System Design using MicroControllers:-

Frame at:-

Objective

*) System design using microcontrollers
*) Architectural features & Programming aspects of (ARM controller/Processors)
*) Memory management in system design Application

ARM Architectures
↓
ARM Programming model-1 (I.S)
↓
ARM Programming mode-2 (A.I.S)
↓
ARM Programming
↓
memory mangment

UNIT-1

ARM Architecture → ARM families
ARM Architecture → Design Philosophy
ARM Architecture → Registers
ARM Architecture → Pipeline
ARM Architecture → Interrupt & vector table

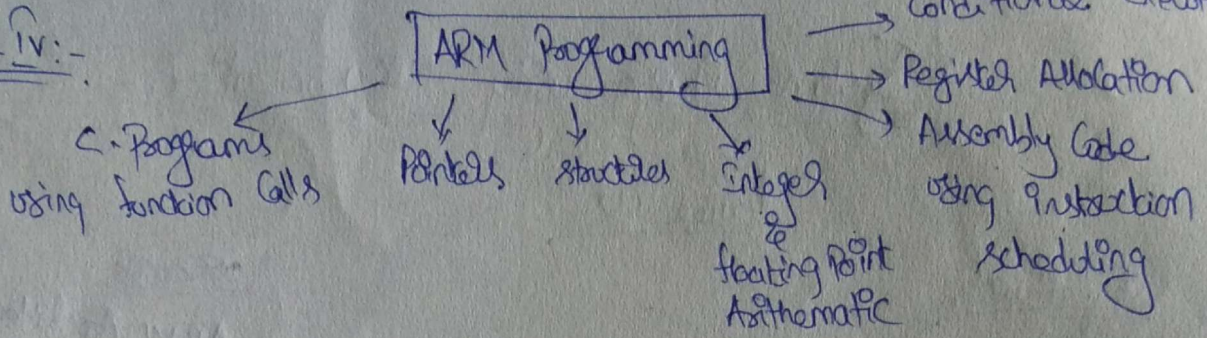
UNIT-II:-

ARM Programming Model-1
↓
Instruction set → Conditional instructions
Instruction set → Data Processing Instruction
Instruction set → PSR instructions
Instruction set → Branch
Instruction set → Load
Instruction set → Store instructions

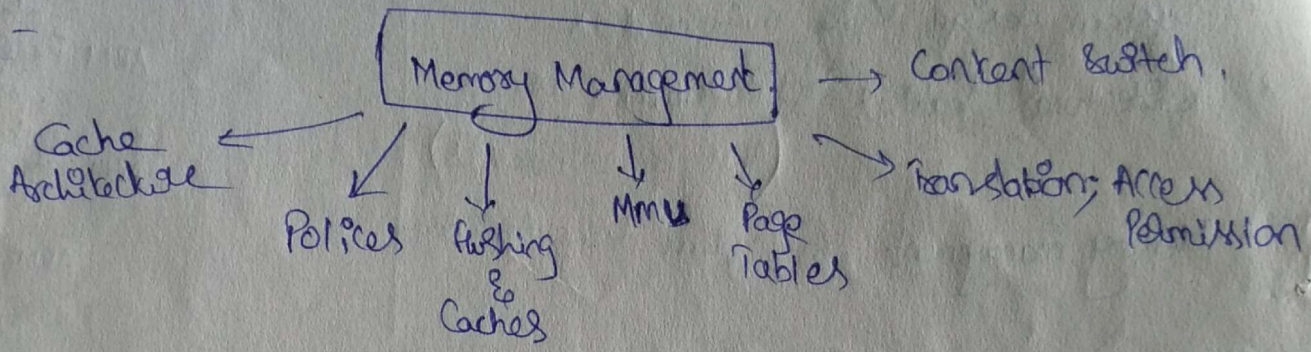
UNIT-III:-

ARM Processing Model-II
↓
Thumb Instruction set → Software Interrupt Instructions
Thumb Instruction set → Register usage
Thumb Instruction set → Stack
Thumb Instruction set → other Branch instruction
Thumb Instruction set → Data Processing instruction
Thumb Instruction set → single & multi Register load-store instructions

Unit - IV :-



Unit - V :-



System Design using MicroControllers

Unit - 1

Introduction :-

1) What is ARM ?:-

✓ → ARM originally Acorn RISC Machine, later known Advanced Reduced Instruction Set Computing (RISC) Machine (ARM) which is family of RISC architectures for Computer Processor. (1983 & 1985)

✓ → ARM makes 32-bit & 64-bit RISC multi-core processors.
→ RISC Processors are designed to perform a small number of types of Computer instructions so that they can operate at a higher speed, performing more "Millions of Instructions Per Second (MIPS). By stripping unneeded instructions and optimizing pathways.
→ ARM Processors are extensively used in Consumer electronics device such as smart phones, tablets, multimedia players .. etc.,

Features :-

✓ → load/store Architecture.
✓ → An orthogonal instruction set. → uses all addressing modes & instruction type & Addressing mode vary independently (orthogonal means)
✓ → Mostly single cycle execution
✓ → Enchanced power saving design
✓ → 64 & 32-bit execution states for scalable high performance
✓ → Hardware virtualization support.

2) ARM Embedded Systems :-

✓ The ARM Processor core is a key component of many successful 32-bit Embedded Systems, which are widely used in mobile phones, handled organizers, multitude of other everyday portable Consumer devices.

✓ → ARM Core is not a single core but whole family of design principles and a Common instruction set.

Example:- ARM's most successful core is ARM7TDMI. It provides upto '120 Dhrystone MIPS' and is known high core density and low Power Consumption, making it ideal for mobile embedded devices.

3) The RISC Design Philosophy:-

✓ → aimed at delivering simple but powerful instructions that execute within a single cycle at a high clock speed.

✓ → It Constraints on reducing the Complexity of instructions performed by hardware because it is easier to provide greater flexibility and intelligence in software rather than hardware.

✓ As a result RISC design places greater demands on the Compiler. In Contrast, the traditional complex instruction set Computer (CISC) relies more on the hardware for instruction functionality and Consequently the CISC instructions are more Complicated.

→ Implemented in 4-design rules:- Instruction (IPRL) → Pipelines, Registers, Load store Architectures.

① Instructions:-

✓ → RISC Processors have reduced no. of instruction types classes.

✓ → These classes provide simple operations that can each execute in single cycle.

✓ → The Compiler (or) Programmer synthesize Complicated operations by Combining several simple instructions.

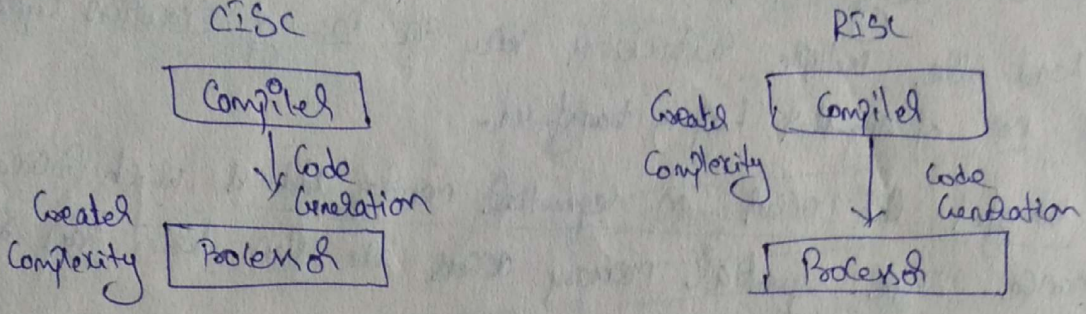
✓ → Each instruction is fixed length to allow the pipeline to fetch further instruction before decoding the current instruction.

② Pipelines:- The processing of instructions is broken down into smaller units that can be executed in parallel by pipelines.

→ Ideally pipelines advances by one step on each cycle for maximum throughput.

→ Instructions can be decoded in one pipline stage.

(3) Registers - RISC Machines have large no. of General Purpose Registers
→ Registers act as fast local memory store for all data processing operations.



(4) Load - Store Architecture :-
The processor operates on data held in Registers. Separate load and store instruction transfer data b/w the register bank and external memory.

4) The ARM Design Philosophy :-

✓ ARM Processor has been specially designed to be small to reduce Power Consumption and external battery operation.

bcoz :- 1st portable Embedded System require some form of battery Power. Application such as Mobiles & Personal digital Assistants (PDA)

✓→ High Code density is another major ~~density~~ requirement since Embedded System have limited memory due to cost / physical size restriction.

→ E.S are price sensitive and use slow and low cost memory devices.

→ To reduce area of the die taken up by Embedded processor.

✓→ ARM has incorporated hardware debug technology within the processor so that Software Engineers can view Processor & the Processor is executing code.

✓→ ARM Core is not pure RISC architecture bcoz of the Constraints of Primary applications - the Embedded System.

