MODULE -I ARM Embedded Systems / ARM Processor Fundamental

1.1. Microprocessor Vesus Microcontrollers.

1.2. The RISC Design philosophy.

→ Instruction Set for Embedded System.

1.3 ARM Design philosophy

1.4. Embedded System Hardware.

→ ARM Bus Technology

→ AMBA Bus protocol.

→ Memory

→ Peripherals.

1.5. Embedded System Software

→ Initialization (Boot code)

→ Operating System.

→ Applications.

Chapter-1

ARM Embedded Systems

1.6. Registers.

1.7. Current Program Status Register.

→ Processor Modes

→ Banked Registers

→ State & Instruction set.

→ Interrupt Masks

→ Condition Flags.

→ Condition Execution

1.8 Pipeline → pipelined Executing Characteristics.

1.9 Exceptions, Interrupts and Vector Table.

1.10 Core extensions.

→ Cache and Tightly coupled Memory.

→ Memory Management.

→ Co processors.

Chapter 2

ARM Process Fundame

Introduction and Overview of the Course

An Embedded system comprises of an embedded hardware and Software designed to suffice a specific requirement. An embedded system developers and system on-chip designers select specific microprocessor cores and family of tools, Libraries and off the shelf components to develop new microprocessor-based products. ARM plays a major role in an embedded System. ARM architecture has become the most widely used 32 bit architecture over the past decade. ARM processors are embedded in products ranging from cell phones to automative braking systems.

This course aims in describing the operation of ARM core from a product developer's perspective with a clear emphasis on software. Clear understanding of ARM architecture is very important for embedded systems development. ARM processor design philosophy is discussed and it's difference from a traditional RISC philosophy is elaborated. A simple embedded system based on ARM processor is described. Detailed hardware details of ARM processor is discussed. ARM instruction set is discussed which forms the fundamental basis for writing an effient ARM code.

1.1 Microprocessor Versus Microcontrollers :-

Microprocessor	Microcontrollers
① CPU is a stand alone, RAM ROM, I/O and timer are Separate	① CPU, RAM, ROM, I/O and timer all on Single chip.
② Designer can decide the amount of RAM, ROM and Input-output ports to be interfaced.	② Fixed amount of on chip RAM, ROM and I/O ports.
③ Expensive.	③ Suitable for applications in which cost, power and space are critical.
④ General Purpose.	④ Specific purpose (Embedded System)
⑤ High power Consumption.	⑤ Low power consumption.
⑥ Deep pipelining	⑥ Single cycle / Two stage pipeline.
⑦ High clock rate	⑦ Low clock speed.

1.2. The RISC Design philosophy :-

ARM stands for Advanced Reduced Instruction Set Computing ARM Core uses a RISC architecture.

Reduced Instruction Set Computing Architecture (RISC) uses simple but powerful instructions that execute with in a single cycle at high clock speed.

RISC Versus CISC :-

Complex Instruction Set Computer	Reduced Instruction Set Computer
① Emphasis on Hardware	Emphasis on software.
② Multiple Instruction sizes & formats	Instructions of same size with few formats.
③ Less Registers.	Uses more registers.
④ More addressing Modes	Fewer Addressing modes.
⑤ Extensive use of Microprogramming (Complexity in Processor)	Complexity in Compiler.
⑥ Instructions take varying amount of cycle time.	Instructions take one cycle time.
⑦ Pipelining is Difficult.	Pipelining is easy.
Compiler → Code Generation → Processor (Greater Complexity)	Compiler (Greater Complexity.) → Code Generation. → Processor.
⑧ Any Instruction may reference Memory.	Only LOAD/STORE references the Memory.

The RISC philosophy is implemented with four major design roles :-

(a) Instructions :-

RISC processors have reduced numbe of instruction classes. These set provide simple operations that can executed in a single cycle. The compiler or programmer synthesizes complicated operations by combining several simple instructions. Each instruction is of fixed length thus allowing the pipeline to fetch the next instruction before decoding the current instruction.

CISC processors have instructions which are often of variable size and take many cycles to execute thus making pipelining difficult.

(b) Pipelines :-

Pipeline is an implementation technique whae multiple instructions are overlapped in execution by breaking down the instruction into smallest units that can be executed in parallel. Each step in pipeline advances by one step on each cycle to maximize the through put.

t	IF	DE	EX
1	i_1		
2	i_2	i_1	
3	i_3	i_2	i_1

In CISC, instructions are executed by a miniprogram called microcode.

© Registers:

RISC machines have large set of general purpose registers. Registers act as fast local memory store for all data processing operations. Registers can store either data or an address.

CISC processors have fewer dedicated registers for specific purpose.

(d) Load/Store Architecture:-

Separate LOAD and STORE instructions are used in RISC machines to transfer data between register bank and external memory. Separating memory access from data processing has an advantage because data items held in the register bank can be accessed multiple times without needing multiple memory accesses which is costly.

CISC Machines have instructions that can act directly on memory.

The above design rules infer that RISC processor are simpler and can operate at higher clock frequencies. CISC processors are more complex and operate at lower clock frequencies. However the distinction between RISC and CISC has blurred and CISC processors have implemented more RISC concepts.

1-4. The ARM Design philosophy.

The primary application of an ARM processor is the embedded system (which includes mobile phones and Personal Digital Assistants). The physical features of ARM processor design includes the following:

(a) The ARM processor has been designed to be small so as to reduce power consumption and extend battery operation essential for applications such as mobile phones and personal digital Assistants.

(b) High Code Density :. Embedded systems have limited memory due to cost and physical size restrictions. High code density is useful for applications with limited on board memory. Code density is the amount of space an executable program takes up in a memory.

(c) Ability to use Low cost Memory devices as embedded systems are price sensitive.

(d) Area taken by the embedded processor on Die is reduced so as to accomodate more space for specialized peripherals. This intern reduces the cost of design and manufacturing as fewer discrete chips are required for the end product.

(e) ARM has incorporated hardware Debug Technology with in Processor so that software engineers can view what is happening while processor is executing code. This allows issues to be resolved

faster, reducing the time to market and overall development costs.

Thus ARM is not a pure RISC architecture because of its primary application constraint as Embedded system.

Instruction set for Embedded Systems :-

ARM stands for Advanced Reduced Instruction Set Computing machine. Several features are improved over RISC to make it suitable for embedded applications which includes the following :-

(a) Variable cycle Execution for certain Instructions :-

Not all ARM instruction execute in a single cycle.

for eg :- LOAD / STORE instruction used to transfer data between memory and registers. Thus the number of execution cycles depend upon the number of register being transfered.

The transfer could be on sequential memory addresses or random memory. Performance on sequential memory access are faster than random memory access.

Code density is also improved since multiple register transfers are common at the start and end of fonctions.

(b) In Line Barrel Shifter Leading to more complex Instructions :-

The inline Barrel shifter is a hardware component that preprocesses one of the input register before it is used by an instruction. This expands the capability of many instructions to improve core performance and code density.

(c) Thumb 16 bit Instruction Set :-

Thumb is a second 16 bit instruction set that is added to ARM permitting ARM core to execute either 16 or 32 bit instructions. The 16 bit instructions improve code density by about 30% over fixed 32 bit length instructions.

(d) Conditional Execution :-

Conditional Execution allows execution of an instruction only when a specified condition has been met. This feature improves performance and code density by reducing branch instructions.

