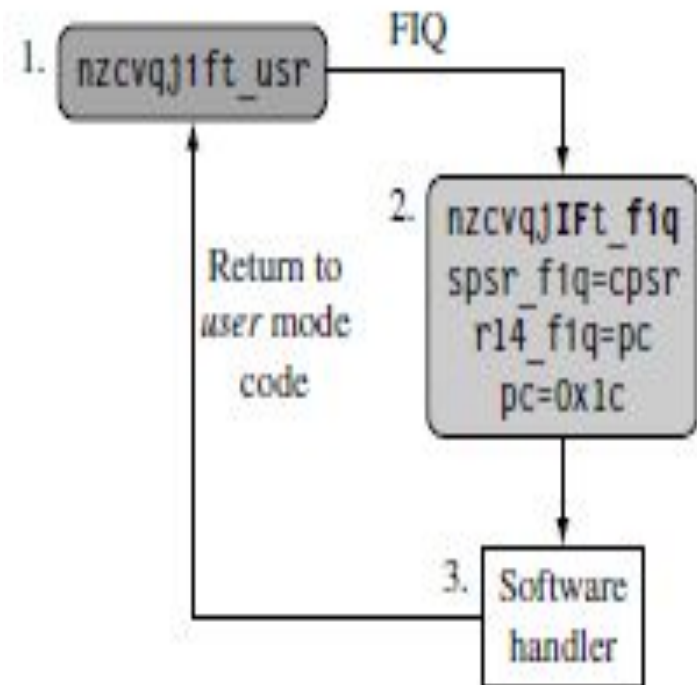
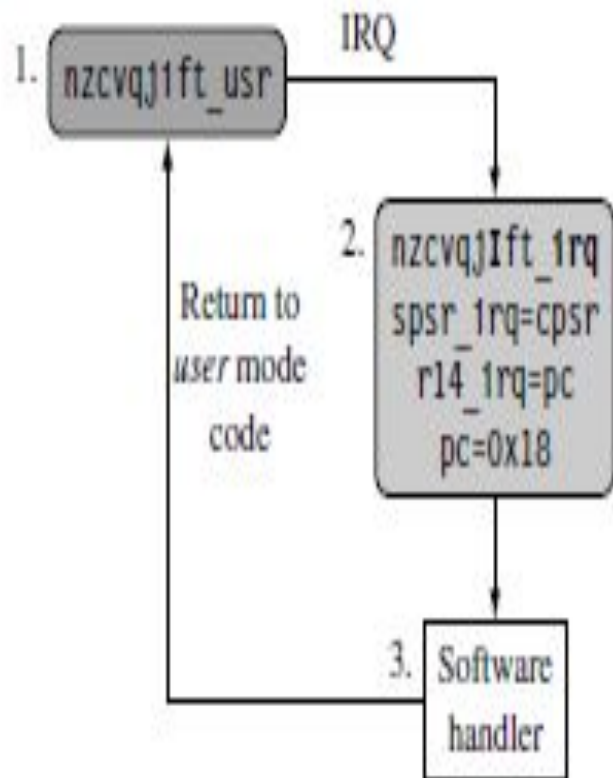
The second method involves *prioritization*.

- You program the *interrupt controller* to ignore interrupts of the same or lower priority than the interrupt you are handling, so only a higher-priority task can interrupt your handler.
- You then re-enable interrupts. The processor spends time in the lower-priority interrupts until a higher-priority interrupt occurs.
- Therefore higher-priority interrupts have a lower average interrupt latency than the lower-priority interrupts, which reduces latency by speeding up the completion time on the critical time-sensitive interrupts.