5) Instruction Set for Embedded Systems:-

⇒ Variable Cycle Execution for Certain instructions:-

✓ Not every ARM instruction executes in a single cycle.

Eg:- Load store multiple instructions vary in no: of execution cycle depending upon no:- of register being transferred.

✓ → The transfer can occur on sequential memory address, which increases performance since sequential memory accesses are often faster than random accesses.

→ Code density also improved since multiple register transfer are common operations at start and end of functions.

⇒ Inline Barrel shifter leading to more complex instructions:-

✓ The inline barrel shifter is a hardware component that preprocesses one of ip register before it is used by instruction.

✓ → It expands capability of many instructions to improve core performance and code density.

⇒ Thumb 16-bit instruction set:-

→ ARM Enhanced the processor core by adding a second 16-bit instruction set called Thumb that permits the ARM core to execute either 16/32-bit instructions

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

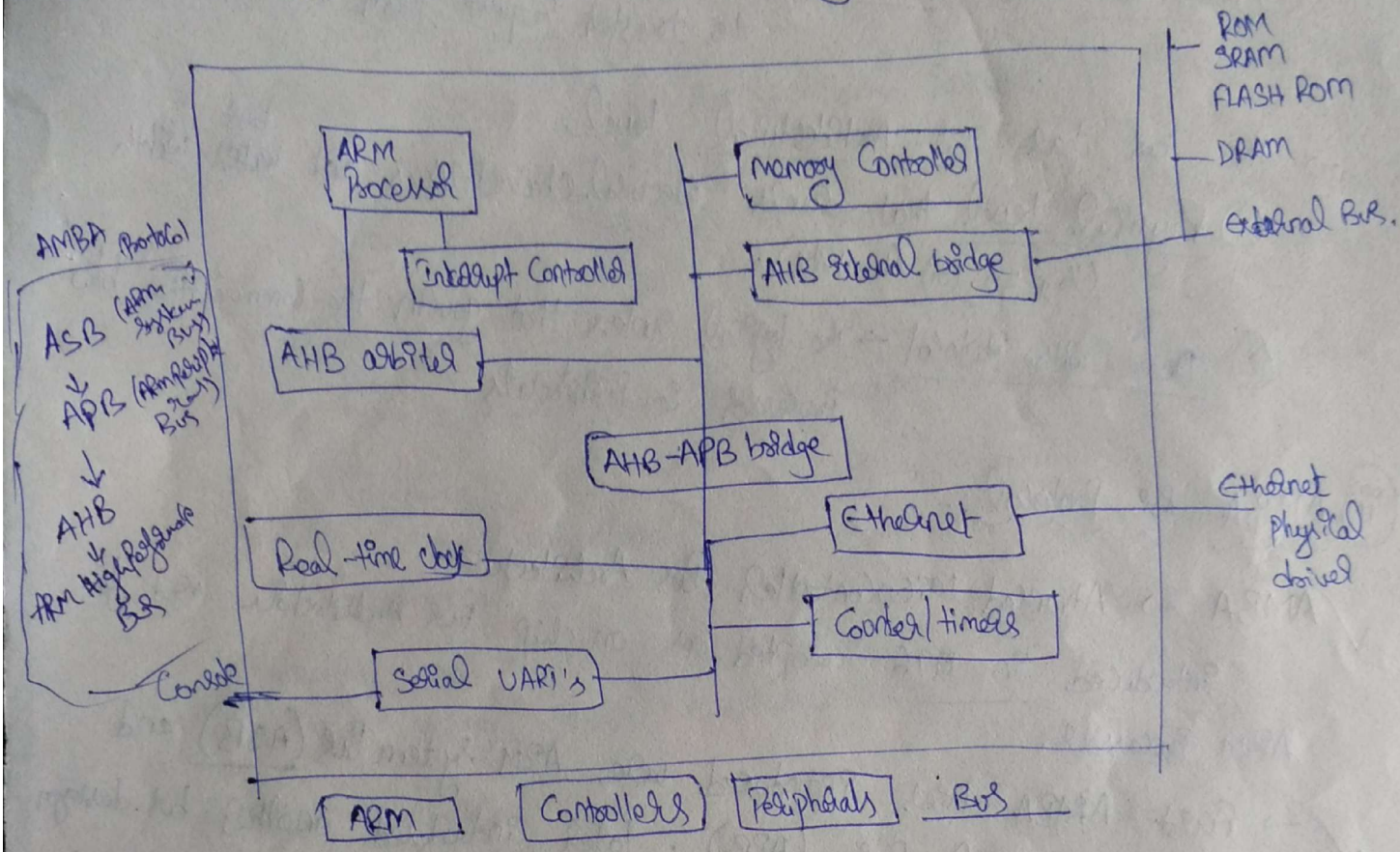
⇒ Conditional execution:- An instruction only executed only when specified condition has been satisfied.

⇒ Enhanced instructions:- Enhanced DSP instructions were added to standard ARM instruction set support fast 16×16-bit multiplier operations and saturation.

Embedded System Hardware :-

" " Can Control many different devices from small sensors found on production line, to real time Control systems.

All devices use Combination of Hardware and Software Components.



Eg:- ARM based Embedded device, a micro Controller :-

* ARM Processor Controls the Embedded device. Different version of ARM Processor are available to suit the desired operating characteristics.
* Controllers Co-ordinate important functional blocks of the System. Two Commonly found Controllers are interrupt and memory Controllers.
* Peripherals Provides i/p & o/p Capability external to chip
* ARM Bus Technology :- E.S use different bus technologies than those designed for X86 PC's. the most Common PC bus technology, the Peripheral Component Interconnect (PCI) bus, Connects such devices as Video Card and hard disks Controllers to the X86 Processor bus. This type is called External (or) off-chip. and built in to mother board of PC.

There are 2 different classes of devices attached to the bus.

→ ① ARM Processor Core is bus master - a logical device capable of initiating data transfer with another device attached to the bus.
access same bus.

② Peripherals tend to bus slaves - logical devices capable only of responding to transfer request from a bus master device.

→ The bus has 2 - Architectural level
① Physical level that covers electrical characteristics and bus width (16, 32 (or) 64 - bits).
② Deal with Protocol → The logical rules that govern the communication b/w Processor & Peripherals

(11) AMBA Bus Protocol:-

✓ AMBA → Advanced MicroController Bus Architecture introduced in 1996 adopted as on-chip bus Architecture used for ARM Processors.

✓ → First AMBA buses introduced were ARM System Bus (ASB) and ARM Peripheral Bus (APB). later introduced another bus design ARM High Performance Bus (AHB).

→ AMBA peripheral designers can reuse the same design on multiple Projects.

✓ → AHB provided higher data throughput than ASB bcoz it is based on Centralized multiplexed bus scheme. rather than ASB bidirectional bus design.

✓ → AHB bus to run at higher clock speeds and to be the first ARM bus to support width of 64 & 128 bits.

✓ → ARM introduced 2 variations on Bus → Multilayer AHB
→ AHB lite.

9/30/5

→ In Contrast original AHB allows single bus master to be active on bus at any time.

✓ → Multilayer AHB bus allows multiple active bus masters.

✓ → AHB lite is a subset of AHB bus and it is limited to a single bus master.

✓ → AHB & multilayer AHB support same protocol for Master & slave but have different interconnect.
↳ They permit operations in Parallel and allow for higher throughput rates.

(iii) Memory :-

ES has to have some form of memory to store and execute code.

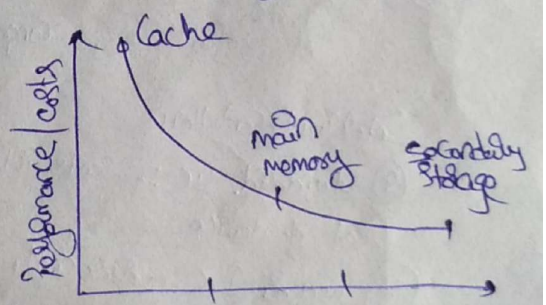
Compute: Price, Power Consumption, Performance.

characteristics: Hierarchy, width, type.

If memory has to run twice as fast to maintain a desired bandwidth then the memory power requirement may be higher.

(a) Hierarchy :-

The Cache is placed b/w main memory and the CPU used to speed up data transfer b/w Processor and main memory.



→ Cache provides an overall increase in Performance but with a loss of predictable execution time.

→ main memory size around (256 KB to 256 MB) depending on application

(b) width :-

The memory width is no:- of bits the memory return on each access typically 8, 16, 32, 64 bits.

If an embedded system using 32-bit ARM instruction and 16-bit wide memory chip, the Processor have to make 2-memory fetches for instruction., Each requires 16-bit loads.

this obviously has effect on system performance.

→ If the Core executes 16-bit thumb instructions it will archive better performance with 16-bit memory.

(c) Types :-

instruction size	8-bit memory	16-bit memory	32-bit memory
ARM 32-bit	4-Cycles	2-Cycle	1-Cycle
Thumb 16-bit	2-Cycles	1-Cycle	1-Cycle.