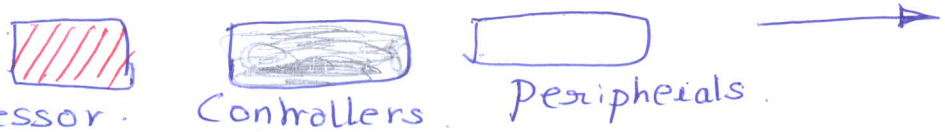
(e) Enhanced Instructions :-

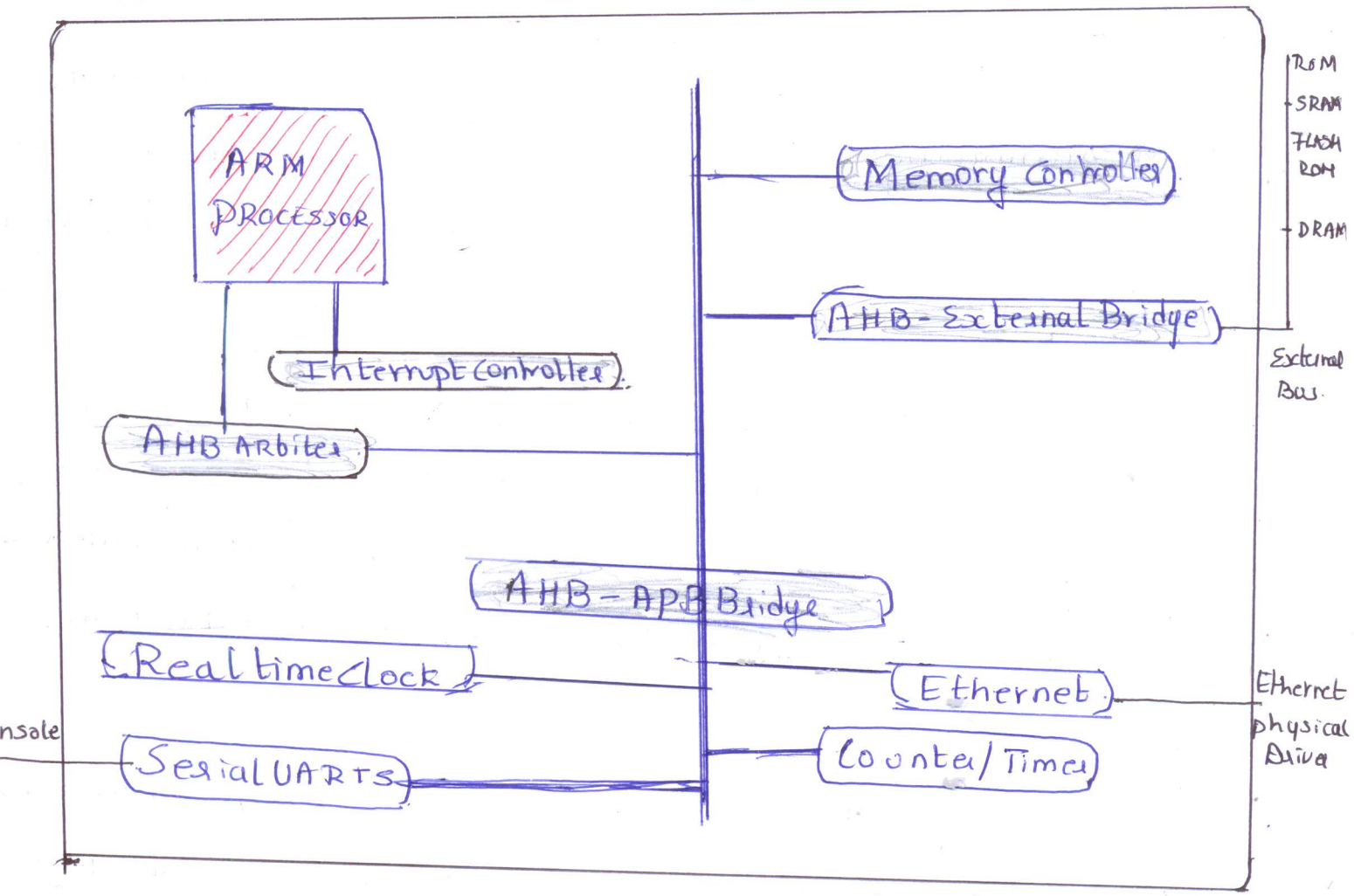
Enhanced Digital Signal processor (DSP) instructions were added to the standard ARM set which support 16 x 16 multiplier operations and saturation. Thus fast performing ARM processor can be used to replace traditional combination of processor plus a DSP.

1-5. Embedded System Hardware :-

Embedded System is a combination of hardware and Software components. Each component is chosen for efficiency and may be designed for future extension and expansion.

The block diagram of a typical embedded device based on ARM core is as shown in the figure,





AHB → ARM high performance Bus.

APB → ARM peripheral Bus.

Each block in the diagram represents a function or feature. The various fonctional blocks are connected by means of buses carrying data.

The four major hardware components used in ARM based embedded system includes – ARM processor, Controllers, peripherals and Buses.

ARM Processor :- The ARM processor controls the embedded device. Different versions of ARM processors are available to suit the particular application. An ARM processor comprises of a core and surrounding components. Core is the execution engine that processes instruction and manipulates data and

Surrounding components interface ~~with~~ core with bus. These components may include memory management & caches.

Controllers :- Controllers coordinate important fonchonal blocks of the system. Too common controllers include Interrupt controller & Memory controlla.

Peripherals :- Provide all input-output capability external to the chip and responsible for the uniqueness of embedded system.

Bus :- A bus is used to communicate between different parts of the embedded device. The above block diagram has three buses → AHB Bus (ARM High performance Bus) for the high performance peripherals.

→ APB (ARM peripheral Bus) for slower peripherals.

→ Third Bus → for external peripherals.

ARM Bus Technology :-

8086 uses Peripheral Component Interconnect (PCI) bus which connects several devices such as video cards and hard disk controllers to x86 processor bus. This type of technology is external or offchip and is build into a the motherboard of a pc. offchip Bus is designed to connect mechanically and electrically to devices external to chip.

Embedded Devices uses on chip bus that is internal to the chip and allows different peripherals to be interconnected within an ARM core.

Two classes of devices attached to bus are Bus Master and Bus slaves. Bus master is a logical device capable of initiating a data transfer with another device across the bus.

Bus slaves are Logical devices capable of responding to a transfer request from a bus master device.

Buses have two architecture Levels → Physical Level
→ Protocols.

Physical Level covers physical characteristics and bus width (16, 32 or 64 bits).

Protocols are logical rules that govern the communication between processor and a peripherals.

ARM being a design company seldom implements electrical characteristics but specifies the bus protocol routinely.

AMBA Bus Protocol:-

Advanced Microcontroller Bus Architecture (AMBA) is a widely adopted as the on chip bus architecture used for ARM processor. ARM peripheral Bus (APB) and ARM system Bus (ASM) were introduced first by AMBA. Later it introduced ARM high performance Bus (AHB)

AHB provides higher data throughput than ASB as it uses centralized multiplexed bus scheme unlike ASB bidirectional bus design. This makes AHB bus to run at higher clock rate and supports widths of 64 bits and 128 bits.

ARM has introduced two variations on AHB

→ Multilayer AHB

→ AHB Lite.

Multilayer AHB allows multiple active Bus Master.

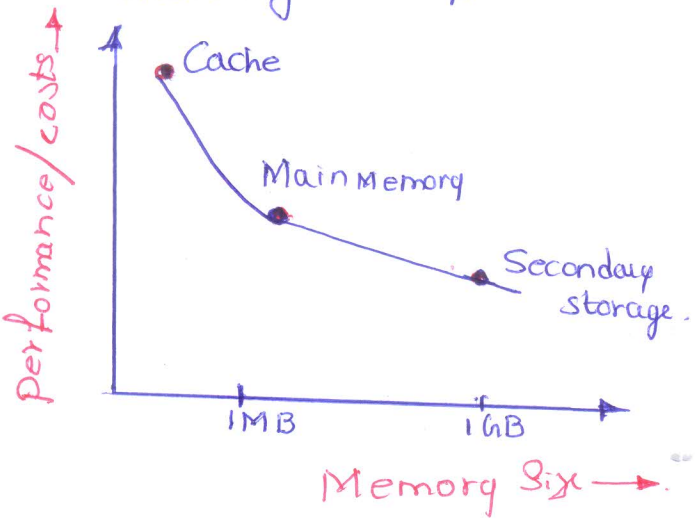
AHB Lite is a subset of AHB and is limited to single Bus master. Multilayer AHB and AHB Lite are used for designs that do not require full features of standard AHB bus.

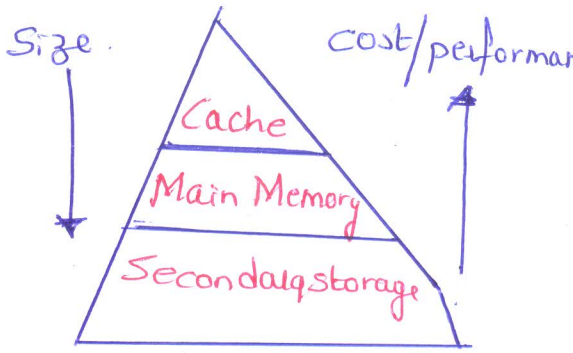
Multilayer AHB are used for systems with multiple processors. They allow parallel processing giving high throughput rates.

