Page No.
Date / /

(18CS44) MICROCONTROLLERS AND EMBEDDED SYSTEMS.

MODULE - 1.

* Microprocessors versus Microcontrollers
* ARM Embedded Systems:
  - ★ - The RISC design philosophy
  - ★ - The ARM Design philosophy
  - ★ - Embedded system Hardware
  - ★ - Embedded system Software
* ARM Processor Fundamentals:
  - ★ → Registers
  - ★ → Current Program status Register
  - ★ → Pipeline
  - ★ → Exceptions
  - ★ → Interrupts
  - ★ → Vector Table
  - ★ → Core Extensions

www.takeiteasyengineers.com

Microprocessor:

A microprocessor is an electronic component that is used by a computer to do its work.

It is a central processing unit on a single integrated circuit chip containing millions of very small components including transistor, resistors and diodes that work together.

- In December 1970, Gilbert Hyatt filed a patent application entitled "Single chip Integrated Circuit Computer Architecture", the first basic patent on the microprocessor.

- The microprocessor was invented in the year 1971 in the Intel Labs.

The first processor was a 4-bit processor and called 4004.

If processor can access 4-bit of data then it is called 4-bit processor.

If processor can access 8-bit of data then it is called 8-bit processor eg: 8008, 8080.

- In 1978, Intel released the 8086 microprocessor - a 16 bit microprocessor

- Addressed 1M bytes
- Executed 2.5 MIPs (millions of instructions per second)
- A small 6-byte instruction cache or queue that pre-fetched a few instruction before they were executed
- Its instruction set contained over 20,000 instructions

The microprocessor sometimes called CPU

- Heart of microprocessor based on computer systems
- Controls the memory and I/O through buses
- Transfers address, data and control information between an I/O device or memory and the microprocessor via buses.
- Three buses exist for the transfer of the following information -
  1. Address
  2. Data
  3. Control
- Memory and I/O are controlled through instructions
- Instructions are stored in the memory and executed by the microprocessor

www.takeeasyengineers.com

The three main tasks performed by microprocessor for the computer system are

→ Data transfer between itself and the memory or I/O systems

→ Simple arithmetic and logic operations

→ Program flow via simple decisions

The power of microprocessor is that

→ It has capability to execute hundreds of millions of instruction per second

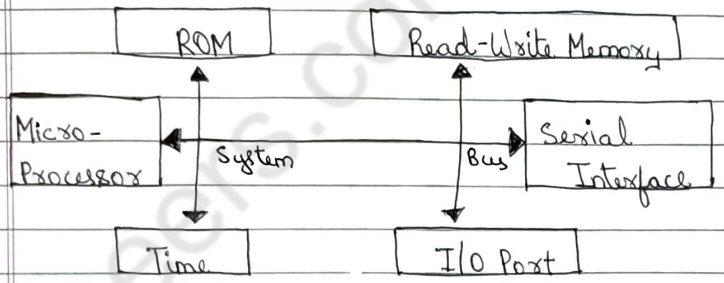
→ Also, a microprocessor can make simple decisions based upon numerical facts

Applications of microprocessor:

- Computer applications
- Control application (micro controllers, embedded controllers etc.,)
- Communication (DSP processors, cell phones etc..).

Block diagrams:

Microprocessor:



Microcontroller:

Micro Controller	Read Only memory	Read-write memory
Timer	I/O Port	Serial Interface

Microcontrollers:-

It is a compact integrated circuit designed to govern a specific operations in an embedded system.

A typical microcontroller includes a processor, memory, I/O peripherals on a single chip.

www.easyengineering.com

Microprocessor	Microcontroller
* heart of computer system	* heart of embedded system
* Cost of the entire system increases	* Cost of the entire system is low
* Since memory and I/O has to be connected externally the circuit becomes large.	* Since memory & I/O are present internally, the circuit is small.
* They have less number of registers, hence more operations are memory based.	* They have more number of registers hence the programs are easier to write.
* Cannot be used in compact systems and hence in-efficient	* Can be used in compact systems and hence it is an efficient technique
* Mainly used in personal computers	* Used mainly in washing machine, MP3 players.
* Processing speed is above 1GHz.	* Processing speed is 8MHz to 50MHz.

ARM Embedded System:-

ARM - Acorn Risc Machine

The ARM processor core is a key component of many successful 32-bit embedded systems.

Widely used in mobile phone, handheld organizers and a multitude of other everyday portable consumer devices

History of ARM

The first ARM1 prototype was designed in 1985.

Success → a simple and powerful original design, which continues to improve today through constant technical innovation.

Example :- ARM7TDMI :-

→ This provides upto 120 dhrystone MIPS
→ High code density and low power consumption
→ ideal for mobile embedded systems/ devices.

www.takeiteasyengineers.com

★ The RISC Design Phylosophy.

→ The ARM core uses reduced instruction set computer (RISC) architecture.

RISC

→ RISC is a design philosophy aimed for delivering simple but powerful instructions, that execute within a single cycle at a high clock speed.

→ It reduces the complexity of instruction performed by the hardware because it is easier to provide greater flexibility and intelligence in software rather than hardware.