Different types of memory

ROM → is least flexible of all memory types & contains image that is permanently set at production time and can not be reprogrammed.

Flash ROM → can be written/Read but it is slow to write

DRAM (Dynamic RAM)

SRAM (Static RAM) → on Power up memory controller allow to be active. (allow to initialize the code to be executed)

(d) Peripherals :- → memory controller

→ interrupt controller. → Normal IC → Priority

→ Vectoring IC → Priority, single interrupt dispatch.

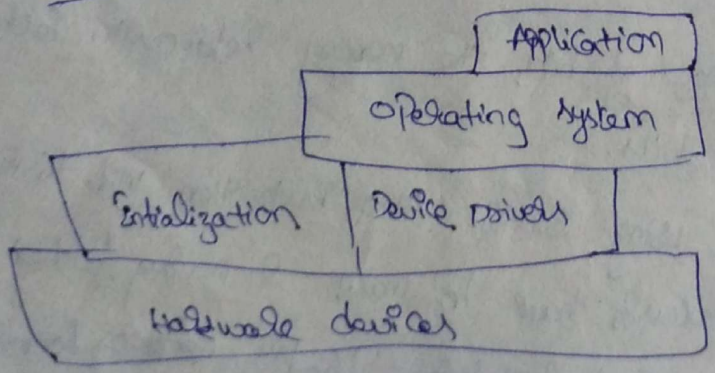
A peripheral range from simple serial comm device to a more complex 802.11 wireless device.

interrupt controllers :-

(i) Standard IC :- Sends interrupt signal can be programmed to ignore (or mask any individual (or) set of device

(ii) vector I.C :- Associate a Priority and a handler address to each interrupt

(7) Embedded System Software :-



(8) 1 DataPath :-

→ Data enters the Processor through the Data bus.

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

→ ARM Processors like all RISC Processors, uses a load-store architecture ie 2-instruction types are transferring data.

(i) Load instruction Copy data from memory to registers in the Core &

(ii) Store " " " " " Reg → memory.

→ Data items are placed in Reg file - a storage bank made up of 32-bit Reg

∴ ARM Core is 32-bit Processor

↳ most instruction treat the registers as holding signed & unsigned 32-bit values.

(i) The sign extend hardware converts 8-bit & 16-bit (signed) → 32-bit value as they read from memory and placed in Reg.

(ii)

→ ARM instructions typically have 2-source Registers "Rn & Rm" and a single result & destination register, Rd.

↳ Source operands are read from register file using internal buses A & B.

→ ALU (Arithmetic logic Unit) & MAC (multiply-accumulate unit) takes the Reg values Rn & Rm from the A & B buses. and computes a result.

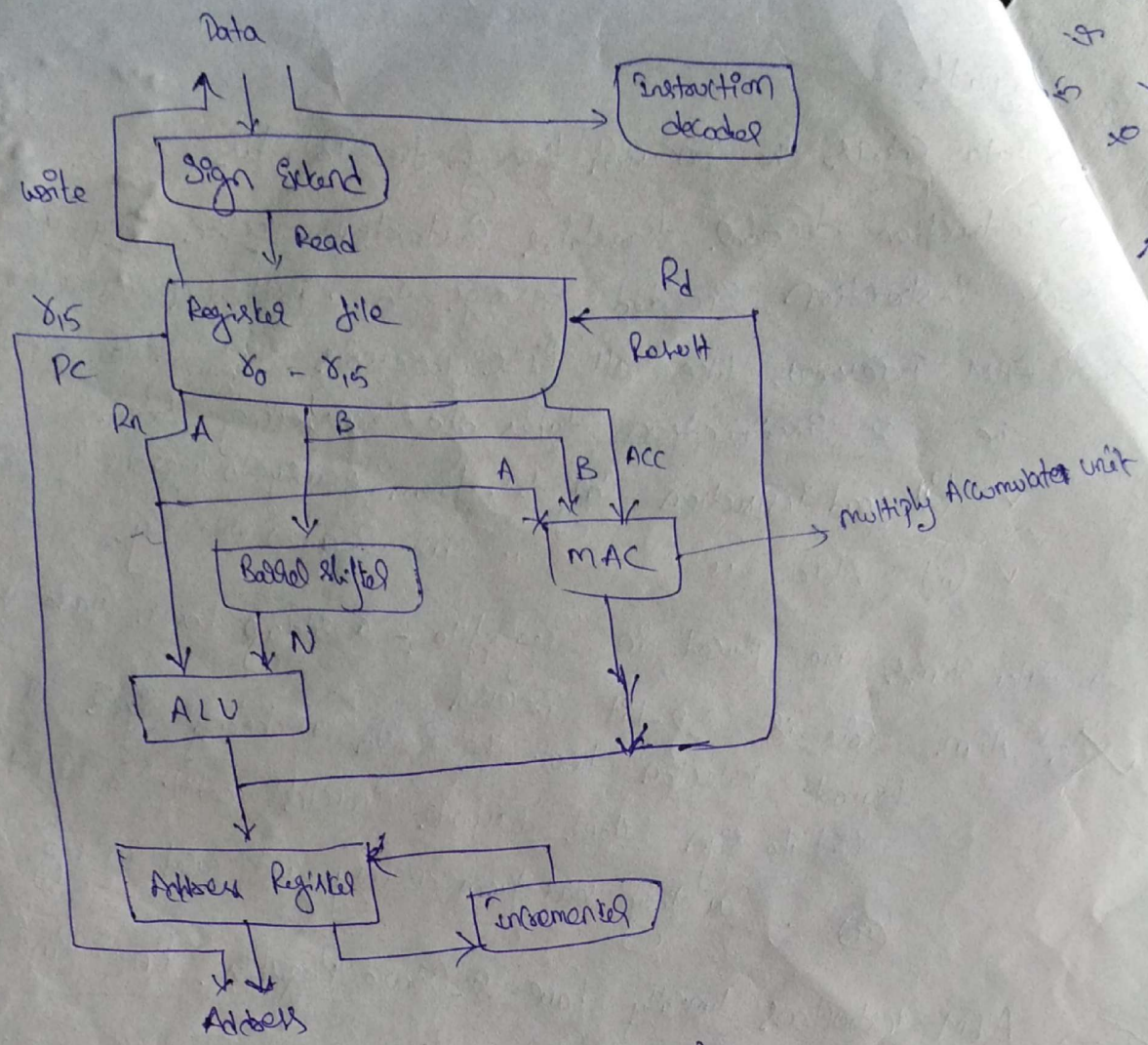
→ Load & Store instructions use the ALU to generate an address to be held in address Register and broadcast on address bus

→ Imp feature in ARM is Register Rm alternately pre-processed in barrel shifter before it enters the ALU.

→ Both barrel shifter & ALU can calculate wide range of Expression and Address. after passing through functional unit the result in Rd is back in Reg file using Result bus.

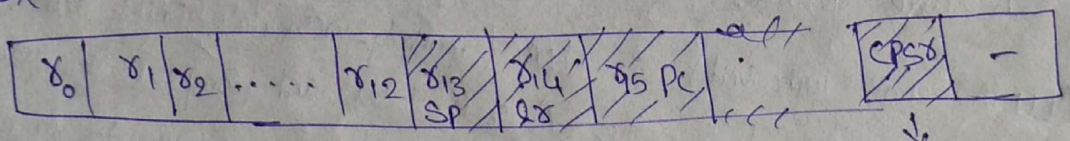
→ Barrel shifter is combinational ckt which shift data to left/Right by arbitrary no. of position in same cycle & Rsdy.

→ Inc/decrement logic can update register content for sequential access independent of ALU.



(9) **Registers:-** (holds either data (or) Address)

CPSR → Saved Program SR



Current Program Status register

→ All Reg are of 32-bit size

→ There are upto 18-Active registers: 16-data registers & 2-process status reg. ($r_0 - r_{15}$)

→ ARM Process has 3-registers assigned to particular task (or) special functions r_{13}, r_{14}, r_{15}.

✓ → r_{13}, r_{14}, r_{15} & CPSR are used for special-purpose Reg.

✓ → r_{13} is traditionally used as stack pointer (SP) and stores the head of stack in Current process mode.

✓ → r_{14} is called the link register (LR) and it whole code puts the return address when it calls subroutine.

r_{15} is Program Counter (PC) Contains the address of next instruction to be fetched by Processor.

→ In ARM registers ($r_0 - r_{13}$) are orthogonal — any instruction that you apply for 'r_0' you can equally well apply to any of that register.

✓ 2-Program status reg: CPSR & SPSR.
(Current & Save Program Status registers).

(10) Current Program Status Register:-

→ ARM Core use CPSR to monitor and control internal operations.

✓ → It is 32-bit Register.

→ It shows basic layout of generic Program status register.