Memory:-

An embedded System needs memory to store data & executable code. Memory characteristics includes → Hierarchy, Width and Type.

Hierarchy:- The hierarchical arrangement of memory in current computer architectures is called memory hierarchy. The memory tradeoffs is shown in the figure below,





Cache:- Cache is the fastest memory, physically located nearer the ARM processor. It is used to speed up data transfer between the processor and main memory. Cache is not suitable in real time system response. Many small embedded systems does not require the performance benefits of the cache.

Main Memory:- Main memory is generally large and is stored in separate chips. Main memory is accessed using LOAD/STORE instruction if the values are not available on cache.

Secondary Memory:- Secondary memory is largest and slowest Example includes CD ROM drives, Hard Disk drives.

Width:- The memory width is the number of bits the memory returns on each access. Width is typically 8, 16, 32 or 64 bits. The memory width has direct effect on overall performance and cost ratio.

An uncached system using 32 bit ARM instruction and 16 bit wide memory chips, requires two memory fetches per instruction. Each fetch requires two 16 bit loads reducing system performance but has the benefit of less cost.

The improved performance can be observed in 16-bit Thumb instructions as the core makes only a single fetch to memory to load an instruction from 16 bit memory device.

	8-Bit Memory	16 Bit Memory	32-Bit Memory
ARM 32 bit	4 cycles	2 cycles	1 cycle
Thom 16-bit	2 cycles	1 cycle	1 cycle.

Types :- There are different types of memory and popular memory devices found in ARM based embedded systems includes ROM, Flash ROM, DRAM, SRAM, SDRAM.

ROM (Read only Memory) :- It contains data that is permanently set at the production time & cannot be reprogrammed. ROMs are used in high volome devices that requires no updates or corrections. It can also be used to hold boot code.

Flash Memory :- Can be used to write as well as read. It cannot be used to hold dynamic data as it is slow to write into FLASH ROM. It is used for holding the device firmware or storing long term data that is to be held even after the power is off.

The erasing and writing flash ROM are software controlled without any additional hardware circuitry reducing the cost.

Flash ROM is the most popolar types of ROM and is used as an alternative for mass or secondary storage.

Dynamic Random Access Memory (DRAM) :- DRAM is dynamic and needs a DRAM controller to refresh the storage cells and recharge every few milliseconds. DRAM has the lowest cost per megabyte compared to other types of RAM.

Static Random Access Memory (SRAM) :- SRAM is static and does not requie refreshing. The accen time for SRAM is shorter than the equivalent DRAM because SRAM does not requie a pause between data accen. SRAM requies more Silicon area. SRAM is mostly used for smaller high speed tasks because of it's high cost.

Synchronous Dynamic Random Accen Memory (SDRAM) :- SDRAM is a type of DRAM that can run at much higher clock speeds than conventional memory. SDRAM is clocked and Synchronizes itself with processor bus. The data is fetched internally from memory cells, pipelined and brought out on the bus with a burst.

Peripherals :-

Embedded Systems interact with outside world through peripheral devices. A peripheral (I/O Interface) performs the input and output functions by connecting the devices or sensors that are off chip. Each peripheral device performs a single function and may reside on the chip. A peripheral may be simple serial communication device or a complex 802.11 coireless device.

ARM peripherals are memory mapped where programming interface is set of memory addressed registers. The address of these register is an offset from a specific peripheral base address.

Higher levels of functionality can be implemented with an Embedded System with specialized peripherals called controllers.

Two important controllers are :-

① Memory controller.

② Interrupt Controller.

① Memory Controllers :-

Memory controller connects different types of memory to the processor bus. A memory controller connects different types of memory to the processor bus. A memory controller is configured in hardware to allow certain memory devices to be active. These memory devices allow initialization code to be executed. Some memory devices like DRAM must be set by software.

② Interrupt controller :-

When a peripheral or I/o device wants a service from the processor, it raises an interrupt request to the processor. An interrupt controller determines which device can interrupt the processor at any specific time by setting the appropriate bits in interrupt controller register.

There are two types of interrupt controller for ARM processors :-

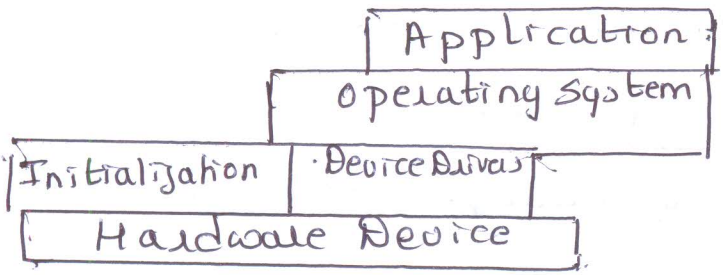
(1) Standard Interrupt Controller
(2) Vector Interrupt Controller.

(1) Standard Interrupt controller :- A standard interrupt controller sends the interrupt signal to the processor core when an external device request a service. It can be programmed to mask or unmask the interrupts. The interrupt handler determines which device requires service by reading the contents of bitmap register in the interrupt controller.

(2) Vector Interrupt Controller :- VIC is more powerful than standard interrupt controller because it assigns priorities to the interrupts and determines which interrupt is to be serviced. The VIC inserts an interrupt to the core only if new interrupt has a priority higher than currently executing interrupt.

1.5. Embedded System Software :-

An embedded system needs a software to drive it. The four typical software components required to control an embedded device is as shown in fig. below.



Software Abstraction Layers Executing on Hardware.

Each Software component in the stack uses a higher level of abstraction to separate the code from the hardware device.

Initialization code :- is the first code executed on board and is specific to a particular target or group of targets. Initialization code sets up the minimum parts of board before handing the control to the operating system.

The operating System :- Provides an interface to control application and manage hardware system resources. Many embedded systems do not require full operating system but merely a simple task scheduler which is either event or poll driven.

Device Drivers: Provide a consistent interface between software and peripherals on the hardware device.

Applications: Performs the tasks required for a device. There can be multiple applications runs on the same device controlled by operating System. The software components can run from ROM or RAM. ROM code fixed on device is called firmware

Initialization (Boot) code :-

Initialization or Bootcode is first code executed on board. It takes the processor from reset state to a state where the operating system can run.

→ Initialization code configures the memory controller and processor caches and initializes some devices.

→ operating system may be replaced by simple scheduler or debug monitor in simple system.

→ Initialization code handles a number of administrative tasks before handing the control to an operating system.

The different tasks of initialization code can be categorized into three phases :-

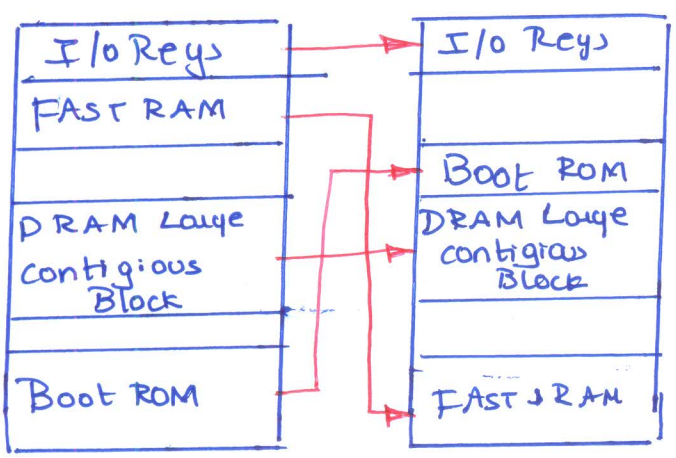
→ Initial Hardware configuration.

→ Diagnostics.

→ Booting.

Initial Hardware Configuration :- involves setting up the target platform so that it can boot an image. Although the target platform comes up in standard configuration, it normally requires modification to satisfy the requirements of the booted image.

for Eg :- A memory system requires reorganisation as shown.



Diagnostics:- The primary purpose of diagnostic code is fault identification and isolation. Diagnostic code tests the system by exercising the hardware target to check if the the target is in working order. It also tracks down the system related issues.