→ It places greater demands on the compiler

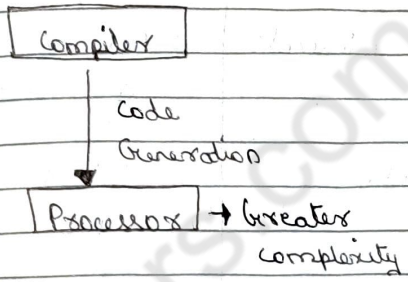
CISC

The complex instruction set computer

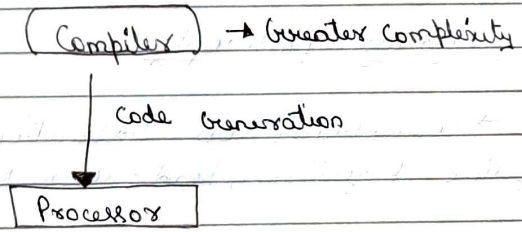
→ It relies more on the hardware for instruction functionality

→ In CISC instructions are more complicated.

CISC.



RISC.



Difference b/w CISC and RISC.

CISC	RISC
★ Instructions can take several clock cycles	★ Single-cycle instructions
★ Hardware-centric design	★ Software-centric design
★ Large number of instructions	★ Small number of fixed length instruction

www.takeiteasyengineers.com

CISC	RISC
* May support microcode	* Only one layer of instruction
* Compound addressing modes	* limited addressing modes.
* Conditional jump is usually based on status register bit	* Conditional jump can be based on a bit anywhere in memory.

RISC design philosophy

→ Concentrates on Instructions, pipeline, Registers, load-store Architecture

1. Instructions:-

→ Risc has reduced number of instruction classes.

→ It has simple operations
→ Each instruction can execute in a single clock cycle.
→ Compiler/programmer reduces complicated operations
→ Each instruction is having fixed length allows pipeline to fetch future instructions.

Pipeline :-

→ Processing of instructions is broken down into smaller units and instructions are executed in parallel by pipelines

→ Pipeline advances by one step on each cycle maximum throughput.

Registers :

→ They have a large general-purpose register set

→ Any register can contain either data or an address.

→ Registers act as the fast local memory store for all data processing operations.

Load-store architecture :-

→ Processor operates on data and held in registers

→ Separate load and store instruction transfer data between the register bank and external memory.

→ Separating memory accesses from data processing provides an advantage because you can use data items

www.takeiteasyengineers.com

held in the register bank multiple times without needing multiple memory accesses.

★ The ARM Design Philosophy

- Physical features that have driven the ARM processor design are

→ Battery power

The ARM processor specially designed to be small to reduce power consumption and extend battery operation

It is essential for applications such as mobile phones and personal digital assistants (PDAs).

→ High code density

Embedded system have limited memory due to cost or physical size restrictions.

It is useful for applications that have limited on-board memory such as mobile phones and mass storage device.

→ Price sensitive

- Embedded systems are price sensitive they use slow and low-cost memory devices to get substantial savings

It is essential for high volume applications like digital cameras.

- It reduces the area of the die taken up by the embedded processor to reduce cost of the design and manufacturing for the end product.

Hardware debug technology

- ARM has incorporated hardware debug technology so that software engineers can view what is happening while the processor is executing code.

The ARM core is not a pure RISC architecture

- because of the constraints of its primary applications.

- In some sense, the strength of the ARM core is that does not take RISC concept too far.

www.takeitbasyengineers.com

Instruction set for Embedded Systems

The ARM instruction set differs from the pure RISC definition in several ways, that make the ARM instruction set suitable for embedded applications.

The differ is mainly by :
→ Variable cycle execution
→ Inline barrel shifter
→ Thumb 16-bit instruction set
→ Conditional execution
→ Enhanced instruction

→ **Variable cycle execution :**

- Not every ARM instruction executes in a single cycle
- The transfer can occur on sequential memory addresses
- Code density is also improved since multiple register transfers are common operations at the start and end of functions.

→ **Inline barrel shifter :**

It is a hardware component that preprocesses one of the input registers before it is used by an instruction.

This expands the capability of many instructions to improve core performance and code density

Barrel shifter is a digital circuit that can shift a data word by a specified number of bits without the use of any sequential logic, only pure combinational logic.

→ **Thumb 16-bit instruction set :-**

ARM embedded the processor core by adding a second 16-bit instruction set called Thumb.

Thumb permits the ARM core to execute either 16 or 32-bit instruction

The 16-bit instructions improve code density by about 30% over 32-bit fixed-length instructions.

www.takeeasyengineers.com

Conditional execution :-

An instruction is only executed when a specific condition has been satisfied

This feature improves performa-nce and code density by reducing branch instruction.

Enhanced instruction :

The enhanced digital signal processor (DSP) instruction were added to the standard ARM instruction set to support fast 16x16-bit multiplier operations