IRQ and FIQ Exceptions

- IRQ and FIQ exceptions only occur when a specific interrupt mask is cleared in the *cpsr*
- An IRQ or FIQ exception causes the processor hardware to go through a standard procedure (provided the interrupts are not masked):

1. The processor changes to a specific interrupt request mode, which reflects the interrupt being raised.
2. The previous mode's *cpsr* is saved into the *spsr* of the new interrupt request mode.
3. The *pc* is saved in the *lr* of the new interrupt request mode.
4. *Interrupt/s are disabled—either the IRQ or both IRQ and FIQ exceptions are disabled in the cpsr. This immediately stops another interrupt request of the same type being raised.*
5. The processor branches to a specific entry in the vector table.



Enabling and Disabling FIQ and IRQ Exceptions

Enabling an interrupt.

<i>cpsr</i> value	IRQ	FIQ
Pre	<i>nzcvqjIfT_SVC</i>	<i>nzcvqjIfT_SVC</i>
Code	<i>enable_irq</i>	<i>enable_fiq</i>
	MRS r1, cpsr	MRS r1, cpsr
	BIC r1, r1, #0x80	BIC r1, r1, #0x40
	MSR cpsr_c, r1	MSR cpsr_c, r1
Post	<i>nzcvqjiFt_SVC</i>	<i>nzcvqjiFt_SVC</i>

Disabling an interrupt.

<i>cpsr</i>	IRQ	FIQ
Pre	<i>nzcvqjift_SVC</i>	<i>nzcvqjift_SVC</i>
Code	<i>disable_irq</i>	<i>disable_fiq</i>
	MRS r1, cpsr	MRS r1, cpsr
	ORR r1, r1, #0x80	ORR r1, r1, #0x40
	MSR cpsr_c, r1	MSR cpsr_c, r1
Post	<i>nzcvqjiFt_SVC</i>	<i>nzcvqjiFt_SVC</i>

Basic Interrupt Stack Design and Implementation

Exceptions handlers make extensive use of stacks, with each mode having a dedicated register containing the stack pointer. The design of the exception stacks depends upon these factors:

- *Operating system requirements—Each operating system has its own requirements for stack design.*

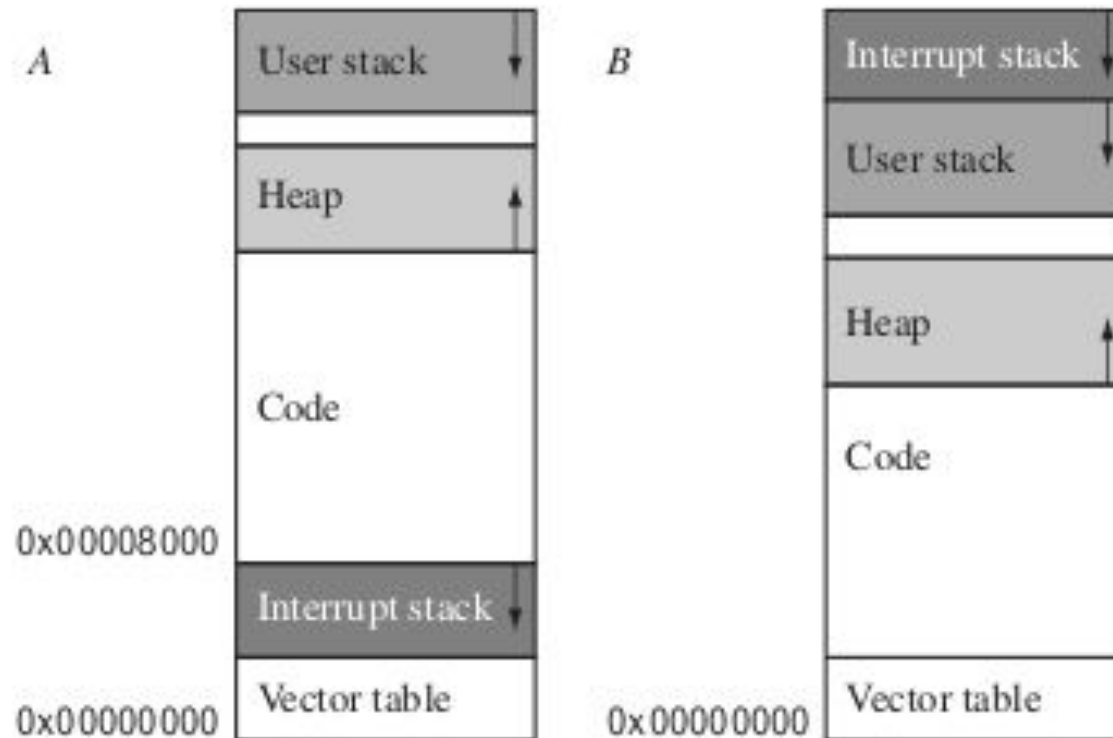
- *Target hardware—The target hardware provides a physical limit to the size and positioning of the stack in memory.*

