ARM7 Exceptions - Interrupts

* ARM7 interrupts, also called exceptions.
There are 7 intr. They make the processor to enter different modes. Seven intrs used to enter 5 Modes. To enter system mode and user mode you do not need any interrupt. So five modes are invoked by seven intrs.

Int	Mode	Low vectors	High vectors
Reset	SVR	0000 0000 H	FFFF 0000 H
Undefined	UND	" 0004 H	" 0004
SWI	SVR	" 0008	" 0008
Prefetch Abt	ABT	" 000C	" 000C
Data Abt	ABT	" 00D0	" 0010
IRQ	IRQ	" 0018	" 0018
FIQ	FIQ	" 001C	" 001C

* When Reset interrupt is called processor goes to supervisor mode. In SVR bois is stored. After calling reset bios is started and after few second it goes to user mode. (restart operation).

* When we are in a program if at that time 'Undefined' is called

it goes to ~~UND~~ 'Undefined' mode where the running operation is being killed. Then back to user mode.

* When we want to change ~~any~~ ~~f~~ ~~prof~~ ~~this~~ anything in system function. we call swi. calling swi, we goes user mode to SVR mode to change anything in system.

Both swi and reset works on SVR mode. Reset starts the bios to turn on the device. swi is used to modify something in system. (Eg: installing any app)

* Prefetch ABT and Data Abt both work on 'ABT' mode.

when user wants to access system program and system data, the ABT mode

comes in the picture.
Then it killed the application
and back went back to
user mode.

* There are two types of
ABT intrs.

case-1: when ~~yr~~ application tries
to access program in system
then the intr is called
prefetch ABT.

case-2: when application tries
to access ~~the~~ ~~fat~~ system data
then the intr is called
Data ABT.

There are two ~~he~~ external hardware
interrupt. IRQ and FIQ.
when IRQ is activated it goes to
IRQ mode
when ~~it goes to~~ FIQ interrupt
occurs it goes to FIQ mode.

* When any interrupt occurs processor goes to corresponding ~~IR~~ ISR. If the ISR has fixed address then the interrupt is called vectored interrupt.

* All interrupts in ARM are vectored interrupt.

all the interrupts have fixed ISR address. But location 0x00000014 to 0xFFFF0014 is empty. it was kept empty for further development.

* The vectors stores the address of ISR. It does not store the ISR itself.

So, for every interrupt there is one branch of instr for each of the ~~IR~~ ISR.

One instr size is 32 bit = 4 Byte.

That's why the gap from one vector to another is 4.

An interesting thing is that there is no branch instruction for FIQ. The exact ISR of FIQ is stored from location 0000 001CH. to as many bytes are required to store ISR of FIQ.

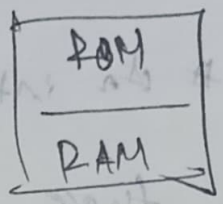
As there is no branch for FIQ so it is the fastest from all of ints.

There are is another way of vector. which is started from FFFF 0000 H. That means we can store vectors ISR address from 0000 0000 H location ors FFFF 0000 H location.

It is the programmers choice to from which location the define will store address of ISR.

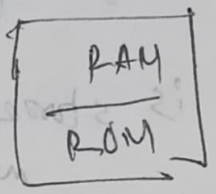
There is a reason behind it.

In some school of thought



ROM in higher address
low address in RAM (8051)

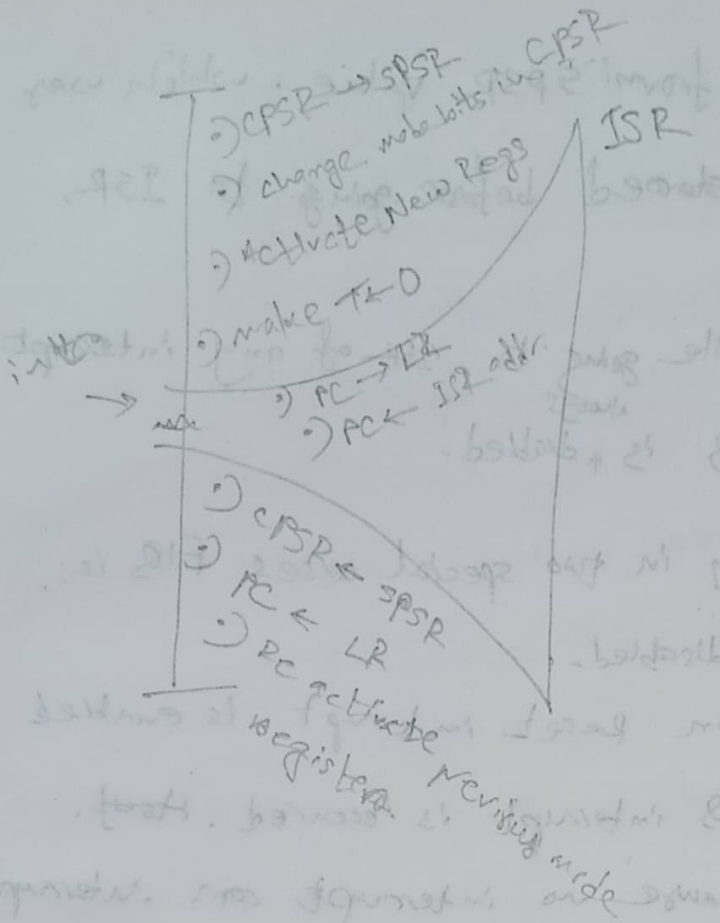
in some school of thought.



RAM in higher address
ROM in lower address (8086)

ROM is not volatile. So ISR address is stored in ROM. So ARM gives the freedom of memory (ROM+RAM) positioning to the programmers.

When an interrupt occurs pc stores its current value to LR. Then it takes the address of ISR for corresponding interrupt and finally it goes to ISR location. and execute the interrupt.



→ first, CPSR of current mode ~~in SPSR~~ is stored in ~~SPSR~~ SPSR.

→ then change CPSR bit value according to the new mode.

→ Activate new registers according to the new mode.

→ The current operation before intr may ~~can~~ be in normal state or may be ~~it~~ in thumb state. But the ISR is always in thumb state. So before going ISR location. we have to make thumb bit 0. for ensuring normal state for ISR operation.

→ while back from ISR, the state (normal / thumb) is restored

from SPSR value. which was stored before going to ISR

while going to ISR of any interrupt IRQ is always disabled.

Only in two special cases FIQ is ~~da~~ disabled.

i. when Reset interrupt is enabled

ii. FIQ interrupt is occured. ~~Itself~~. because no interrupt can interrupt itself.