Booting:- Booting involves loading an image and handing control over to that image. The boot process can be complicated if the system must boot different operating system or different versions of the same operating system. Booting an image is the final phase but the image must be loaded first. Loading an image involves copying an entire program including code and data into RAM or just copying a data area containing volatile variables into RAM.

Why is memory reorganization required?

Memory reorganization is an important part of the initialization code as many operating systems expect a known memory layout before they start.

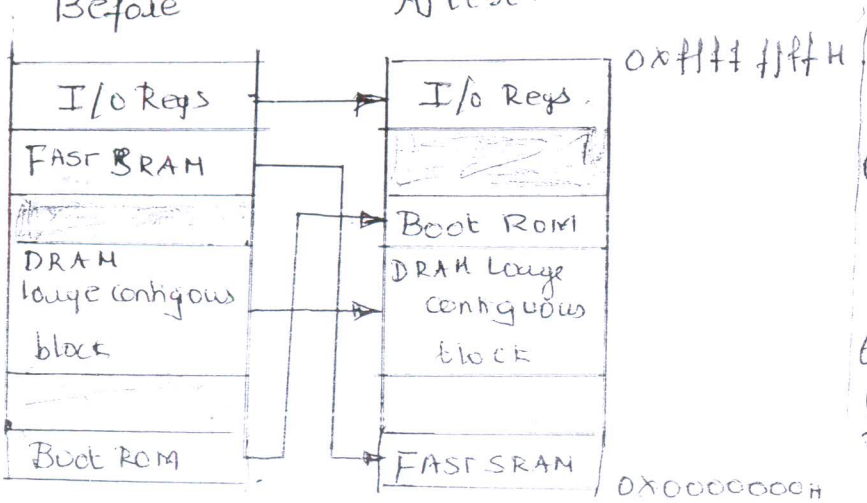


Figure shows memory before and after reorganization. It is common for ARM-based embedded system to provide memory remapping because it allows the system to start the initialization code from RAM at power up.

Operating System

ARM processors support more than 50 operating system. The initialization process prepares the hardware for an operating system to take control. ~~& peripherals, memory~~

The operating system organizes the system resources - peripherals, memory and processing time. With an operating system controlling these resources, it can be efficiently used by different applications running within the operating system environment.

The operating systems can be categorized into two :-

① Real time operating systems (RTOS)

② Platform operating systems.

① Real time operating Systems (RTOS) :- RTOs provide guaranteed response times to events. Different operating systems have different amounts of control over the system time.

A hard real time application requires a guaranteed ~~response~~ response to work. (A miss can lead to a disaster or huge damages)

A soft real time application requires a good response time, but the performance degrades if response time overruns. Systems run on RTOs usually do not have secondary storage.

② Platform operating Systems :- require a memory management unit to manage large, non real time applications and tend to have secondary storage (Linux is an example of platform OS)

The above two operating systems are not mutually exclusive.

Applications :-

Applications are code dedicated to handle a particular task. The operating system controls the environment. An embedded system can have one active application or several applications running simultaneously.

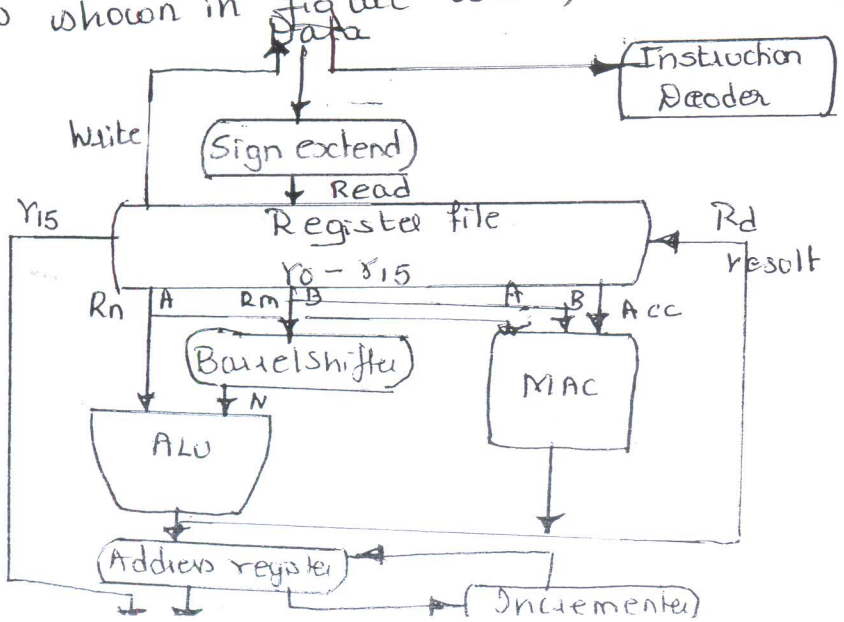
ARM processors find applications in different market segments including → automotive, mobile & consumer devices, mass storage and imaging & within each segment it can have multiple applications.

ARM processors are also found in mass storage devices such as hard drives and imaging product such as inkjet printers i.e applications that are cost sensitive & high volume.

ARM processors ~~does~~ are not found in applications that require leading high performance as they tend to be of low volume & high cost.

ARM Core Processor :

The data flow model of an ARM core processor is as shown in figure below,



The functional units are connected by data buses. The arrows represent the flow of data, the lines represent the buses and each block represent an operation unit or storage area.

Data enters the processor core through the Data bus. The data could be an instruction or data. Figure shows the Von Neumann architecture where data and instructions share the same bus. Harvard implementations of the ARM uses different buses.

The instruction decoder decodes the instruction. ARM processor uses Load-store architecture. Load instruction copies data from memory to registers in the core. Store instructions copies data from registers to the memory.

Data processing instructions does not manipulate data in memory and all data processing is done only in the registers.

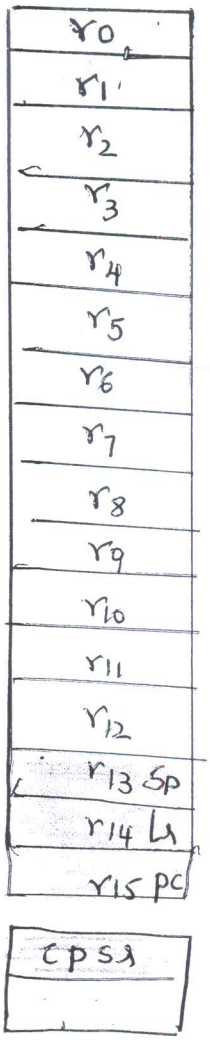
Data items are placed in a register file. Register file is a storage bank made up of 32 bit registers. Register could hold 32 bit signed or unsigned values. 8 bit and 16 bit numbers are converted to 32 bit values by sign extend hardware when they are read and stored from memory to register.

ARM instructions have two source registers Rn and Rm and destination register Rd. Source operands are read from register file using internal buses A and B respectively.

The ALU (Arithmetic Logic Unit) or MAC (Multiply accumulate Unit) takes the source operands from Rn and Rm from buses A and B, computes the result and writes the result directly in to Rd in register file.

Load and Store instruction uses ALU to generate an address of memory which is stored in address register and sent on the address bus.

4.6 Registers :-



Registers of an ARM processor available in user mode is as shown in figure. All registers are 32 bit in size. The 18 active registers can be grouped as 16 data registers and 2 processor status registers

Registers r_0 to r_{15} are program visible registers. r_{13}, r_{14} and r_{15} are assigned a special task. ~~always being a general purpose register~~.

Register r_{13} is used as stack pointer to store the top of the stack in the current processor mode.

Register r_{14} is called link register where the return address is stored when ever a subroutine is called by the core.

Register r_{15} is the program counter which contains the address of next instruction to be fetched by the processor.