Two design decisions need to be made for the stacks:

- The *location determines where in the memory map the stack begins. Most ARM-based systems are designed with a stack that descends downwards, with the top of the stack at a high memory address.*
- *Stack size depends upon the type of handler, nested or non nested. A nested interrupt handler requires more memory space since the stack will grow with the number of nested interrupts.*

Basic Interrupt Stack Design and Implementation

Stack Layouts



Basic Interrupt Stack Design and Implementation

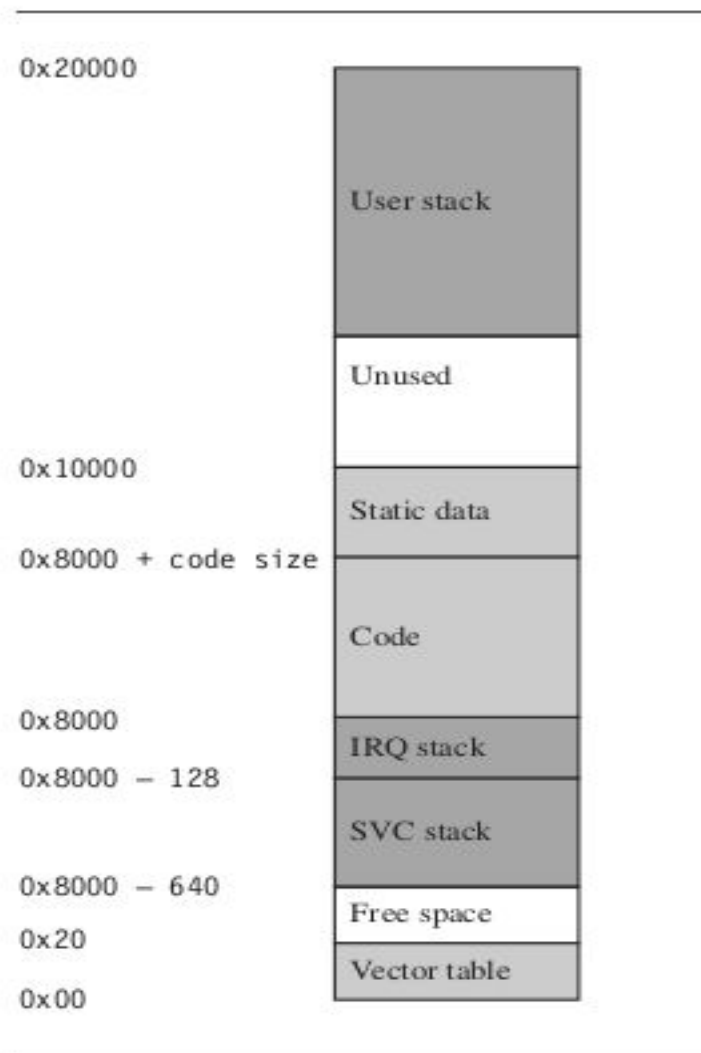


Figure 9.7 Example implementation using layout A.