These instruction allow a faster-performing ARM processing

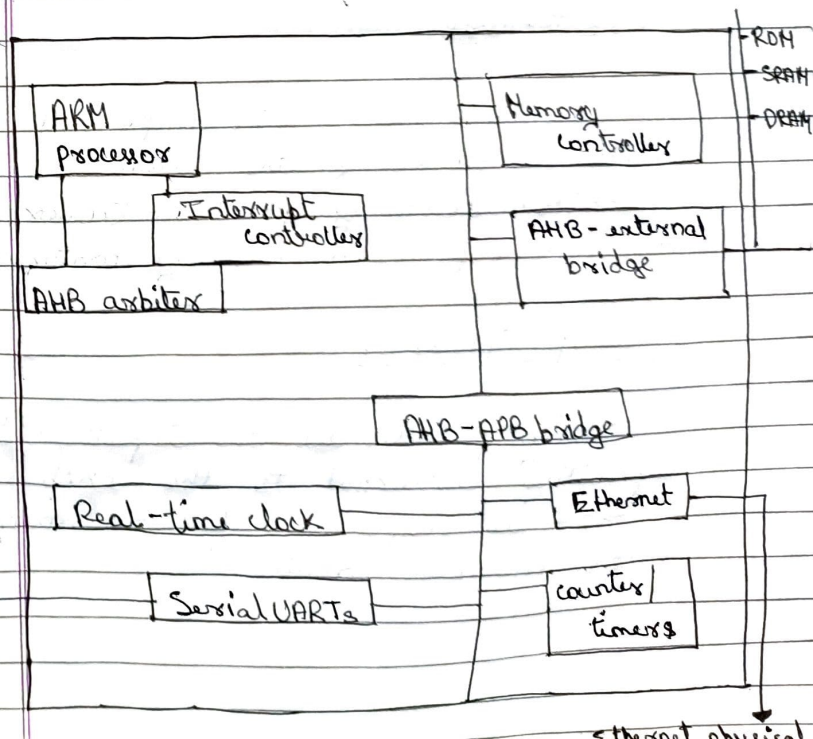
These additional features have made the ARM processor one of the most commonly used 32-bit embedded processor cores.

→ EMBEDDED SYSTEM HARDWARE.

→ Basically embedded system can control many different devices from small sensors to real-time control systems.

→ All these devices use a combination of software and hardware components.

→ Embedded system basic components are ARM processors, controllers, peripherals and Buses.



www.takeuteasyengineers.com

1. The ARM processor.

→ It controls the embedded device
→ Different versions of the ARM processor are available based on embedded system to suit the desired operating characteristics.
→ It comprises a core.
The core is the execution engine that processes instruction & manipulates data and also surrounding components. The surrounding component can include memory and cache also.

2. Controllers.

→ Controllers are the important functional blocks of the system
→ Two commonly found controllers are interrupt and memory controllers

3. The peripherals.

→ They provide all the input - Output capability external to the chip
→ It is responsible for the uniqueness of the embbed device

4. BUS.

→ Used to communicate b/w different part of the devices.
→ Different parts can be ARM processors, peripherals, controllers.

ARM Bus Technology.

→ Embedded device use an on-chip that is internal to the chip.

→ It allows different peripheral devices to be interconnected with an ARM core.

→ There are two different classes of device attached to the bus:
1. The ARM processor core is a bus master - a logical device capable of initiating a data transfer with another device across the same bus.
2. Peripherals tend to be bus slaves - logical devices capable only of responding to a transfer request from a bus master device.

→ A bus has two architecture levels:
- A physical level - It covers the electrical characteristics and bus width (16, 32 or 64 bits)
- The protocol - The logical rules that govern the communication b/w the

www.takeeasyengineers.com

processor and a peripheral.

AMBA BUS Protocol.

→ The Advanced Microcontroller Bus Architecture (AMBA) was introduced in 1996 and has been widely adopted as the on-chip bus architecture used for ARM processors.

→ The first AMBA buses introduced were,

- ASB - ARM System Bus
- APB - ARM Peripheral Bus

Later
- AHB - ARM High performance Bus

Using AMBA

→ Peripheral designers can reuse the same design on multiple projects

→ A peripheral can simply be bolted onto the on-chip bus without having to redesign an interface for each different processor architecture

→ This plug-and-play interface for hardware developers improves availability and time to market.

→ AHB provides higher data throughput than ASB because it is based on a centralized multiplexed bus scheme rather than the ASB bidirectional bus design. This change allows the AHB bus to run at higher clock speeds.

→ ARM has introduced two variations on the AHB bus: Multi-layer AHB and AHB-lite.

→ The Multi-layer AHB bus allows multiple active bus masters.

→ AHB-Lite is a subset of the AHB bus and it is limited to a single bus master.

The example device shown in fig- has three buses:

- An AHB bus for the high-performance peripherals
- An APB bus for the slower peripherals
- A third bus for external peripherals

→ EMBEDDED SYSTEM SOFTWARE.

- An embedded system needs software to drive it.

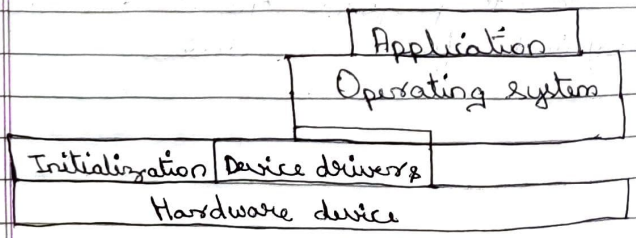


fig-2.

www.rakeshbasyengineers.com

Four typical software components required to control an embedded device are:-

- **The initialization code:**
  → It is a first code executed on the board/system and is specific to a particular target or group of targets
  → It sets up the minimum parts of the board before handing control over to the operating system.

- **The Operating System:**
  → It provides an infrastructure to control applications and manage hardware system resources.