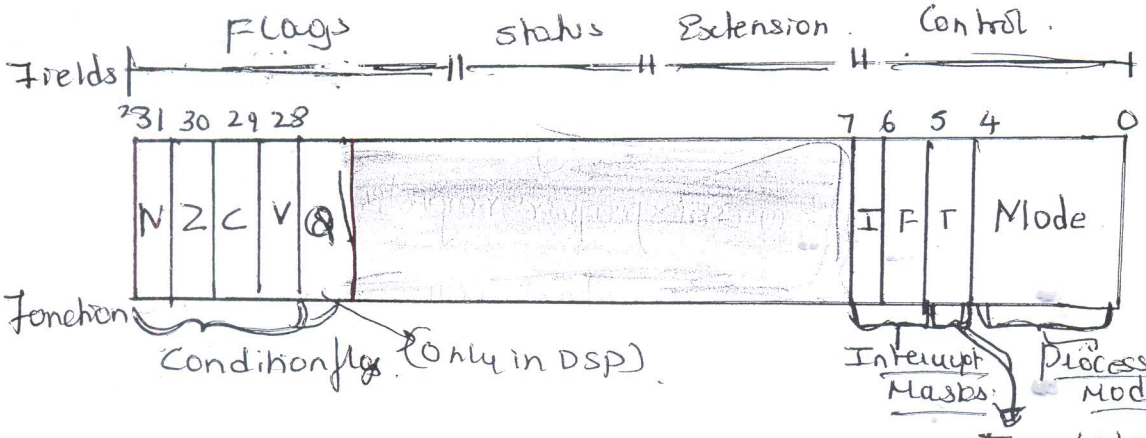
r_{13} and r_{14} can also be used as general purpose registers based on the context. But it is dangerous to use r_{13} as gen. purpose register because when processor is running any form operating system, OS always assumes r_{13} is pointing to stack frame.

Registers r0 through r_{13} are orthogonal i.e any instruction on r0 can be equally well applied to all other registers.

The two processor status register includes cpsr and spsr (current and saved program status registers).

4.7 Current Program Status Register :-

← The ARM core uses the cpsr to monitor and control the internal operations. It is a dedicated 32 bit register that resides in the register file. The basic layout of program status register is as shown below,



The current program status register has four fields, each 8 bits wide → Flags, Status, Extension and control.

The status and extension fields are reserved for future use.

The control field contains processor mode, State and Interrupt Mask bits.

The flag field contains condition flags.

Some ARM processor cores have extra bits allocated. for eg:- J bit can be found in flags field of Jazelle enabled processors which execute 8 bit instructions.

Processor Modes :-

Processor mode determines which registers are active and the access rights to cpsr register itself.

Processor mode can be → Privileged
→ Non privileged

A privileged mode allows full read-write access to cpsr. (12)

A non privileged mode allows read only access to the control field and read write access to condition flags.

There are seven modes → Six privileged which includes abort, fast interrupt, interrupt request, Superwisor, System and Undefined.

Non privileged mode → User mode.

Abort mode :- processor enters abort mode when there is a failed attempt to access memory.

Fast interrupt and Interrupt request : corresponds to the two interrupt levels available on ARM processor.

Superwisor mode is the mode that processor is after the reset and is generally the mode that an operating system kernel operates in.

System mode is a special version of user mode that allows full read write access to cpsr.

Undefined mode is the mode when processor encounters an instruction not defined or not supported by the implementation.

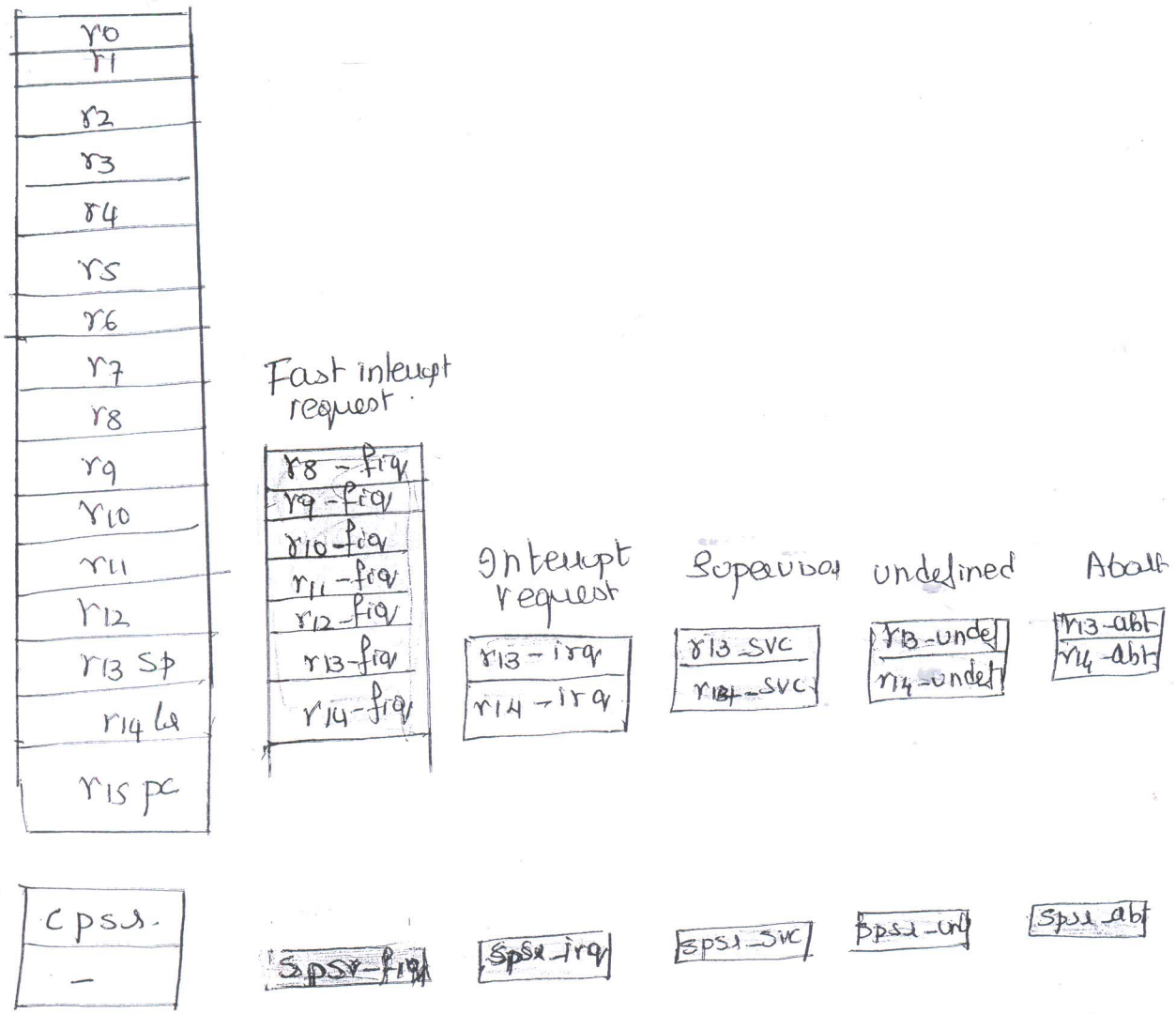
User mode is used for programs and applications.

Banked Registers : are registers that are hidden from a program at different times. They are available only when a processor is in a particular mode.

Figure shows the complete register set of ARM. The unshaded registers are program visible register and shaded are

The banked registers..

User & System.



All processor modes except user mode can change the mode by directly writing into mode bits of cpsr.

All processor modes except system mode have a set of associated banked registers that are subset of main 16 registers as shown above. A banked register maps one to one onto a user mode register. If the processor mode is changed, a banked register will replace an existing register.

for eg:- When the processor is in interupt-request mode, the registers r13 and r14 are banked as r13-irq and r14-irq. The user mode registers r13_usr and r14_usr are not affected by instruction refering these registers. A program still has access to other registers r0 to r12.

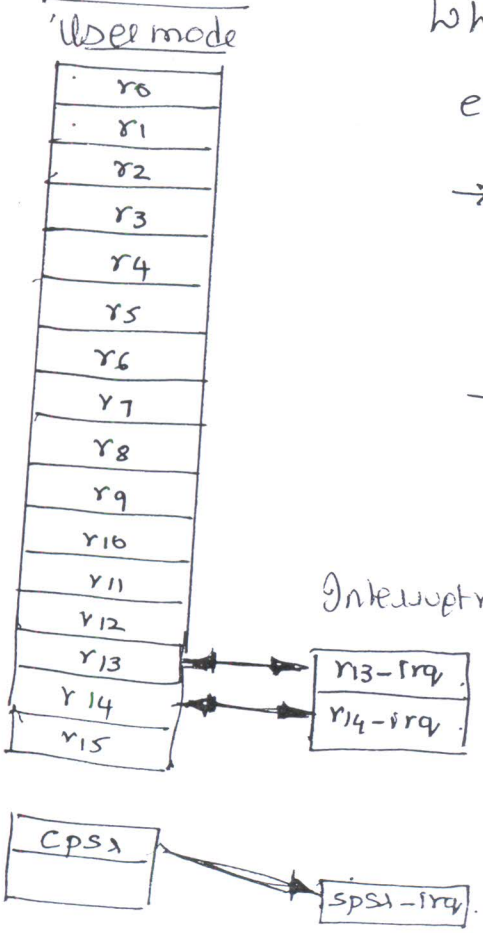
Processor mode can be changed :-

→ by a program that writes directly to the cpsr.