Interrupt Handling Schemes

- Non-Nested Interrupt Handler
- Nested Interrupt Handler
- Re-Entrant Interrupt Handler
- Prioritized Simple Interrupt Handler
- Prioritized Standard Interrupt Handler
- Prioritized Direct Interrupt Handler
- Prioritized Grouped Interrupt Handler
- VIC PL190 based ISR

Interrupt Handling Schemes Non-Nested Interrupt Handler

- Simplest:
- Interrupts are disabled till control is returned back to program which was being executed
- Services single Interrupts at a time
- Not suitable for complex embedded systems

Various stages in a NNIH –

Disable Interrupts

IRQ raised → Interrupts disabled, no further Interrupts → processor mode changed → cpsr saved in spsr_{interrupt request mode} → pc is saved → pc points to VT → VT points to Interrupt handler

Save context

Handler saves a subset of non-banked registers

Various stages in a NNIH –

Interrupt Handler

Handler identifies interrupt source and executes appropriate ISR

Interrupt Service Routine

ISR Services the interrupt source and resets the interrupt

Restore Context

ISR returns back to handler which restores the saved context

Various stages in a NNIH –

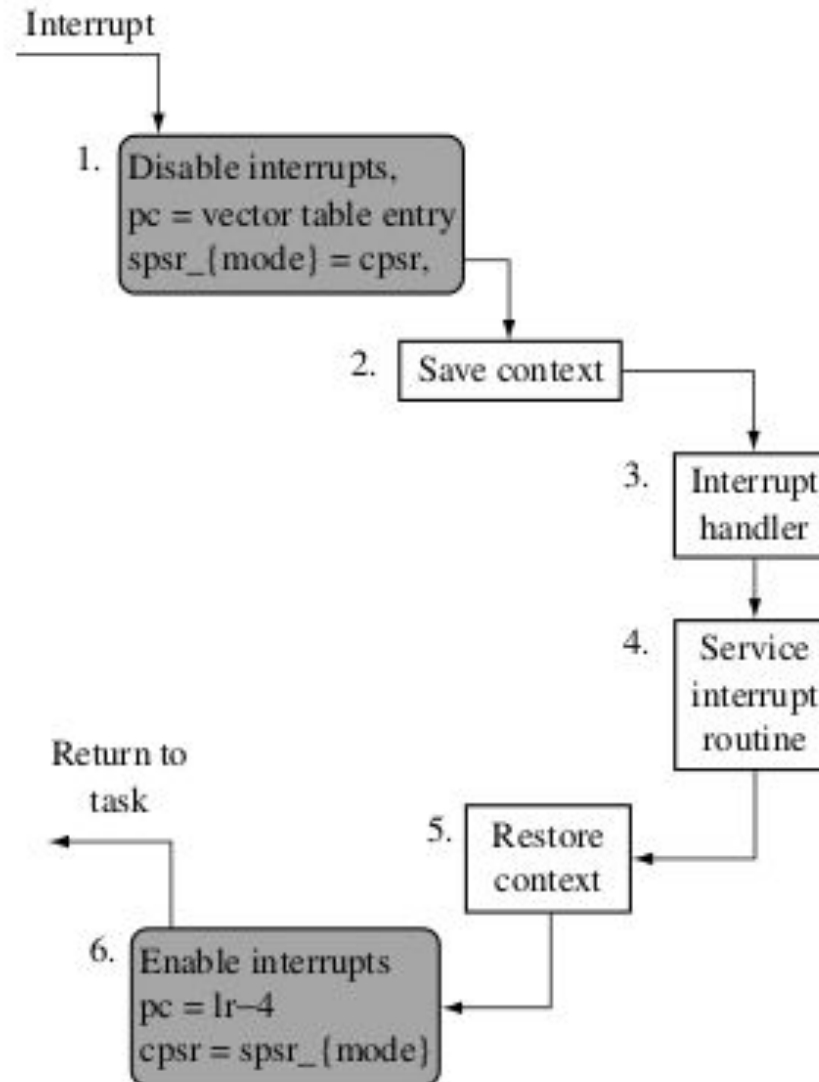
Enable Interrupts

$\text{spsr} \leftarrow \text{spsr}_{\{\text{interrupt request mode}\}}$

pc is restored

```
interrupt_handler
    SUB    r14,r14,#4                ; adjust lr
    STMFD  r13!,{r0-r3,r12,r14}     ; save context
    <interrupt service routine>
    LDMFD  r13!,{r0-r3,r12,pc}^     ; return
```