- **The device drivers:**
  → It provides a consistent software interface to the peripherals on the hardware device

- **An application:**
  → It performs one of the tasks required for a device

Initialization (Boot) Code:

→ Initialization code take the processor from the reset state to a state where the Operating system can run.

→ It usually configures the memory controller and processor caches and initializes some devices

→ It handles a number of administrative task prior to handing control over to an operating system.

→ We group these different tasks into three phases:
1. Initial hardware configuration
2. Diagnostics
3. Booting

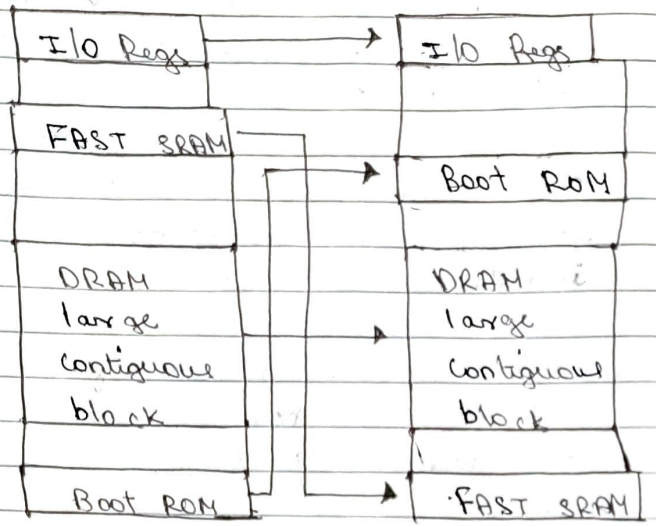
- **Initial hardware Configuration:**
  → It involves setting up the target platform, so that it can boot an image.

→ Initializing or organizing memory is an important part of the initialization code, because many Operating system expect a known memory layout before they can start

www.takeeasyengineer.com

Before

After



→ Memory remapping allows the system to start the initialization code from ROM at power-up.

→ The initialization code then redefines or remaps the memory map to place RAM at address 0x00000000 - an important step because then the exception vector table can be in RAM and thus can be reprogrammed.

2. Diagnostics:

→ It is a set of code to check whether target is in working order.

→ It also tracks down standard system-related issues.

→ The primary purpose of diagnostic code is fault identification and isolation.

3. Booting:

→ It involves loading an image and handing control over to that image.

→ The boot process itself can be complicated if the system must boot different operating system or different versions of the same system.

→ It is nothing but loading operating system from RAM to memory [Another meaning for booting].

→ Booting an image is the final phase, but first you must load the image.

→ Loading an image involves anything from copying an entire program including code and data into RAM, to just copying a data area volatine variable into RAM.

www.sakiteasyengineering.com

→ Once booted, the system hands over control by modifying the program counter to point into the start of the image.

- **Operating System:**

→ The initialization process prepares the hardware for an O.S to take control.

→ An O.S organizes the system resources: the peripherals, memory, processing time

→ ARM processor support 50 OS
→ Operating System (OS) is divided into two main categories:
  - Real-time operating System (RTOSs)
  - Platform OS

Real time OS
→ It provides guaranteed response time to events.
→ Different OSs have different amounts of control over the system response time.
→ A hard real-time application - requires a guaranteed response to work at all.
→ A soft real-time application - requires a good response time, but the performance degrades more gracefully if the response time overruns.

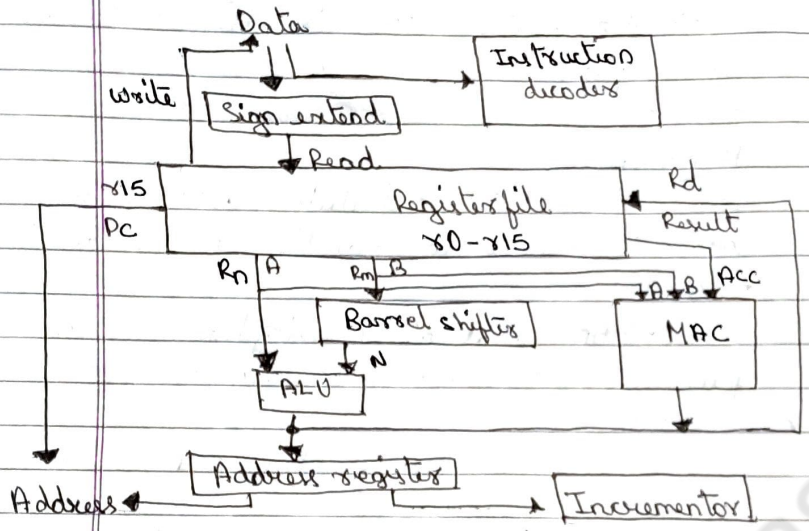
Platform OS
→ It requires a memory management unit to manage large, non-real-time applications and tend to have secondary storage.
→ The Linux OS is the typical example.

Applications :-
→ The OS schedules applications - code dedicated to handle a particular task.
→ An application implements a processing task; the OS controls the environment
→ An embedded system can have one active application running simultaneously
→ ARM processors are found in numerous market segments including networking, auto-motive, mobile and consumer devices, mass storage and imaging.
→ ARM processor is found in networking applications like home gateways
→ found in mobile device segment
→ found in mass storage devices such as hard drives
→ found in imaging products such as inkjet printers & applications that are cost sensitive and high volume.

www.takeiteasyengineers.com

ARM Processor Fundamentals.