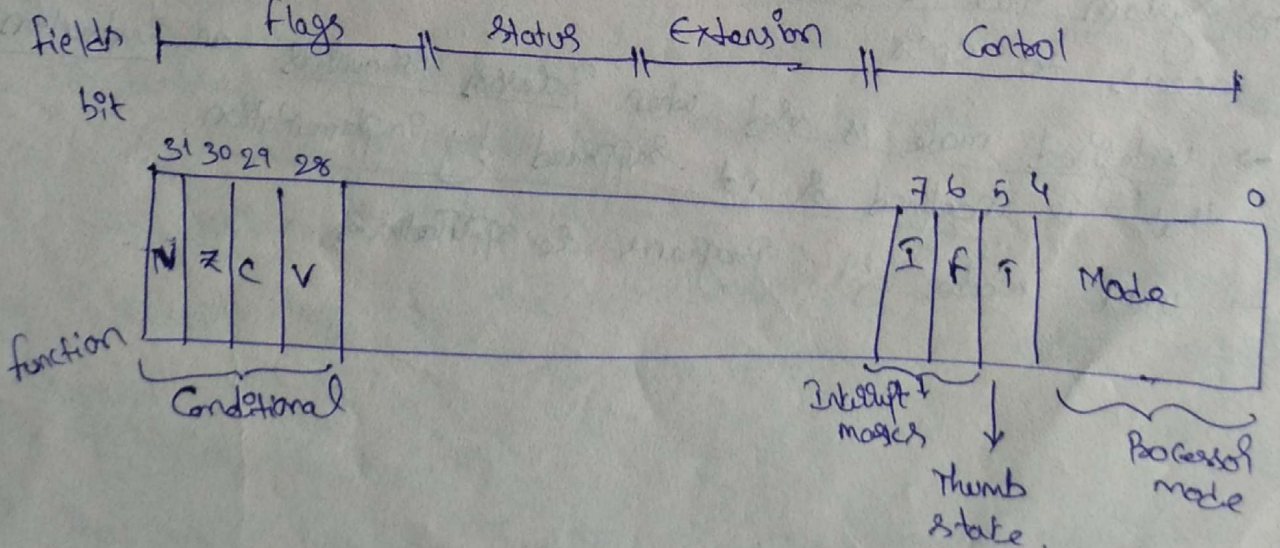
✓ → Divided in 4-fields each 8-bit wide:-
↳ Flags ; Status ; Extension and Control.

✓ → In current design extension and status field used for future use

✓ → The Control field contains Processor mode, State and interrupt mask bit.

✓ → The flag field contains conditional flags.

✓ → ARM Processor Core have extra bit allocated. for Eg:- J-bit which can be found in flag field is only available on Jazelle-Enabled Processors. which execute 8-bit instruction.



(a) Processor mode :-

→ determines which register is active and access right to cpsr.
Each processor mode is privillaged (or) non privilliaged.

↳ A privilliaged mode allow full read - write access to cpsr.

↳ non " " " condition flags

→ There are total 7-processors in that

↳ 6- privilliaged mode (abort ; fast ; interrupt ; request ; interrupt request ; supervisor ; system and undefined)

↳ 1- non privilliaged mode (user).

→ the processor enters abort mode when there is a failed attempt to access memory.

→ fast interrupt & interrupt request & interrupt request modes correspond to 2- interrupt levels available on ARM processor.

→ ① supervisor mode that the processor is in after reset and in ~~processor in Abort mode~~ ~~mode is failed attempt to access memory~~.

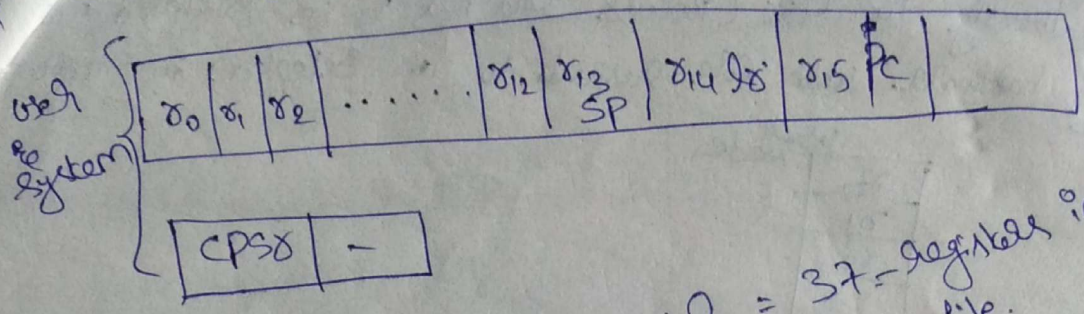
~~→~~ generally mode that an os kernel operates.

→ ② system mode is special version of user mode allows full read-write access to cpsr.

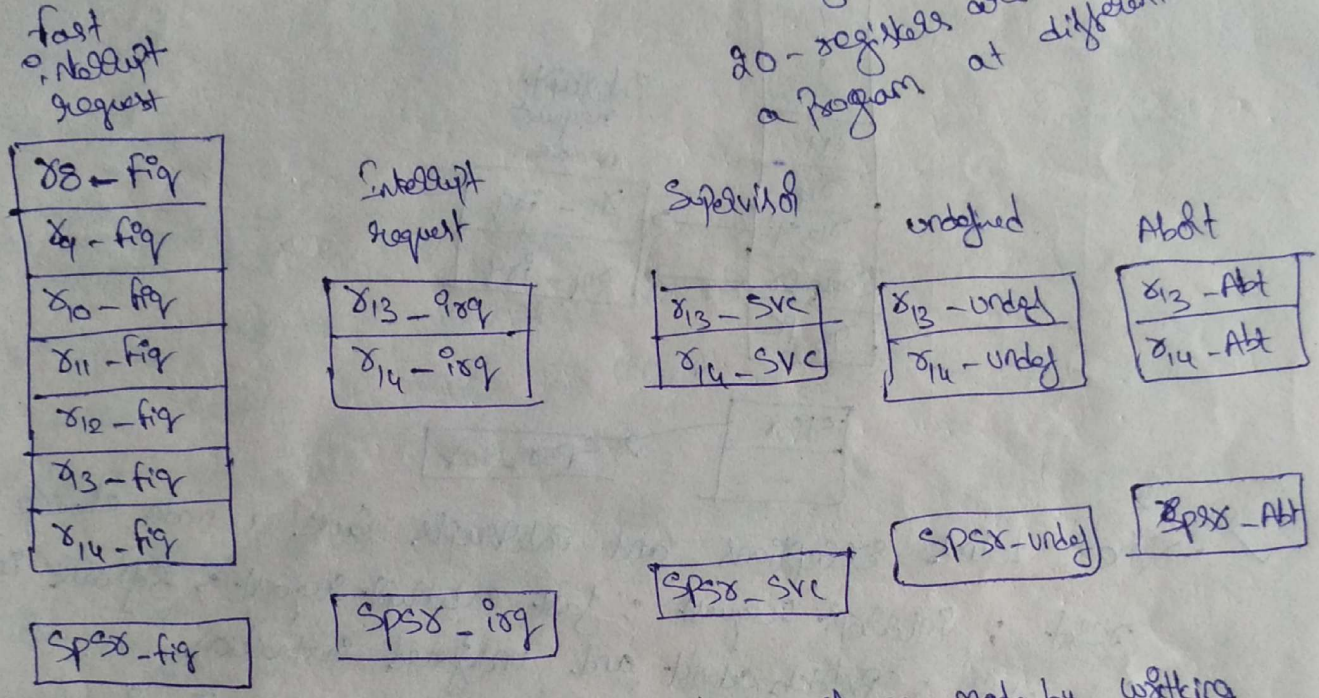
→ undefined mode is used when processor encounters an instruction that is undefined or not supported by implementation.

→ user mode for programs & applications

Linked Registers :-



total = 37 registers in register file.
20 registers are hidden from a program at different times.



→ Every processor mode except user mode can change mode by writing directly to the mode bit of CPSR.

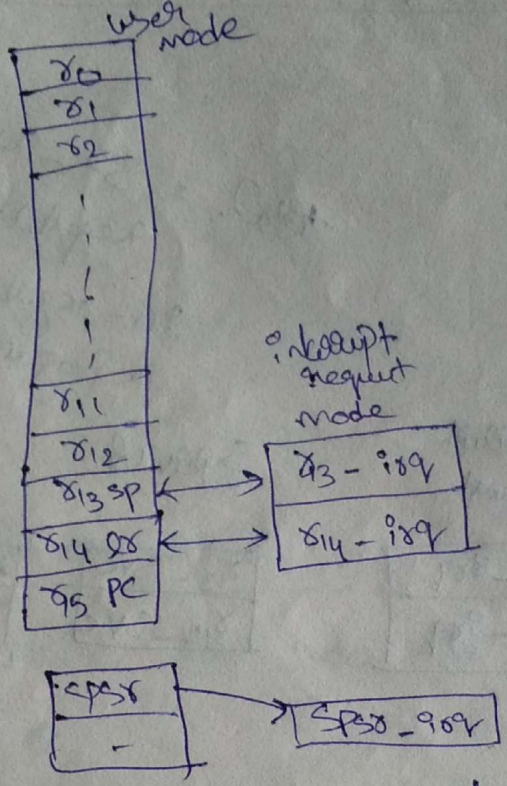
→ All processor mode except system mode have a set of associated bank registers that are a subset of main 16 registers.