Interrupt Handling Schemes Non-Nested Interrupt Handler



Priority Interrupts

- A priority interrupt is a system which decides the priority at which various devices, which generates the interrupt signal at the same time, will be serviced by the CPU.
- The system has authority to decide which conditions are allowed to interrupt the CPU, while some other interrupt is being serviced.
- Generally, devices with high speed transfer such as *magnetic disks* are given high priority and slow devices such as *keyboards* are given low priority.
- When two or more devices interrupt the computer simultaneously, the computer services the device with the higher priority first.

Programmable Interrupt Controller

- 8 levels of Interrupts
- Can be cascaded in master- slave configuration to handle 64 levels of interrupts.
- Internal Priority Resolver
 - Fixed Priority Mode & Rotating Priority Mode
 - Individually Maskable Interrupts
 - Modes and masks can be changed dynamically
 - Accepts IRQ, determines priority, checks whether incoming priority current level being serviced, issues interrupt signal.
 - Polled and vectored mode
 - Starting address of ISR or Vectored number is programmable.

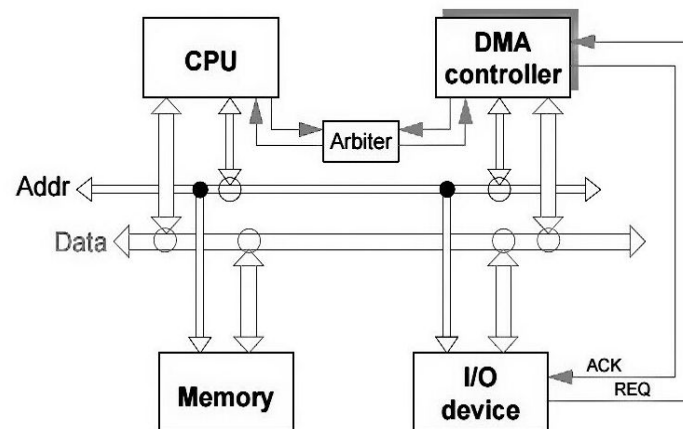
Direct Memory Access(DMA) Controller

What is actually DMA?

- The direct memory access is a system where the samples are saved in the memory of the system while the processor does something else.

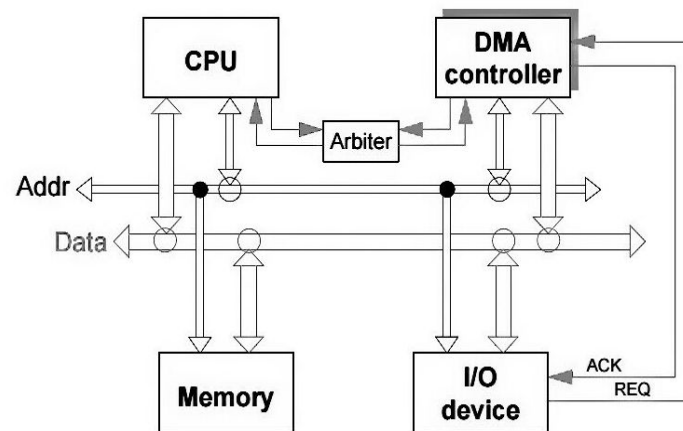
Why we need DMA?

- The assumption about the I/O machines like keyboards, mouse, and printer etc. are genuinely very slow when compared with the central processing unit (CPU).
- To overcome this issue an interrupt handler was used and the I/O machine produces all the signals that the CPU produce, then the I/O machine can bypass the information alienates to central processing unit and hence increases the speed.



