Key capabilities needed to support multitasking

- **Memory subsystems for instructions storage and some RAM for execution**

  If you don't have multiprocessing then you don’t really need a memory subsystem and can use a cheaper chip such as the Arduino .

- **Interrupt Controller**

  We need to gather , prioritize and control the generation of interrupts to the processor

  Interrupts also need on the chip itself

- **Time for scheduling**

  Some kind of timer to swap tasks.

- **Access to I/O**

  Graphics Controllers , Network Interfaces, Mouse , Keyboards etc.

The processor sits at the center of the platform (SOC) and interacts with all the other devices on the SOC.

The system memory map presents the locations of these devices (to the processor), but how does the processor know where everything sits ?

Some are built in and some are not built in addresses , so we introduce how addresses are known when the system is booted.

The System Memory Map (where is everything)

This is the key to understanding the system on a chip, and this map is generated from the processor's perspective (typically)

- A list of all physical addresses of all the devices on the SOC.
- DRAM (Dynamic RAM)
- Interrupt Controllers
- I/O Devices

IA-32 (Intel Architecture 32 bits) Platform

Two distinct address spaces

1. **Memory space (covers DRAM space and most I/O devices)**

   Ranges from 0 to 4GB (Space) , No all address spaces map to a memory or device.

   Space itself is 0 - 4GB, created and dynamically ised , not every address is mapped to something real, and as a result some addresses in the range are used or mapped to something real.

   This part of the space is accesses by read / write instruction (move / stop)

2. **Legacy IO Space (typically 64 kb)**

   Only accessed by in/Out instructions.

   Slower that memory space actions

   Intel set aside 64kb to support legacy IO devices from old chips.

ARM / Power PC only use Memory Space

They do not have the legacy issues that Intel have to deal with.

When a processor executes a read/ write instruction , the address that is generated by thar R/W is decoded by the **system memory address decoders** and is eventually rooted to the appropriate device to complete the operation.

This decision logic might match the address to

1. System DRAM controller which acts as the appropriate memory device
2. Resolve or match it to a hardware register.
3. an ethernet network controller saying we want to read or write to the network, typically on a PCI indicating a packet is ready from transmission

The address in the memory space is split here again into two separate spaces.

1. THe DRAM - Main memory address range
2. IO - Memory mapped IO (MMIO)

The top of the local memory (TOLM) (Upper Range) in the SOC indicates the top of the DRAM which extends from 1 MB range - TLOM

First megabyte in the memory is not used for the RAM

In the IA-32, 0 - 1MB is built from a mix of system memory and MMIO , (legacy and returned for compatability)

Memory range split in two , 64kb is part of this 1M*B*

The MMIO range is also divided up , in which IO devices can reside , is divided into subregions

- Fixed memory mapped addresses

  Hard coded addresses ranges
  Address of a flash device, (BIOS firmware , timer registers, Interrupts controllers) and do not change

- PCI Bus

  Range of MMIO address directed to the PCI / PCIe bus.

Devices appearing on the PCI bus have a configurable address decoder known as a **base address register (BAR)** . These are provisioned as part of the address space occupied by the devices on the PCI Bus.

Plug a network card into the PCI bus, the PCI bus talks to memory base address register , BIOS on boot sees that a thing has been plugged into a slot and assigns a range of address.