→ by hardware when core responds to exception or interrupt.

Exceptions and interrupts that cause mode change includes reset, interrupt request, fast interrupt request, Software interrupt, data abort, prefetch abort and undefined instruction. Exceptions and interrupts suspend the normal execution of sequential instructions & jump to a specific location.

Figure shows core changing from user mode to interrupt request mode.



When an interrupt request occurs from an external source to processor core,

→ registers r_{13} and r_{14} are to be banked and replaced with r13-irq and r14-irq respectively.

→ Saved program status register (spsr) Saves the previous mode's cpsr.

To return the control back to user mode, a special return instruction is used that instructs the cpsr to restore original values from spsr-irq. and bank in registers r13 and r14.

Spsr is not available in user mode.

psr contents is not copied to spsr when mode change is forced due a program writing directly to cpsr. It is copied only during mode hange caused due to exception or interrupt.

The current active processor mode occopies the five least Significant bits of the cpsr according to the bit format specified in the table below. When power is applied, the core starts in Sopervisor mode which is priveleged. Starting in priveleged mode is useful as initialization code can use full access to the cpsr.

Mode	Abbreviation	priveleged	Bit pattern (mode bits 4:0)
Abort	abt	yes	10111
FastInterupt request	fiq	Yes	10001
Interuptrequest	irq	yes	10010
Sopervisor	SVC	yes	10011
System	sys	yes	11111
Undefined	und	yes	11011
User	usr	No	10000

State and Instruction Set :-

The core supports three instruction sets.

→ ARM (32 bit instruction set)

→ Thumb (16 bit instruction set)

→ Jazelle (8 bit instruction set)

ARM State :- The processor is in ARM state when Jazelle J and thumb T bits in the cpsr are zero. When processor is in ARM State, it executes 32 bit ARM instructions. Processor enters ARM State when power is applied to it.

Thumb state :- When T bit = 1, the processor is in thumb state and executes 16 bit instructions.

Jazelle State :- Few processors support Jazelle state. processor enters Jazelle state when J bit = 1 and executes 8 bit instructions and is a hybrid mix of software and hardware designed to speed up the Java byte codes. Jazelle technology with modified version of Java virtual machine is required to execute Java virtual machine.

ARM and Thumb instruction set features.

Features	ARM (cpsr T=0)	Thumb (cpsr T=1)
Instruction Size	32 bit	16 bit
Core instructions	58	30
Conditional Execution	most	only branch Instruction
Data processing Instructions	access to barrel shifter ALU	Separate barrel shifter and ALU instructions.
Program status register	read-write in privileged mode	· no direct access.
Register Usage.	15 general purpose register + pc	8 gen. purpose reg + 7 high registers + pc.

Jazelle instruction set features :-

	Jazelle (cpsr T=0, J=1)
Instruction Size	8 bit.
Core Instructions.	over 60% of Java byte codes are implemented in hardware; the rest of the codes are implemented in software.

Jazelle instruction set is a closed instruction set and is not openly available.

Interrupt Masks :-

Interrupt requests are requests from the peripherals asking for a service from the processor. There are two interrupt request levels available on ARM processor core - interrupt request (IRQ) and fast interrupt request (FIQ).

Interrupt Masks are used to stop or allow interrupt requests from interrupting the processor.

The cpsr has two interrupt mask bits (7 and 6).

I bit is used to control the masking of IRQ. when I=1, IRQ is masked

F bit controls the masking FIQ. when F=1, FIQ is masked.

Condition flags :-

Condition flags are status flags which are set based on the result of a comparision or ALU operations that specify the S instruction suffix.

for eg :- SUBS subtraction instruction sets Z flag if result of a substraction operation results in zero.

The various condition flags supported are listed below,

Q → Saturation flag :- set when result causes an overflow and or saturation

V → Overflow : set when result causes a signed overflow.

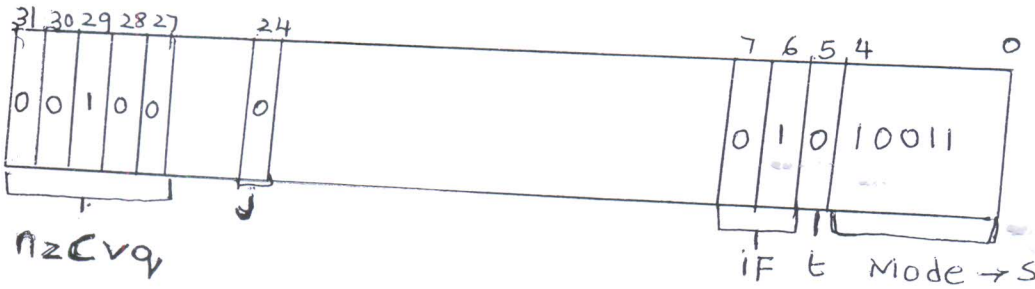
C → Carry : set when result causes an unsigned carry.

Z → Zero : set when result is zero ; used to indicate equality.

N → Negative : bit 31 of result is binary 1 (when result is negative)

Bits 27 to 31 of cpsr are condition flag bits. as shown in the cpsr format below. Figure shows cpsr format with Jazelle and Dsp extensions

The condition flag is represented in Lower case when Correspon ding bit is zero. and in uppercase when bit is one. j bit reflects the state of processor.



j=0 and t=0 indicates that processor is in ARM state. F bit. is 1 allows fast interrupt request, i=0 masks interrupt request. C flag is the only conditional flag set & remaining nzvq is clear. Processor is in SVC mode (supervisor mode) since mode [4:0] is 10011.

Conditional Execution :-

Instructions can be executed only when a specific ondition is met using conditional execution. Most instructions ave a condition attribute which decides whether a instruction is to e executed or not. Before execution of instruction processor ompares the condition attribute with the condition flag in cpsr, e instruction is executed if they match else it is ignored. he conditional code mnemonics are listed below:-

emonic	Name	condition flag
a →	equal →	Z
E →	not equal →	z
IS →	Carry set / unsigned higher or same	C
CC LO →	Carry clear / unsigned lower	c
MI →	minus / negative.	N
PL →	plus / positive or zero	n
VS →	overflow	V

VC → no overflow 0. GE → greater than equal NV or NN

HI → Unsigned higher zC LT → Signed less than NV or NV

LS → Unsigned lower or same ZorC. GT → Signed greater than NzV or NzV.

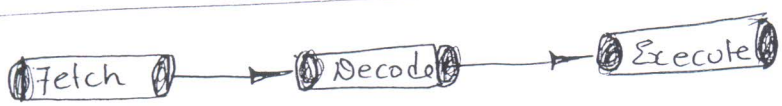
LE → less than or equal ZorNVorNV. AL → always (Unconditional)

Pipeline :-

Instruction execution happens in three phases fetch, decode and Execute.

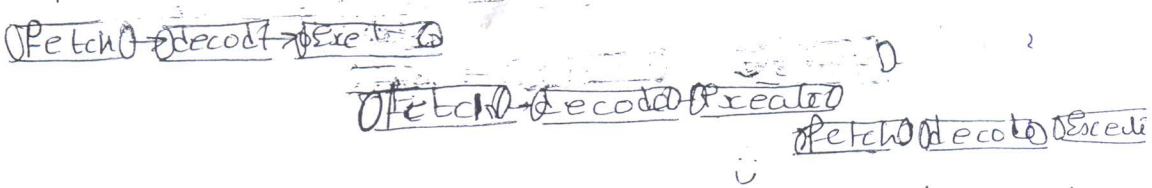
Fetch loads an instruction from memory. Decode decodes the fetched instruction Execute phase processes the instruction and write the result back into register.

A pipeline is the mechanism used to speed up the execution by fetching the next instruction while other instructions are being decoded and executed.