Direct Memory Access(DMA) Controller

- 罅 **Cycle Stealing Process:** Traditionally it is a method of accessing RAM or bus without interfering with the CPU.
- 罅 DMA allows I/O controllers to read or write RAM without CPU intervention.
- 罅 Clever exploitation of specific CPU or bus timings can permit the CPU to run at full speed without any delay if external devices access memory not actively participating in the CPU's current activity and complete the operations before any possible CPU conflict.
- 罅 Such systems are nearly dual-port RAM without the expense of high speed RAM.
- 罅 Most systems halt the CPU during the **steal**, essentially making it a form of DMA by another name



AMBA (ADVANCED MICROPROCESSOR BUS ARCHITECTURE) PROTOCOL

The Arm AMBA protocols are an open standard, on-chip interconnect specification for the connection and management of functional blocks in a System-on-Chip (SoC).

It facilitates right-first-time development of multi-processor designs with large numbers of controllers and peripherals.

Features of AMBA Interfaces:

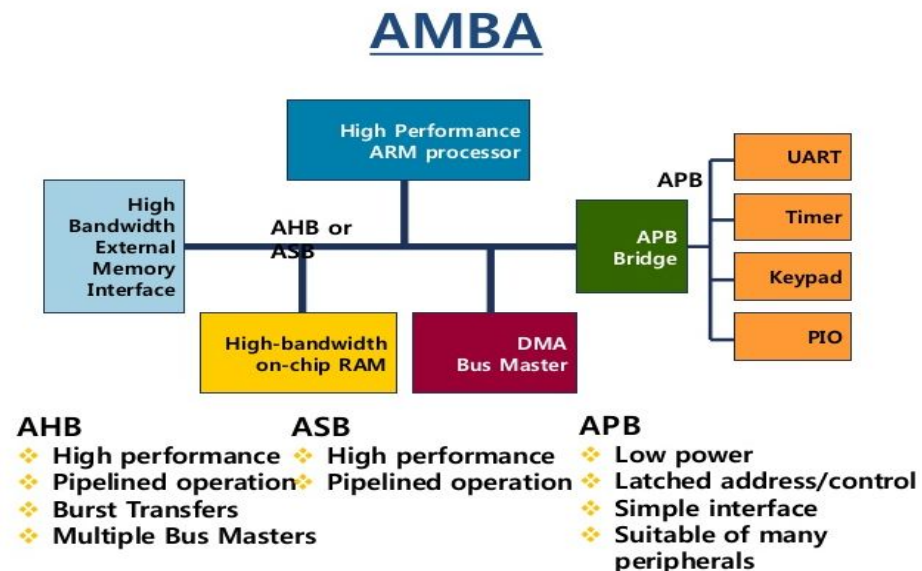
IP reuse is essential in reducing SoC development costs and timescales. AMBA specifications provide interface standards that enables IP reuse if the following essential requirements are met:

Flexibility:

Wide Adaption:

Compatibility:

Support:



APB (ADVANCED PERIPHERAL BUS) PROTOCOL

- The Advanced Peripheral Bus (APB) is part of the Advanced Microcontroller Bus Architecture (AMBA) protocol family.
- It defines a low-cost interface that is optimized for minimal power consumption and reduced interface complexity.
- The APB protocol is not pipelined, use it to connect to low-bandwidth peripherals that do not require the high performance of the AXI protocol.
- The APB protocol relates a signal transition to the rising edge of the clock, to simplify the integration of APB peripherals into any design flow.
- Every transfer takes at least two cycles.
- The APB can interface with:
 - AMBA Advanced High-performance Bus (AHB)
 - AMBA Advanced High-performance Bus Lite (AHB-Lite)
 - AMBA Advanced Extensible Interface (AXI)
 - AMBA Advanced Extensible Interface Lite (AXI4-Lite)

Q & A

on Interrupts , Interrupt Handling Mechanism

PIC & DMA Controller