ARM Core Data Flow Model.



The arrows represent the flow of data. The lines represent the buses and the boxes represent either an operation unit or storage area.

Data enters the processor core through the data bus. The data may be an instruction to execute or a data item.

Figure shows a Von Neumann implementation of the ARM - data item and instructions share same bus.

The instruction decoder translates instructions before they are executed. Each instruction executed belongs to a particular instruction set.

The ARM processor, like all RISC processors uses load-store architecture - means it has two instruction types for transferring data in and out of the processor.

- Load instructions copy data from memory to register in the core.
- Store instructions copy data from register to memory.

There are no data processing instructions that directly manipulate data in memory. Thus data processing is carried out in registers.

The sign extend hardware converts signed 8-bit and 16-bit numbers to 32-bit values as they are read from memory and placed in a register.

ARM instructions typically have two source registers, Rn and Rm, and a single result or destination register, Rd. Source operands are read from the register file using the internal buses A & B.

The ALU or MAC (multiply-accumulate unit) takes the register values Rn & Rm from the A & B buses & computes results.

Data processing instructions write the result in Rd directly to the register file.

Load and store instructions use the ALU to generate an address to be held in the address register and broadcast on the Address bus.

www.takeitseasyengineers.com

One important feature of the ARM is that register Rm alternatively can be preprocessed in the barrel shifter before it enters the ALU. Together the barrel shifter and ALU can calculate a wide range of expressions and addresses.

After passing through the functional units, the result in Rd is written back to the register file using Result bus.

For load and store instruction the Incrementer updates the address register before the core reads or writes the next register value from or to the next sequential memory location.

The processor continues executing instructions until an exception or interrupt changes the normal execution flow.

→ Registers.

r0
r1
r2
r3
r4
r5
r6
r7
r8
r9
r10
r11
r12
r13 sp
r14 lr
r15 pc

- There are upto 18 active registers : 16 data register and 2 processor status registers
- The data register visible to the programmer are r0 to r15.
- The ARM processor has three register assigned to a particular task or special function: r13, r14, r15.

→ Register r13 is traditionally used as stack pointer (SP) and stores the head of the stack in the current processor mode.

→ Register r14 is called as link register (lr) and is where the core puts the return address whenever it calls subroutine.

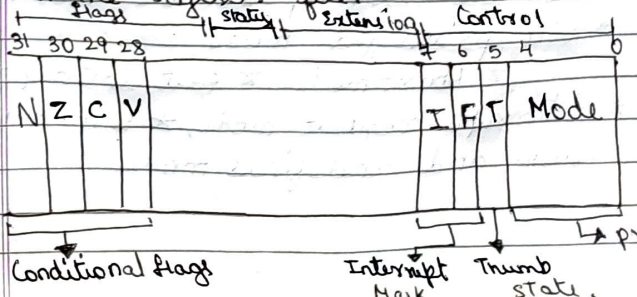
→ Register r15 is the program counter (PC) and contains the address of the next instruction to be fetched by the processor.

In ARM state the register r0 or r13 are orthogonal - any instruction that you can apply to r0 you can equally well apply to any of the other registers.

In addition to the 16 data registers, there are two program status registers : cpsr (current program status register) & spsr (saved program status register).

- Current Program Status Register.

→ The ARM core uses the cpsr to monitor and control internal operations. The cpsr is dedicated 32-bit register and resides in the register file.



The cpsr is divided into four fields, each 8 bits wide: flags, status, extension and control. In current designs the extension and status fields are reserved for future use.

→ The control field contains the processor mode, state and interrupt mask bits.

→ The flags field contains the conditional flags.

★ Processor Modes:-

The processor mode determines which registers are active and the access rights to the cpsr register file.

- A privileged mode allows full read-write access to the cpsr.
- A non-privileged mode only allows read access to the control field in the cpsr, but still allows read-write access to the conditional flags.

There are seven processor modes
→ Six privileged mode
★ Abort mode, ★ fast interrupt mode, ★ interrupt request mode, ★ supervisor mode, ★ system and undefined mode.

★ → Banked Registers:-

The below figure shows all 37 registers in the register file. Of these, 20 registers are hidden from a program at different times. These registers are called banked registers and are identified by the shading in the diagram.

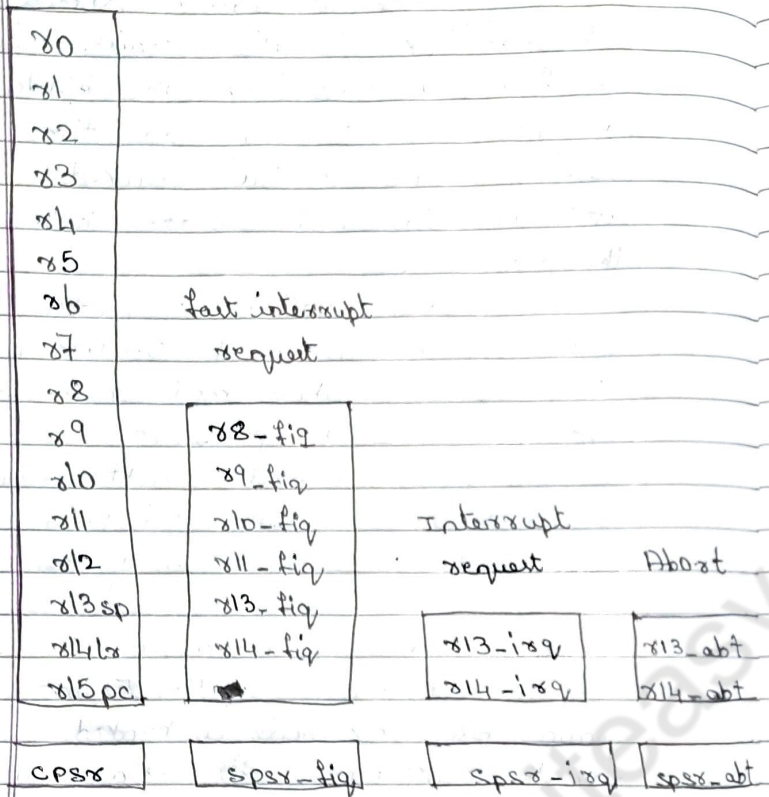
They are available only when the processor is in a particular mode. for example:- abort mode has banked registers r13_abt, spsr_abt.