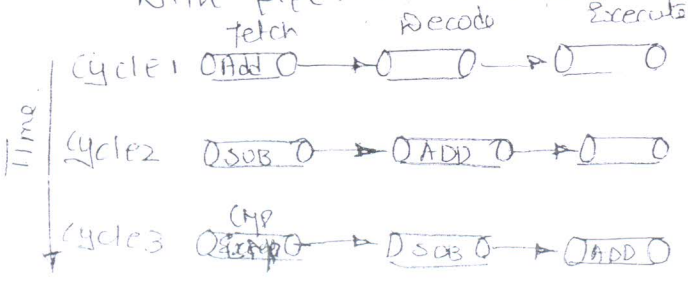
Ex:- Consider three instructions to be executed in sequence as shown below,

ADD
SOB
CMP.



Without pipeline the instructions would take 9 clock cycles to complete in 9 clock cycles.

With pipeline, execution of instruction is as shown below.

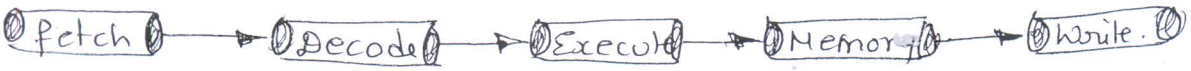


In the first cycle, core fetches ADD from memory & in second cycle, core fetches SOB from memory & decodes ADD. In third cycle, ADD is executed, SOB is decode & CMD is fetched. The pipeline

allows the core to execute an instruction every cycle.

The amount of work done at each stage is reduced as the pipe length increases and processor attain higher operating frequency. which intern increases the performance. The system Latency also increases because it takes more cycles to fill the pipeline before the core can execute an instruction. There can be data dependency between certain stages as the pipeline length increases.

ARM 9 five stage pipeline.



ARM10 Six stage pipeline

Even though ARM9 and ARM10 pipelines are different, pipeline executing characteristics are same as on ARM7. Code written for the ARM7 will execute on ARM9 as ARM 10.

Pipeline Executing Characteristics :-

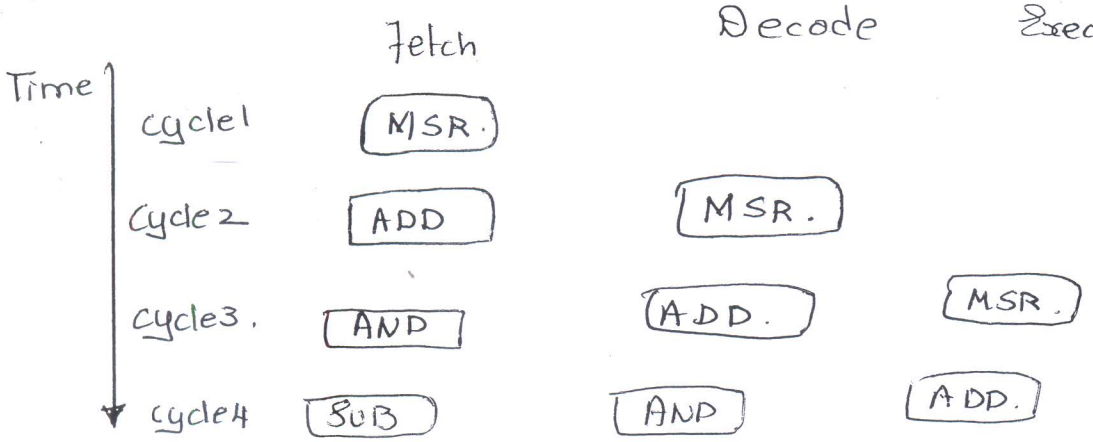
The processing of an instruction is complete once the instruction completes the execution phase. i.e. fourth instruction is fetched This can be illustrated considering a simple example of sequential execution of :-

MSR (Instruction to move the contents of (4 gen purpose) register to CPSR)

ADD (Instruction to add the two operands)

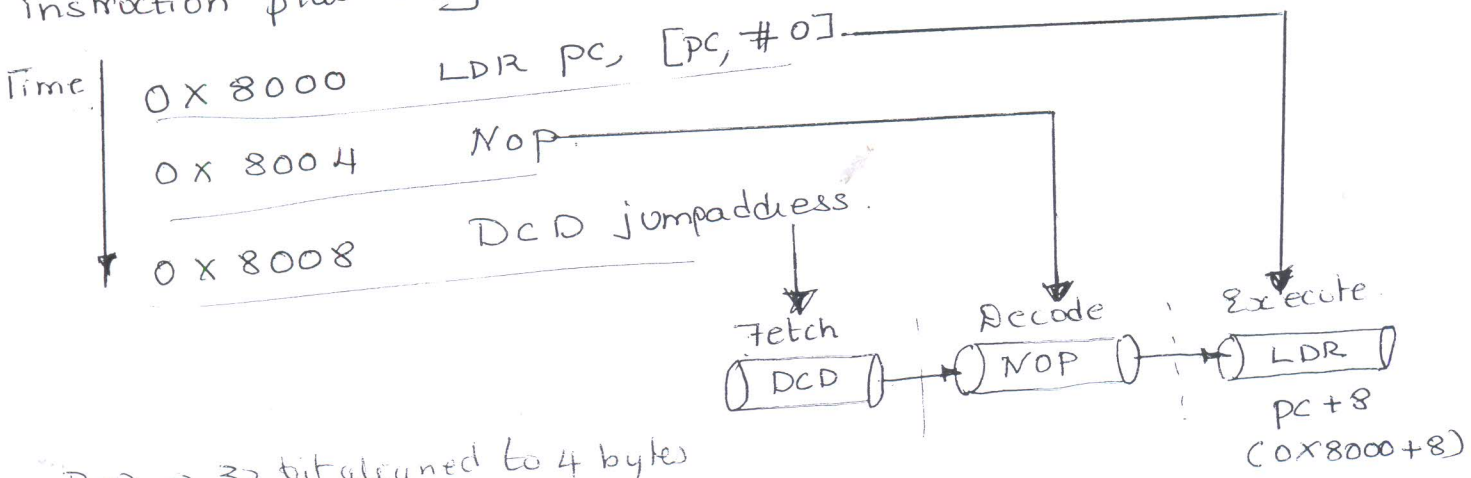
AND (" AND ")

(" ")



MSR instruction moves the contents of a general purpose register to CPSR and can be used to enable IRQ interrupts. IRQ is enabled only after MSR instruction completes the execution stage in pipeline. Thus IRQ are enabled once ADD instruction enters the execution stage of the pipeline.

Pipeline and program counter characteristics is illustrated in figure below. program counter is register which points to the next instruction to be fetched. i.e pc always points to the address of current instruction plus 8 bytes.



DCD → 32 bit aligned to 4 bytes

Nop → no operation

LDR → load word into a register

Other characterstics of ARM 10 includes :-

① The execution of a branch instruction as branching by direct modification of the pc causes the ARM core to flush it's pipeline.

② ARM10 uses branch prediction, which reduces the pipeline flushes by predicting possible branches and loading the new branch address prior to the execution of instruction.

③ The interrupt instruction in the execution phase will be completed even after an interrupt has been encountered. Other instructions in the pipeline are aborted. The processor will the pipeline from appropriate entry in the vector table.

4-4. Exceptions, Interrupts and the Vector Table :-

Interrupt is a mechanism by which allows the processor to transfer control from current program execution to another program of more importance or higher priority.

Exceptions are interrupts invoked by the processor when there conditions CPU cannot handle.

When an exception or interrupt is encountered, the processor runs a specific routine designed to handle a particular interrupt or exception. The control is transferred by loading pc with specific address range. These called the Vector table. The entries in the vector table are instructions that branch to specific routines designed to handle a particular

The Vector table shown below begins at address 0x00000000
* Some processors the vector table is located at higher address in memory starting at (0xffff0000h). Operating systems such as Linux and Microsoft's embedded products has this feature.

When an interrupt or exception is encountered, the processor suspends normal execution and starts loading from exception vector table.