⇒ The banked reg map one-to one onto a user mode register.
i.e when processor mode changed banked reg change new mode will replace an existing reg

→ Interrupt req mode, the instructions you execute still access registers named r_{13} & r_{14}. Coz these registers are banked registers
r_{13} - irq & r_{14} - irq.

→ The user registers r_{13}-usr & r_{14}-usr are not affected by instruction referencing these registers. A program still has normal access to the other registers r_0 to r_{12}

→ The processor mode can be changed by a program that writes directly to CPSR (Processor Core has to be privileged mode) & by hardware when Core responds to an exception or interrupt.



→ The terms exceptions and interrupts cause a mode change as follows reset ; interrupt request ; fast interrupt request ; software interrupt ; data abort ; Prefetch abort and undefined instruction.

→ The user registers are replaced with registers r_{13}-irq and r_{14}-irq respectively.
while r_{14}-irq contains the return address and r_{13}-irq contains the stack pointer for interrupt request mode.

→ The saved program status register (spsr) which stores the previous modes cpsr. which appears a new register in interrupt request mode.

→ To return back to user mode, a special return instruction used that instruct the Core to restore the original cpsr from spsr_irq and bank in user registers r_{13} & r_{14}.
↳ spsr can only be modified and read in a privileged mode. There is no spsr available in user mode.

Mode	Abbreviation	Privileged	mode (4:0)
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

→ Note:- cpsr is not copied into spsr when a mode change is forced due to a program writing directly to cpsr.

→ the saving of cpsr is occured only when an exception (or) interrupt is raised.

→ the current active processor mode occupies 5-least significant bits of cpsr. when power is on it is supervisor mode which is privileged.

↳ starting in " mode is useful. since initialization code can use full access to cpsr to setup the stacks for each of other modes.

(c) State and instruction set:-

→ state determines which instruction set is being executed.

→ there are 3-instruction sets → ARM ; Thumb ; Jazelle

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

↳ Thumb " " " " " " " " Thumb ".

↳ once in thumb state the processor is executing purely Thumb 16-bit instructions.

↳ the JazelleR-J and ThumbT-bits in the cpsr reflect the state of processor.

→ when both J-bit & T-bit are zero(0) the Processor is in ARM state and execute ARM instructions. this the case when Power applied to

→ when T-bit is ('1') Processor is in thumb state.

→ to change state the core executes specialized branch instructions.

S.no	ARM & Thumb Instruction set features.	ARM (cpsr T=0)	Thumb (cpsr T=1)
1	Instruction size	32-bit	16-bit
2	Core instruction	58	30
3	Conditional execution	most	only branch instructions
4	Data processing instructions	access to barrel shifter and ALU	separate barrel shifter and ALU instructions
5	Program status registers	Read-write in privileged mode	no-direct access
6	Register usage	15-general purpose registers +pc	8-general-purpose registers +7 high register +pc

→ 3rd instruction Jazelle. executes 8-bit instruction and is a hybrid mix of software and hardware design to speed up the execution of Java bytecodes.

↳ to execute Java byte code requires Jazelle technology plus a special modified version of Java virtual machine.

↳ Hardware portion of Jazelle only support a subset of the Java byte codes; the rest are emulated in software.

Jazelle instruction set features

	Jazelle (cpsr T=0 ; J=1)
Instruction size	→ 8-bit
Core instructions	→ over 60% of Java bytecode are implemented in hardware; the rest of codes are implemented in software.

& Interrupt Masks:-

→ Used to stop specific interrupt request from interrupting process

there are 2-types interrupt request levels

ARM Processor core
- Interrupt request (IRQ)
- Fast Interrupt Request (FIQ)

✓ → The cpsr has 2-interrupt mask bit 7 and 6 (or I and F) which control the masking IRQ and FIQ

✓ → The I bit masks IRQ when set to binary '1', and
F bit " FIQ " " " " '1'.

(e) Conditional flags:-

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

Eg:- SUBS Subtract instruction results in a register value of zero then Z-flag in cpsr is set.

i.e this subtract instruction specifically update the cpsr.

Conditional flag:-

Flag	flag name	set when
Q	Saturation	the result causes an overflow and/or saturation
V	Overflow	the result causes signed overflow
C	Carry	the result causes an unsigned carry
Z	Zero	the result is zero frequently used to indicate Equality
N	Negative	bit 31 of the result is binary 1

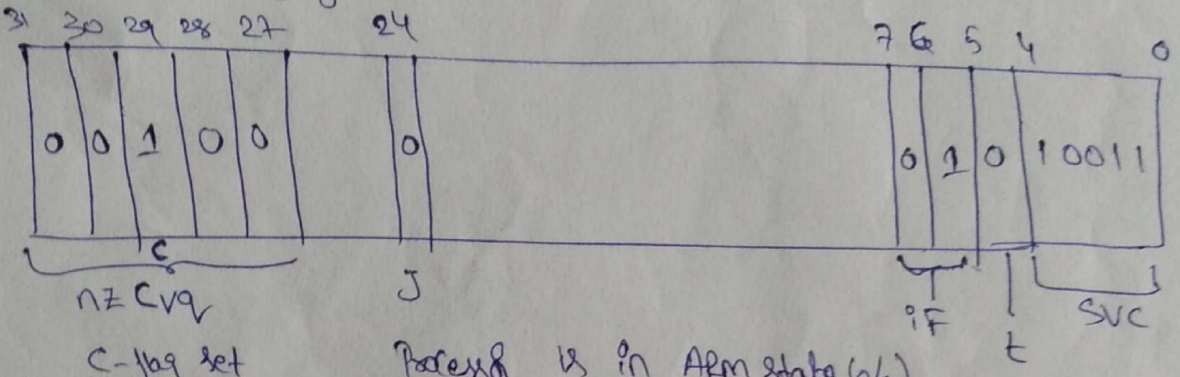
→ when processor cores to include DSP extensions, the Q-Bit indicates if an overflow (or) saturation has occured in Enhanced DSP instruction
↳ The flag is "sticky" in the sense that hardware set the flag
↳ To clear flag you need to write to cpsr directly.

Jazelle - Enabled Processor J-bit reflect state of core if it is in the Core is in Jazelle state.

↳ J-bit generally not usable & available on some Processor Cores.

→ Most ARM instructions can be executed conditionally on the value of condition flag.

↳ These flags are located on most significant bit on cpsr.



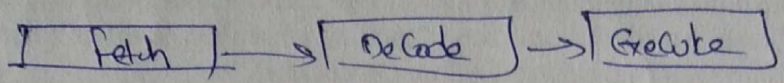
(F) Conditional Execution:- Cortex, whether (or) not core will execute an instruction.

Conditional Mnemonics:-

mnemonic	Name	Conditional flag
EQ	Equal	Z
NE	not Equal	$\bar{Z}$
CS HS	Carry set / unsigned higher (or) same	C
CC LO	Carry clear / unsigned lower.	$\bar{C}$
MI	Minus / negative	N
PL	Plus / Positive or zero	$\bar{N}$
VS	overflow	V
VC	no overflow	$\bar{V}$
HI	unsigned higher	$Z\bar{C}$
LS	unsigned lower or same	Z (or) $\bar{C}$
GE	signed greater than or Equal	NV or $\bar{N}\bar{V}$
LT	signed less than	$N\bar{V}$ or $\bar{N}V$
GT	signed greater than	$N=V$ (or) $\bar{N}=\bar{V}$
LE	" less than or Equal	Z or $N\neq V$ (or) $\bar{N}=V$
AL	always (unconditional)	ignored.