All processor modes except system mode have a set of associated banked registers that are a subset of the main 16 register.

A banked register maps one-to-one onto a user mode register. If you change processor mode, a banked register from the new mode will replace an existing register.

For example:- when the processor is in the interrupt request mode, the instruction you execute still access named r13 & r14. However, these registers are the banked registers r13_irq & r14_irq. The user mode register r13_usr & r14_usr are not affected by the instruction referencing these register. A program still has normal access to the other register r0 to r12.

Page No.
Date / /



Processor Mode Table

Mode	Abbreviation	Privileged	Mode
Abort	abt	Yes	10111
fast interrupt Request	fiq	yes	10001
Interrupt Request	irq	yes	10010
supervisor	svc	yes	10011
System	sys	yes	11111
Undefined	und	yes	11011
User	usr	NO	10000

★ State and Instruction Sets:

- The state of the core determines which instruction set is being executed.

There are three instruction sets: ARM, Thumb & Jazelle.

The ARM instruction set is only active when the processor is in ARM state.

The Thumb instruction set is only when the processor is in Thumb state. Once in thumb state the processor is executing purely thumb 16-bit instructions.

The Jazelle J and Thumb T bits in the cpsr reflect the state of the processor

- When both J and T bits are 0, the processor is in ARM state and

www.takeitbayengineers.com

execute ARM instructions. This is the case when power is applied to the processor.

- When the T bit is 0, then the processor is in Thumb state.

To change states the core executes a specialized branch instruction.

The following Table compares the ARM & Thumb instruction set features.

ARM (CSPR T=0)	
Instruction size	32-bit
Core instruction	58
Conditional execution	most
Data processing instruction	access to barrel shifter and ALU
Program status register	read-write in privileged mode
Register usage	15-general-purpose registers + PC.

Thumb (CSPR T=1)	
Instruction size	16 bit
Core instruction	30
Conditional execution	Only branch instructions
Data processing instructions	separate barrel shifter & ALU instruction
Program status register	no direct access
Register usage	8 general-purpose register + 7 high register + PC.

The ARM designers introduced a third instruction called Jazelle. Jazelle executes 8-bit instruction and is a hybrid mix of software & hardware designed to speed up the execution of Java byte-codes.

Jazelle (CSPR T=0, J=1).	
Instruction size	8-bit
Core Instruction	Over 60% of the Java byte-code are implemented in hardware; the rest of the codes are implemented in software.

www.fullfreeasyengineers.com

Condition Flags :-

Condition flags are updated by comparisons and the result of ALU operations that specify the S instruction suffix.

Table.

Flag	Flag Name	Set when
Q	Saturation	the result caused an overflow and/or saturation.
V	Overflow	the result caused a signed overflow
C	Carry	the result caused an unsigned carry
Z	Zero	the result is zero
N	Negative	bit 31 of the result is a binary 1

These flags are located in the most significant bits in the cpsr. These bits used for conditional execution.

Conditional Execution :-

It controls whether or not the core will execute an instruction. Prior to execution, the processor compares the condition attribute with the condition flags in the cpsr. If they match, then the instruction is executed ; Otherwise the instruction is ignored.

Mnemonic	Name	Conditional flags
EQ	equal	Z
NE	not equal	z
CS HS	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	v
HI	Unsigned higher	zC
LS	unsigned lower or same	Z or c
GT	Signed greater than	NzV or nzv
LE	signed less than or equal	Z or Nv or nV
AL	always (unconditional)	ignored

www.takeiteasyengineers.com

PIPELINE.

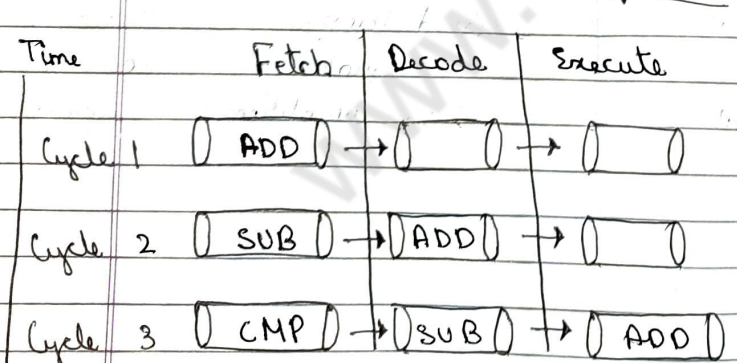
→ A pipeline is the mechanism in a RISC processor, which is used to execute instructions.

