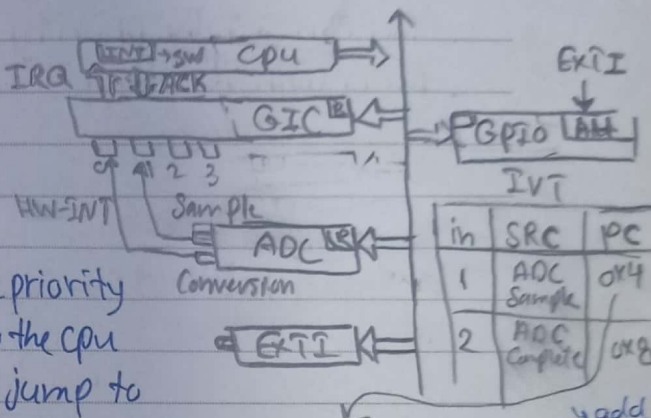
Date:

Subject:

Interrupts

- for every INT ⇒ ISR
- There is a fixed location in memory that holds the address of the ISR.
- The table of memory locations set aside to hold the addresses of ISRs is called the Interrupt Vector Table.

INT ↳ SW ⇒ inside cpu, Segmentation Fault
↳ HW ⇒ outside cpu, Signals, wires



Steps: 1- The peripheral generates IRQ to the GIC
2- GIC decides its masked or not, check the priority then generate IRQ with INT Number to the cpu
3- The cpu will reply with INTA then will jump to specific address based on the INT Number according to IVT on the ROM.
4- This address on the ROM will have branch instruction to the ISR definition in RAM
5- Then cpu will execute the ISR
6- Then come back to the last instruction in main.

add 0x4 : branch ISR_ADC (add y)
32 bit inst

You don't write ISR at 0x4 in order not to overwrite 0x8 who has the add of the another assembley inst to another ISR

0x4: branch add of ISR

How to put function in specific add? ⇒ Embedde C, linker Script

ISR: - each ISR has a default implementation in the system startup code
- All ISRs are declared as weak in the startup code.
- weak: mean that another non-weak subroutine with the same name defined elsewhere can override this one.

Servicing INT: → Vectored Arbitration System
↳ Non-vectored Priority System

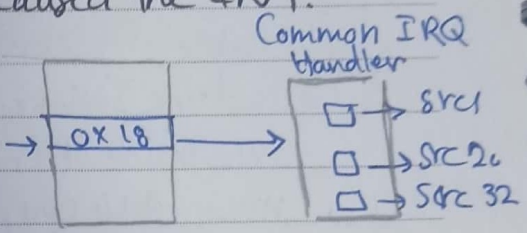
Date:

Subject:

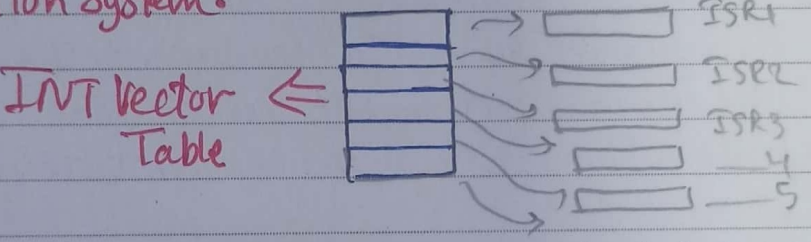
Non_Vectored priority System:

- When INT occurs: pc jumps to specific add.
- At this add you check sequentially what caused the INT.
- Very slow
- You can set the priority
- MCU with less than 5 INTs.

```
ISR()
{ if(TMR0)
  {
  }
elsif(ADC)
  {
  }
}
```



Vectored Arbitration System:



IRQ Number	Vector add	IRQ SRC

An INT Number is used as an index into the IVT to locate the corresponding ISR.

Efficient cpu use, has priority, can be masked

- polling vs INT →
  - If INT happens rapidly ⇒ polling ⇒ Adv / DisAdv of polling
  - ↳ Takes cpu time even with no request
    - "Ties down the cpu"

Interrupt Latency: Time between interrupt occurs until ISR begin.

Sequential INT: if an INT occurs during executing ISR of another INT finishes the current ISR first even if it's lower priority.

Nested INT: The opposite of Sequential INT as it stops executing the current ISR and finishes the higher priority ISR then go back.

Date:

Subject:

Types of INTS: - Asynchronous / HW INT : external
- Synchronous / Exceptions

Processor: Faults, Traps, Aborts

Programmed → Special instruction

Faults: - Writting to readonly memory segment.
Illegal Inst - Reading from an unavailable memory segment.
. Detected before incrementing the PC.

Traps: - Cpu is programmed to automatically switch control to a debugger after it has executed an instruction. "Break point"
. Activated after incrementing the pc.

Error Exceptions: - Divide by zero, Invalid operation, illegal memory reference