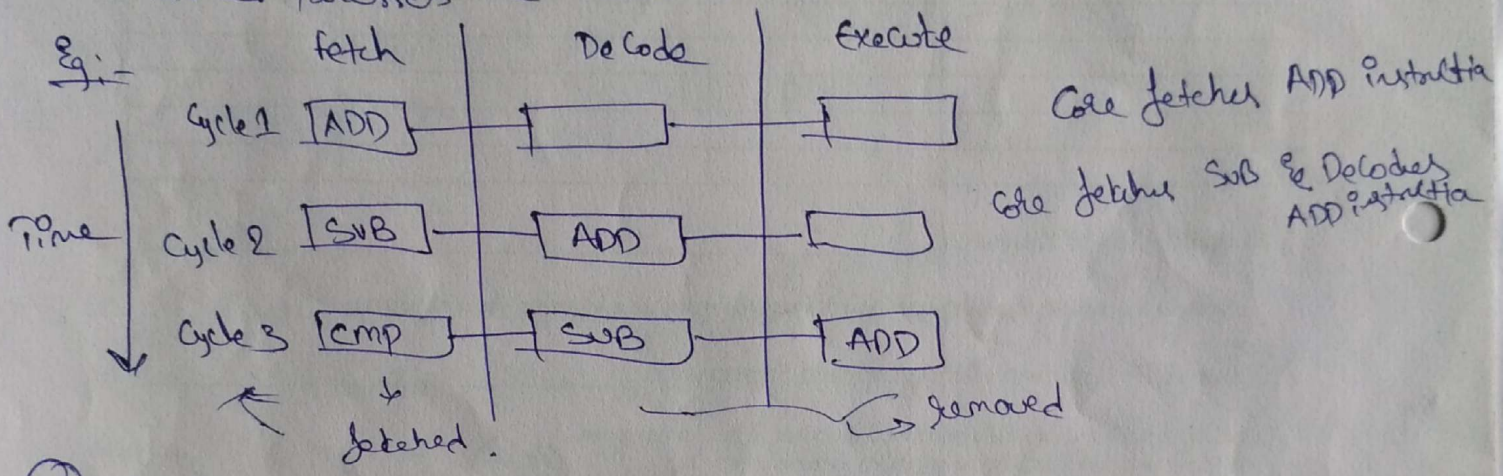
(11) Pipeline :-

→ Is the mechanism a RISC Processor uses to execute instructions
→ using a pipeline speed up execution by fetching the next instruction while other instructions are being decoded and executed.



→ fetch loads an instruction from memory
→ DeCode identifies the instruction to be executed.
→ Execute processes the instruction and writes the result back to a register.

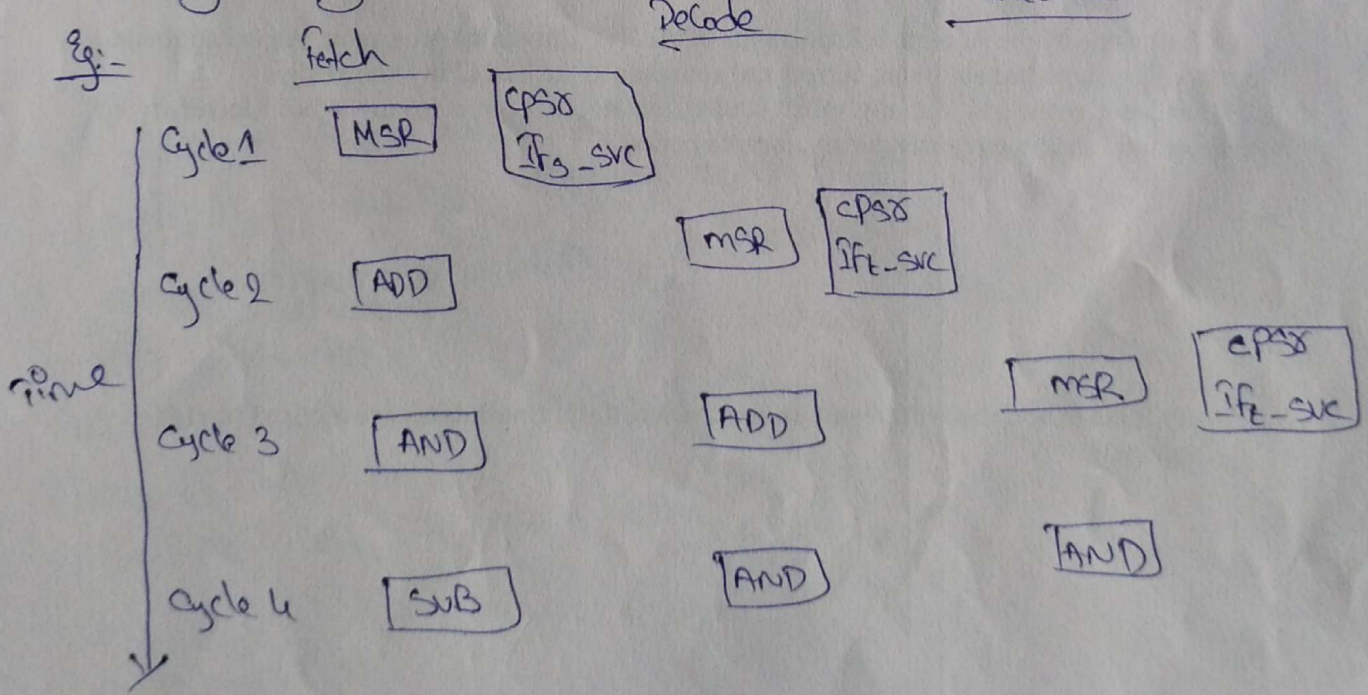
Eg:-



(12) Pipeline Executing characteristics :-

The ARM pipeline has not processed an instruction until it passes completely through the execute stage.

Eg:-



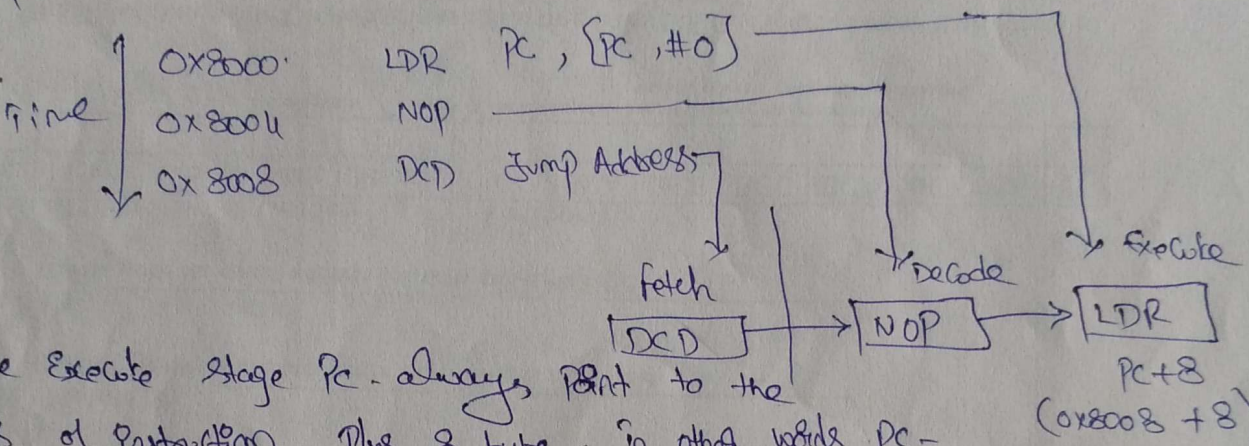
g - shows Instruction sequence on a ARM-7 Pipeline.

The MSR instruction (move to Status Register from an ARM Register)(cpsr to spsr) is used to Enable IRQ interrupts which only occurs once the MSR instruction completes the execute stage of Pipeline.

↳ It clears I-bit in the cpsr to Enable the IRQ interrupt

→ once ADD instruction enters the Execute stage of Pipeline, IRQ interrupt are Enabled.

Eg:-2.



→ In the Execute stage PC always point to the address of instruction plus 8-byte. In other words PC-always Points to address of instruction being executed plus two instructions ahead.

Note:- when the Processor is in Thumb state the PC is the instruction address plus 4.

→ 1st Execution of a branch instruction (8) branching by the direct modification of the PC causes the ARM Core to flush its pipeline.

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

→ IIIrd Instruction in execution stage will complete even though an interrupt has been raised. other instruction in pipeline will be abandoned and Processor will start filling the pipeline from the appropriate entry in vected table.

Exceptions, Interrupts & Vector Tables:-

When exception (or) interrupt occurs the Processor sets the PC to a specific memory address.

Where the address is within a special address range called Vector table.

* The memory map address 0x00000000 is reserved for the vector table, a set 32 bit words. On some processors the vector table can be optionally located at a higher address in memory (starting at the offset 0xFFFF0000). Operating systems such as Linux and Microsoft Embedded products can take advantage of this feature.

→ When an Exception (or) interrupt occurs the Processor suspended normal execution and starts loading instructions from the exception vector table. Each vector table entry contain a form of branch instruction pointing to start of a specific routine.

Vector Table:-

Exception/interrupt	Shorthand	Address	High Address
Reset	RESET	0x00000000	0xFFFF0000
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 first instruction executed by the Processor when Power is applied. This instruction branches to initialization code

→ Undefined instruction vector is 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 an OS routine.

Prefetch abort vector occurs when the Processor attempts to fetch an instruction from an address without correct access permission. 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 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, are not masked in cpsr.

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

(13) Architecture Revisions :-

Early ARM Processor implementation executes a specific instruction set architecture (ISA).

(a) Nomenclature :-

ARM {x} {y} {z} {T} {D} {M} {I} {E} {J} {F} {-S}

x - family
y - memory management / protection unit
z - cache
T - Thumb 16-bit decoder
D - JTAG debug
M - fast multiplier
I - Embedded ICE macrocell
E - Enhanced instructions (assumes TDMI)
J - Jazelle
F - vector floating point unit
S - Synthesizible version.