→ Pipeline speeds up execution by fetching the next instruction while other instruction are being decoded & executed.

Three stages pipeline ARM 7.

- Fetch loads an instruction from memory
- Decode identifies the instruction to be executed
- Execute processes the instruction & writes the results back to a register.

Pipelined Instruction Sequence.



- The three instruction are placed into the pipeline sequentially
- In the first cycle, the core fetches the ADD instruction from memory
- In the second cycle, the core fetches the SUB instruction and decodes the ADD instruction
- In the third cycle, both the SUB & ADD instruction are moved along the pipeline. The ADD instruction is executed, the SUB instruction is decoded, & the CMP instruction is fetched.
- This procedure is called filling the pipeline.
- The pipeline allows the core to execute an instruction every cycle.

As the pipeline length increases, the amount of work done at each stage is reduced which allows the processor to attain a higher & higher operating frequency. This in turn increases performance. The increased pipeline length also means increased system latency & there can be data dependency b/w certain stages.

www.learneasyengineers.com

The pipeline design for each ARM family differs.

ARM 9:- It has 5-stage Pipeline
→ The ARM9 adds a memory & write back stage, which allows the ARM9 to-
- process on average 1.1 Dhrystone MIPS per MHz.
- increases the instruction throughput in ARM9 by around 13% compared with an ARM7.

Fetch → Decode → Execute → Memory → write

ARM 10:-
It has 6-Stage Pipeline
→ It can process on average 1.3 Dhrystone MIPS per MHz.
→ have about 34% more throughput than an ARM7 processor core.
→ but again at a higher latency cost.

Even though the ARM9 & ARM10 pipeline is different, they still have same pipeline executing characteristics of an ARM7, Hence, code written for the ARM7 will execute on an ARM9 or ARM10.

Pipeline Executing Characteristics:

→ The ARM pipeline will not process an instruction, until it passes completely through the execute stage.

Example:- An ARM7 pipeline has executed an instruction only when the fourth instruction is fetched.

Characteristics of the pipeline:

→ The execution of a branch instruction or branching by the direct modification of the pc causes the ARM core to flush its pipeline.

→ ARM 10 uses branch prediction, which reduces the effect of a pipeline flush by predicting possible branches & loading the new branch address prior to the execution of the instruction.

→ An instruction in the execute stage will complete even though an interrupt has been raised. Other instruction in the pipeline will be abandoned and the processor will start filling the pipeline.

www.takeiteasyengineers.com

EXCEPTIONS, INTERRUPTS AND THE VECTOR TABLE.

→ When an exception or interrupt occurs, the processor sets pc to a specific memory address.

The address is within a special address range the vector table.

- The entries in the vector table are instructions that branch to specific routines designed to handle a particular exception or interrupt
- The memory map address 0X00000000 is reserved for the vector table, a set of 32-bit words.

THE VECTOR TABLE

Exception / Interrupt	Symbol	Address	High Address
Reset	RESET	0x00000000	0x00000000 - 00
Undefined instruction	UNDEF	0x00000004	0xffff0004
Software interrupt	SWI	0x00000008	0xffff0008
Prefetch abort	PABT	0x0000000c	0xffff000c
Data abort	DABT	0x00000010	0xffff0010
Reserved		0x00000014	0xffff0014
Interrupt Request	IRQ	0x00000018	0xffff0018
Fast interrupt Request	FIQ	0x0000001c	0xffff001c

→ Reset: Vector is the location of the first instruction executed by the processor when power is applied. This instruction branches to the initialization code.

→ Undefined: This instruction vector used when the processor cannot decode an instruction

→ Software interrupt: Vector is called when you execute a SWI instruction. The SWI instruction is frequently used as the mechanism to invoke OS routine

→ Prefetch abort → vector occurs when the processor attempts to fetch an instruction from an address without the correct access permissions. The actual abort occurs in the decode stage.

→ Data abort: vector is similar to a prefetch abort, but is raised when an instruction attempts to access data memory without the correct access permissions.

→ Interrupt request vector is used by external hardware to interrupt the normal execution flow of the processor. It can only be raised if IRQs are not masked in the cpsr.

→ Fast interrupt request vector is similar to the interrupt request, but is reserved for hardware requiring response time. It can only be raised if FIQs are not masked in cpsr.

www.takeitEasyengineers.com

CORE EXTENSION :-

Core extension are the standard hardware components placed next to the ARM core.

They improve performance, manage resources, and provide extra functionality and are designed to provide flexibility in handling particular applications.

Each ARM family has different extension available.