Exception/Interrupt	Notation	Address	High address
Reset	RESET	0x00000000H	0xffff0000h
Undefined Instruction	UNDEF	0x00000004H	0xffff0004h
Software Interrupt	SWI	0x00000008H	0xffff0004H
prefetch abort	PABT	0x0000000CH	0xffff000CH
Data abort	DABT	0x00000010H	0xffff0010H
Reserved	—	0x00000014H	0xffff0014H
Interrupt Request	IRQ	0x00000018H	0xffff0018H
Fast Interrup Request	FIQ	0x0000001CH	0xffff001CH

Reset Vector :- is the location of first instruction to be executed when power is applied. The instruction branches to initialization code.

Undefined Instruction vector :- used when processor cannot decode an instruction.

Software Interrupt Vector :- used with SWI instruction to invoke an operating system routine.

prefetch abort Vector :- when attempt to fetch an instruction from an address with out access right is made, instruction at this address branches to an appropriate routine. The actual

Data abort Vector :- Branches to an interrupt when an instruction attempts to access data memory without access permissions.

Interrupt Request Vector :- is by external hardware requesting service. It is raised only if IRQs are not masked in cpsr.

Fast Interrupt request :- is request for service by external hardware for faster response times. It is raised only if FIQs are not masked in cpsr.

4.10 Core Extensions :-

Core extensions are hardware components placed around the ARM core to improve performance, manage resources and to provide extra fonctionality. They are designed to provide flexibility in handling a particular application. The three hardware extensions are

→ Cache and tightly Coupled Memory.

→ Memory Management.

→ Coprocessor Interface.

Cache and tightly Coupled Memory :-

The cache is block of fast memory placed between the main memory and core to improve the performance. ARM-based processor uses a single level cache internal to processor.

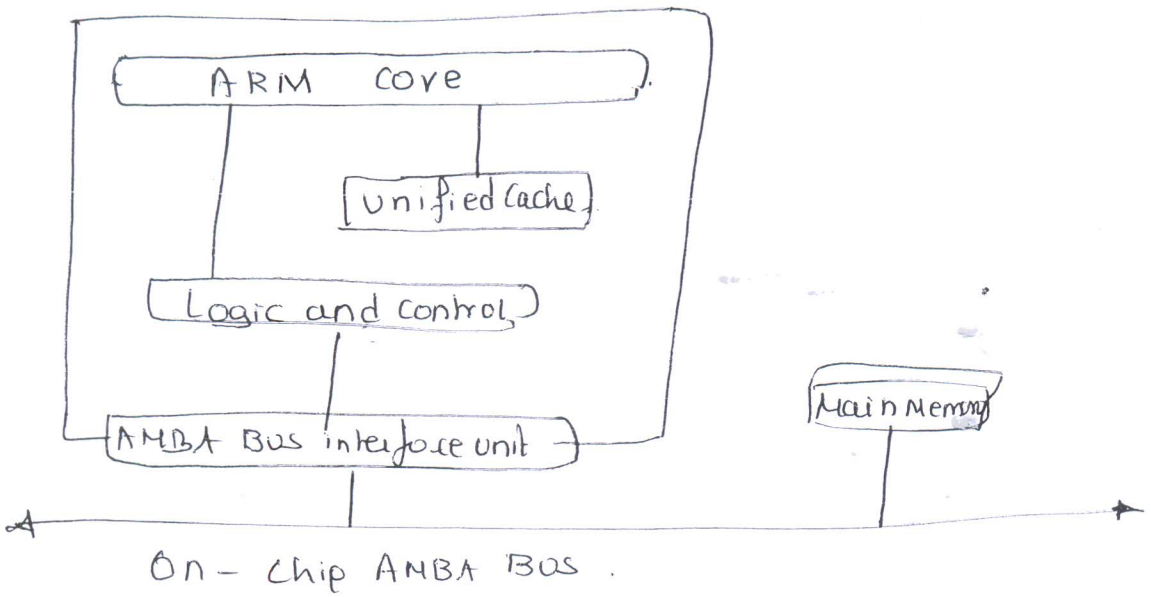
ARM has two forms of cache → Von Neumann architecture with cache

→ Harvard Architecture with TCMs

~~Summarize pipeline characteristics~~ :-

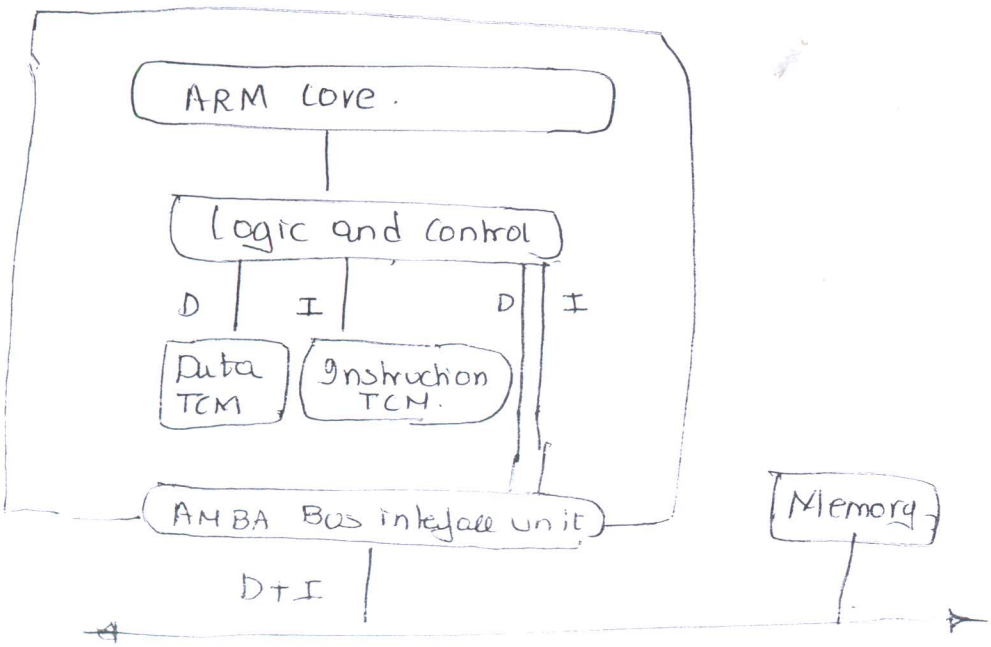
(6) Von Neumann Architecture with cache :-

Von Neumann architecture combines both data and instruction into a single unified cache as shown below,



Harvard Architecture with TCMs :-

Harvard cache has separate caches for data and instruction. A cache provides an overall increase in performance which comes at expense of predictable execution.



In real time System, the code execution should be deterministic i.e the time taken for loading and storing instructions or data must be predictable. This can be achieved by using Tightly coupled Memory (TCM). TCM is fast SRAM located close to the core and guarantees the clock cycle required to fetch instructions or data critical for real time algorithms requiring deterministic behavior. Thus both technologies are combined to have improved performance and predictable real time response.

Memory Management :-

Memory management hardware is used to organise multiple memory devices and to protect the system from applications trying to make inappropriate access to hardware.

ARM cores have three types of memory management hardware

→ Non protected memory (no extension no protection)

→ Memory protection Unit (MPU) (providing limited protection)

→ Memory Management Unit (MMU) (providing full protection).

Non protected memory :- is fixed and provides little flexibility. It is used with simple, small embedded systems that require no protection from rogue applications.

MPUs employs system with limited memory regions. These regions are controlled with a set of special coprocessor registers where each region is defined with specific access permission. Used with System that requires memory protection but does not have complex memory map

MMUs :- are memory hardware available on ARM. It uses a set of translation table to provide control over the memory...

These tables are stored in main memory and provide a virtual to physical address map as well as access permissions.

It is used for more sophisticated platform operating systems that support multi tasking.

Coprocessors :- More than one coprocessor can be added to ARM core via coprocessor interface. It extends the processing features by extending the instruction set or by providing configuration registers. Coprocessor I/f allows more than one coprocessor to be added to ARM core. The coprocessor can be accessed thro' a group of dedicated ARM instructions that provide load/store type I/f.

for eg :- ARM processor uses coprocessor 15 registers to control cache, Tightly coupled Memory (TCMs) and for Memory management

Coprocessor can also extend the instruction set by providing specialized group of instructions.

for eg :- A set of specialized instructions can be added to ARM instructions to perform vector floating pt. (VFP) instructions.

These new set of instructions are processed in decode stage of ARM pipeline. If decode stage encounters a coprocessor instruction, it is directed to relevant coprocessor. If coprocessor is not found, ARM takes an undefined instruction exception.