Architecture Evolution :-

Introduced in 1985 from original version (1-6). are of most significant change to ISA was introduction of Thumb instruction set in ARMv4T (the ARM7TDMI Processor).

(b) ARM Processor families :-

Families are based on ARM7 ; ARM9 ; ARM-10 ; ARM-11 cores.

7, 9, 10, 11 indicates core design

Revision history :-

Revision	Example core implementation	ISA Enhancement :-
(1) ARMv1	ARM 1	First ARM Processor 26-bit addressing
(2) ARM v2	ARM 2	32-bit multiplier 32-bit Co-processor support
(3) ARM v2a	ARM 3	on-chip Cache Atomic Swap instruction
(4) ARMv3	ARM6 and ARM7DI	32-bit addressing Seperate spsr & cpsr New modes - undefined instruction and abort MMU support - virtual memory
(5) ARMv3M	ARM7M	Signed and unsigned long multiply instructions
(6) ARMv4	Strong ARM	Load-Store instructions for signed and unsigned halfword/bytes New mode - system Reserve SWI space for architecturally defined operations 26-bit addressing mode no longer supported
(7) ARMv4T	ARM7TDMI & ARM9T	Thumb
(8) ARMv5TE	ARM9E & ARM10E	Superset of ARMv4T Extra instructions added for changing state b/w ARM and Thumb Enhanced multiply instructions Extra DSP-type instructions Faster multiply accumulate.
(9) ARMv5TEJ	ARM7EJ & ARM926EJ	Java acceleration
(10) ARMv6	ARM11	Improved multiprocessor instructions unaligned and mixed endian data handling new multimedia instructions.

ription of CPSR:-

Parts	Bits	Architectures	Description
Mode	4:0	all	Processor mode
T	5	ARMv4T	Thumb State
I & F	7:6	all	interrupt mask
J	24	ARM V5TEJ	Jazelle State
Q	27	ARM V5TE	Condition flag
V	28	all	"
C	29	all	"
Z	30	all	"
N	31	all	"

(d) ARM family attribute comparison:-

	ARM7	ARM9	ARM10	ARM11
(1) Pipeline depth	3-Stage	5-Stage	6-Stage	8-Stage
(2) Typical MHz	80	190	260	335
(3) mW/MHz	0.06mw/MHz	0.19mW/MHz (+Cache)	0.5mW/MHz (+Cache)	0.4mW/MHz (+Cache)
(4) MIPS/MHz	0.97	1.1	1.3	1.2
(5) Architecture	Von-Neumann	Harvard	Harvard	Harvard
(6) Multiplier	8x32	8x32	16x32	16x32

ARM7 family:-

→ Core has von-Neumann style architecture, with both data & instructions use the same bus.

→ Core has 3-stages pipeline & executes the architectured ARM v4T instruction set.

fetch
↓
Decode
↓
Execute

→ one significant variation on ARM-7 family is ARM7TDMI-S.

→ the ARM7TDMI-S has same operating characteristics as standard ARM7TDMI

→ ARM720T is most flexible member of ARM7 family bcoz includes MMU.

→ The Extensive MMU means the ARM720T is capable of handling the linux & microsoft Embedded Platform operating systems, which includes 8K Cache.

↳ relocatable can be relocated to a higher address by setting a Co-Processor 15-register.

→ Another variation is ARM7EJ-S Processor also synthesizable.

→ ARM7EJ-S is quite different since it includes a five-stage pipeline and executes ARMv5TEJ instructions.

→ This version of ARM7 is the only one that provides both Java acceleration and Enhanced instructions but without any memory Protection.

CPU Core	MMU/MPU	Cache	Jazelle	Thumb	ISA	E[a]
ARM7TDMI	none	none	no	Yes	V4T	no
ARM7EJ-S	none	none	Yes	Yes	V5TEJ	Yes
ARM720T	MMU	Unified-8K Cache	no	Yes	V4T	no
ARM920T	MMU	Separate-16K/16KD+I Cache	no	Yes	V4T	no
ARM922T	MMU	Separate 8K/8K D+I Cache	no	Yes	V4T	no
ARM926EJ-S	MMU	Separate Cache & TCMs configurable	Yes	Yes	V5TEJ	Yes
ARM940T	MPU	Separate-4K/4K D+I cache	no	Yes	V4T	no
ARM946E-S	MPU	Separate-Cache and TCMs configurable	no	Yes	V5TE	Yes

M966E-S	none	separate - ICMS Configurable	no	yes	v5TE	yes
ARM1020E	MMU	separate - 32K/32K D+I Cache	no	yes	v5TE	yes
ARM1022E	mmu	separate - 16K/16K D+I Cache	no	yes	v5TE	yes
ARM1026EJ-S	MMU & MPU	separate - Cache and ICMS configurable	yes	yes	v5TE	yes
ARM1136J-S	MMU	separate - Cache and ICMS configurable	yes	yes	V6	yes
ARM1136JF-S	MMU	separate Cache & ICMS Configurable	yes	yes	V6	yes

(15) ARM-9 Family:- announced in 1997

→ Consist of 5-stage pipeline the ARM-9 Processor can run at higher clock frequencies than ARM7. Extra stages improve performance of Processor.

→ the memory system has been redesigned to follow Harvard Architecture which separates data D & instructions I buses.

→ 1st Processor in ARM9 family was ARM920T includes separate D+I Cache. and an MMU. This Processor can be used by O.S requiring virtual memory support.

→ ARM922T is a variation on the ARM920T but with half the D+I cache size.

→ ARM940T includes a smaller D+I Cache and a MPU.

↳ ARM940T is designed for application not for O.S.

→ Both ARM920T and ARM940T execute the architecture V4T instruction.

ARM9E-S. This Core is Synthesizable version of ARM9 with Extension E. There are 2-variants both executes architecture of V5TE. instructions

ARM946E-S ARM966E-S

which also support optional Embedded trace macrocell (ETM)

→ ARM946E-S includes ICM (tightly Coupled memory), Cache, and an MPU, which this Processor is designed for used in Embedded Control applications. that require deterministic real-time response. ; ARM966E do not have MPU.

→ ARM926EJ-S is latest Core in ARM9 Product line. is announced in 2000, which was designed for use in small Portable Java-Enabled devices such as 3G phones and Personal digital Assistants (PDAs)

→ ARM926EJ-S is 1st ARM Processor Core to include the Jazelle technology accelerates Java byte Code execution. which features MMU, D+I Cache; ICMs, and non-zero wait states memories.

(16) ARM 10 family :- announced in 1999

It extends the ARM9 pipeline to 6-stages supports optional vector floating point (VFP) unit, which adds 7-stages to ARM 10 pipeline

→ The VFP increases floating point Performance and is Compliant with the IEEE 754.1985 floating point standard.

→ ARM1020E is 1st Processor to use an ARM10E Core. like the ARM9E includes E- Enhanced instructions. It has seperate 32K D+I Caches; optional vector floating point unit and an MMU.

↳ It also has 64-bit dual bus interface

→ ARM1026EJ-S is very similar to ARM926EJ-S but with both MPU and MMU. This Processor has the Performance of ARM10 with flexibility of ARM926EJ-S.

ARM11 family :-

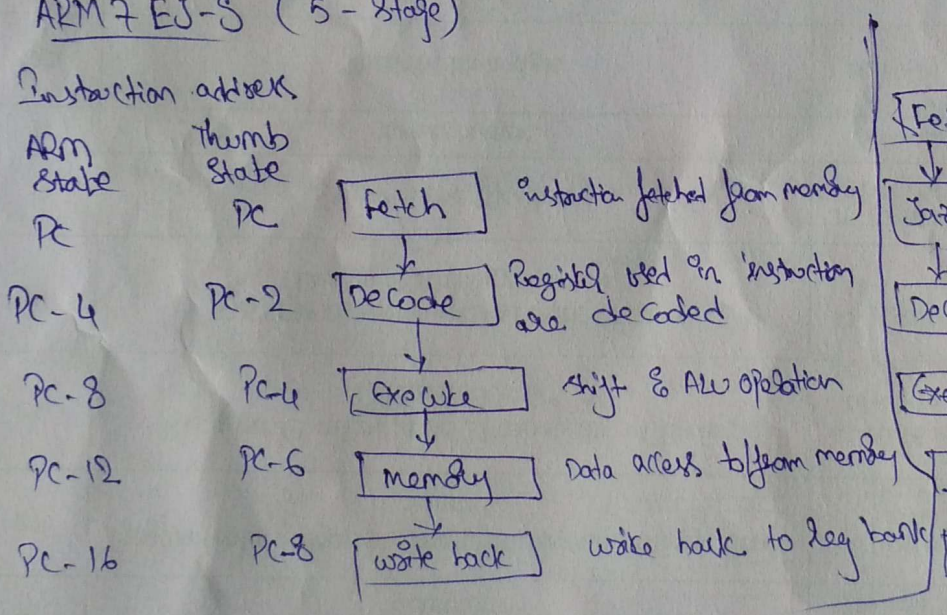
-> ARM 11 36J-S, announced in 2003 designed for high performance and power efficient applications

-> It is 1st Processor implementation to execute architecture ARMv6 instructions.