There are 3 hardware extension ARM wraps around the core

1. Cache and tightly coupled memory.
2. Memory Management.
3. Coprocessor interface.

1. Cache and Tightly Coupled Memory:

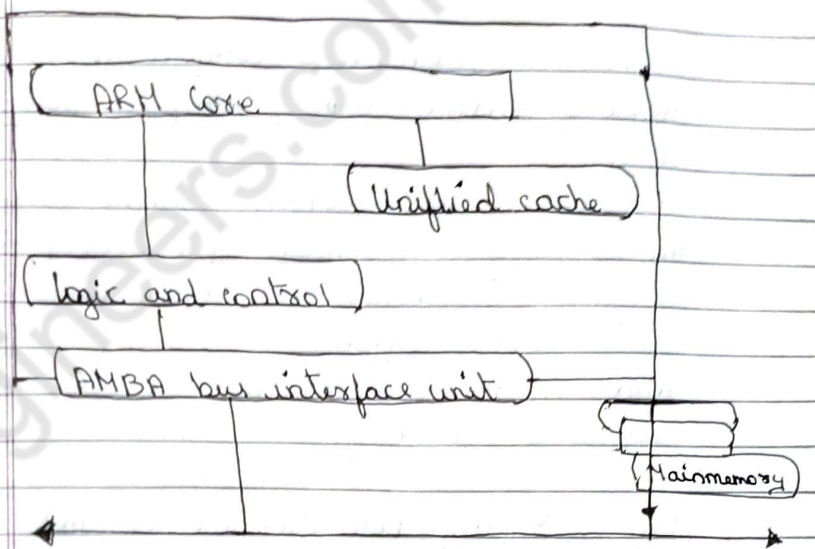
The cache is a block of fast memory placed b/w main memory and the core.

It allows for more efficient fetches from some memory types. With a cache the processor core can run for the majority of the time without having to wait for data from slow external memory.

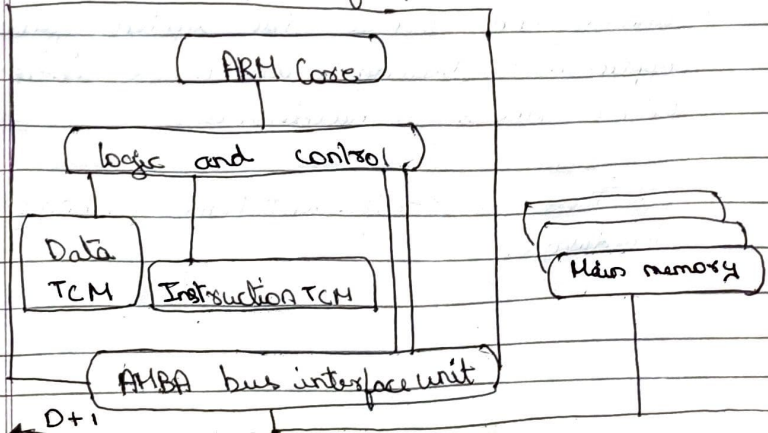
Most ARM-based embedded systems use a single-level cache internal to the processor.

ARM has two forms of cache:

1. Von Neumann-style core:
   → It combines both data and instruction into a single unified cache.



2. Harvard-style core.
   → It has separate caches for data & instructions
   → It increases overall performance but at the expense of predictable execution.



www.makeiteasyengineers.com

→ But the real-time systems require the code execution to be determined — the time taken for loading and storing instructions or data must be predictable.

→ This is achieved using a form of memory called tightly coupled memory (TCM).
→ TCM is fast SRAM located close to the core and guarantees the clock required to fetch instruction or data.
→ TCMs appear as memory in the address map and can be accessed as fast memory.

By combining both technologies, ARM processors can have both performance & predictable real-time response.

2. Memory Management.

→ Embedded systems often use multiple memory devices. It is usually necessary to have a method to organize these devices and protect the system from applications trying to make inappropriate access to hardware.

→ There are 3-different Memory Management hardware.

- no extensions providing no protection
  → Non protected memory is fixed & provides very little flexibility.
  → It is normally used for small, simple embbed systems that require no protection from rogue applications.

- a memory protection unit (MPU)
  → It employ a simple system that uses a limited number of memory regions.
  → These regions are controlled with a set of special registers, & each region is defined with specific access permission.
  → It is used for systems that require memory protection but don't have a complex memory map.

- a memory management unit (MMU)
  → These are the most comprehensive memory management hardware available on the ARM.
  → MMU uses a set of translation tables to provide fine-grained control over memory.
  → These tables are stored in main memory & provide a virtual-to-physical address map as well as access permissions.
  → Designed for more sophisticated platform OSs that support multitasking.

www.takeiteasyengineers.com

Coprocessors:

→ A coprocessor extends the processing features of a core by extending the instruction set or by providing configuration registers.
→ More than one coprocessor can be added to the ARM core via the coprocessor interface.
→ The coprocessor can be accessed through group of dedicated ARM instructions that provide a load-store type interface
→ The coprocessor can also extend the instruction set by providing a specialized group of new instructions.

These new instruction are processed in the decode stage of the ARM pipeline.

→ If the decode stage sees a coprocessor instruction, then it to the relevant coprocessor.

→ If it doesn't recognize the instruction, then the ARM takes an undefined instruction exception, which allows you to emulate the behaviour of the coprocessor in software.

Questions of Model Question Paper.

1. Discuss the ARM design philosophy
2. Discuss ARM bus technology
3. Explain Pipeline in detail
4. Write difference b/w RISC & CISC
5. Describe conditional execution. Write the different code suffix
6. Explain concept of exceptions, interrupts & vector table
7. Write difference b/w microprocessor & microcontroller.
8. Explain ARM core data flow model with diagram.
9. Write the four main hardware components of ARM based embedded device.
10. Explain different processor modes by ARM7
11. Explain Current program Status Register by briefing individual bits.
12. Write about RISC philosophy?

www.aketeasyengineers.com