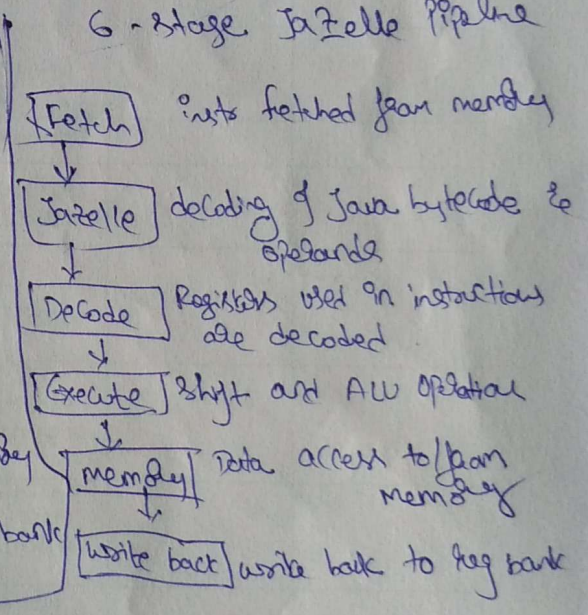
-> It incorporates 8-stages pipeline with separate load store and arithmetic pipelines included in ARMv6 instructions are single instructions multiple data (SIMD) extensions for media processing specifically designed to increase video processing performance.

The ARM 1136JF-S is an ARM1136J-S with addition of vector floating point unit for fast floating point operations.

ARM7EJ-S (5-stage)



6-stage Jazelle pipeline



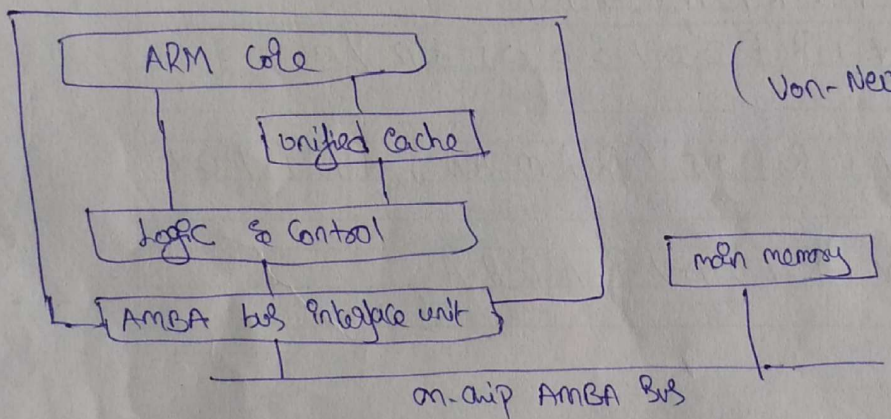
ore Extensions :-

Hardware Extensions placed next to ARM core improve performance, manage resources ; Provide extra functionality.

3- Hardware Extensions
- ↳ Cache
- ↳ tightly coupled memory (TCM)
- ↳ memory managment (mmu)
- ↳ co-Processor interface.

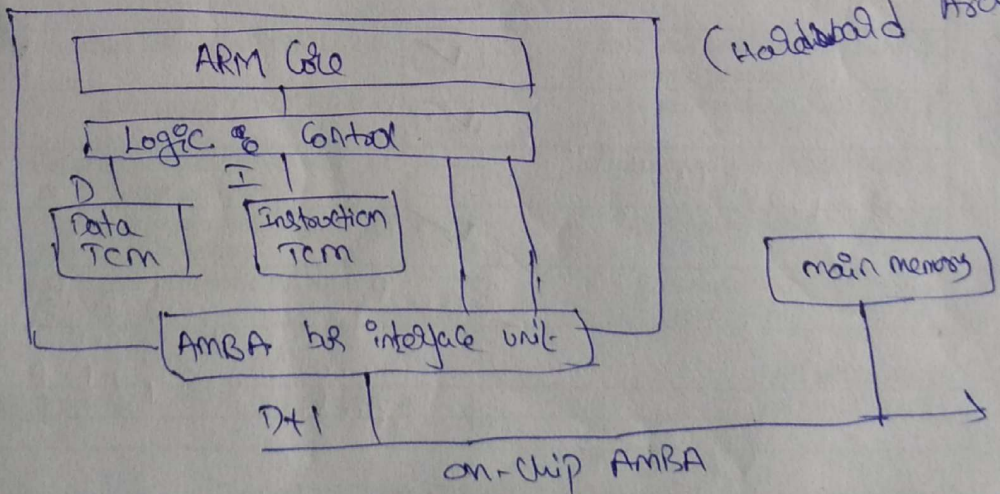
(1) Cache & tightly Coupled Memory :-

The Cache is a block of fast memory placed b/w main memory & core.



(Von-Neumann Architecture with Cache).

ARM has 2-forms of Cache. First is found attached to Von-Neuman style Cores. It Combines both data and instruction in to a single unified Cache.



(Harvard Archi)

Second form is Harvard-Architecture Seperates Cache & data ; Instruction

Cache Provides an overall increase in Performance but expense of Predictable execution.

-> But in real time system it is paramount that code execution is deterministic the time taken for loading and storing instructions (&) data must be Predictable This is achived using a form of memory called TCM (tightly coupled memory).

→ TCM is fast SRAM located close to core guarantees the clock cycles required to fetch instructions (&) data - critical for real time algorithms requiring deterministic behaviour.

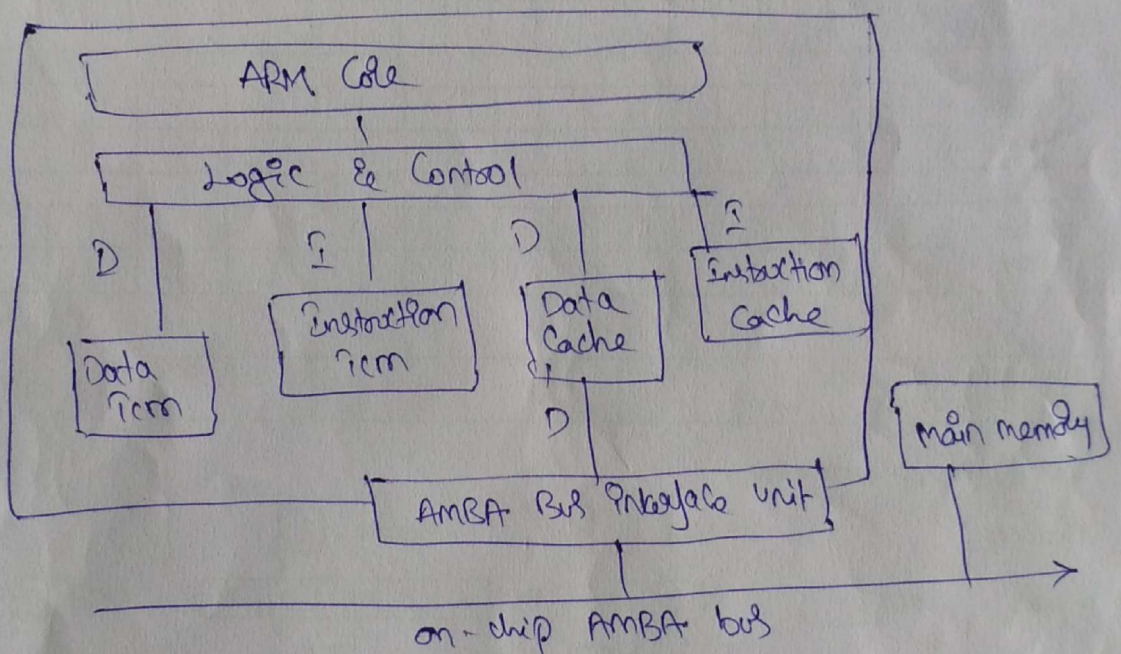


fig:- Combining of both technologies

(ii) Memory Managment:- ARM Core have 3 different types of mm-hdw ore no-Extensions Providing ① no Protection; a ② memory Protection unit (MPU) Providing limited protection and a ③ memory mangement unit (MMU) Providing full Protection.

① Non Protected Memory:- is fixed and Provides very little flexibility used for small, simple e.s that require no protection.

→ Memory Protection Unit :- Employ simple system that use limited no. of memory regions which are controlled by set of special co-processor registers, and each region is defined with specific access permissions.

→ Memory management unit (MMU):- do most comprehensive memory managment hardware available on the ARM

→ It uses set of translation tables to provide fine grained control over memory. these tables are stored over main memory and provide a virtual to physical address map as well as access permissions.