Memory Management

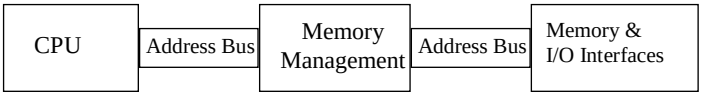
- ***Peter Rounce - room G06***
  p.rounce@cs.ucl.ac.uk

Memory Protection

Restrict addresses that an application can access:-

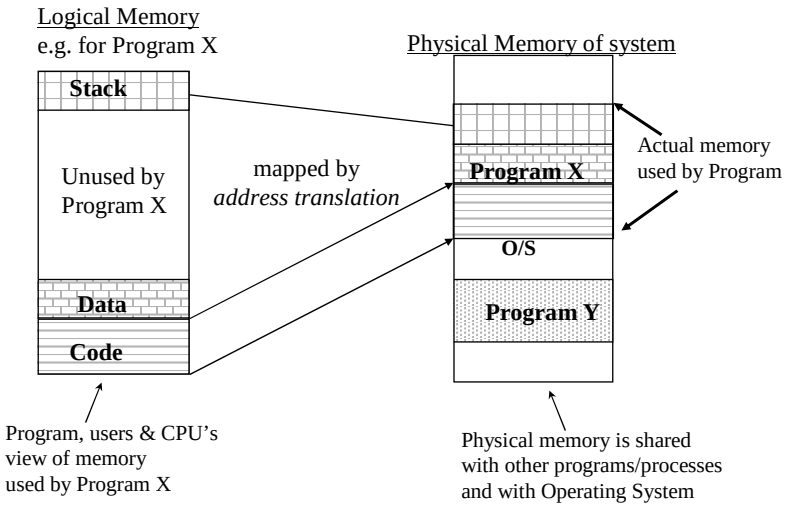
- **Limits application to its own code and data**
- **Prevents access to other programs and OS**
- **Prevents access to I/O addresses**

This address restriction is done by Memory Management hardware:-



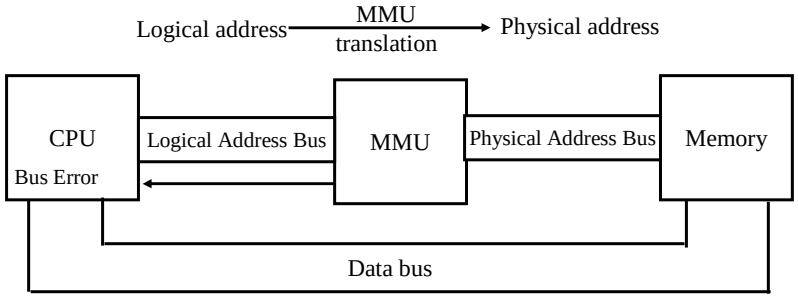
This Memory Management hardware does address translation as well.

Memory Protection + Address Translation:
Creation of illusion that program has all the memory to itself



Solution: have logical addresses used by program and CPU ; have physical addresses to access memory
Use Address translation to turn logical address into physical address.

Address Translation: basis of Memory Protection and Memory Management



Logical address	Address programmed into code and used by CPU CPU outputs Logical address on to CPU address Bus CPU only knows about Logical addresses
Logical address	where CPU considers data/instruction to be stored
Physical address	address where data/instruction is really stored

Note:Physical addresses are never stored in the program, only appear on address bus not data bus.

MMU = Memory Management Unit

Pentium MMU is on same chip as CPU

Base-Limit Registers - *Memory Protection + Address Translation*

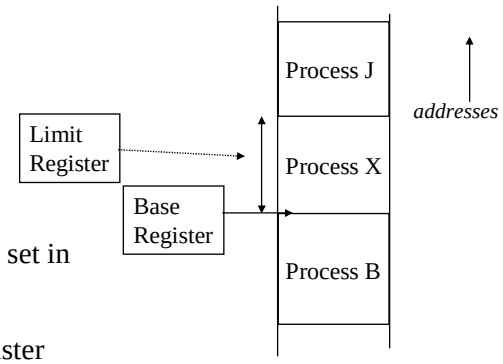
Several processes J,B,X…. in main memory

All processes have logical addresses starting at 0.

Process X is to run

Lowest physical address of X set in Base Register

Length of X set in Limit Register



Memory Protection: exception to CPU if address output by CPU >= limit register contents

Translation: Logical Address + Base Register ⟶ Physical Address

All programs compiled to start at address 0, but program loaded anywhere in memory.

Before running a process, the Operating system sets the Base register with lowest address used by process, and Limit Register with length of process. Addresses used by process must be in range 0 to (length - 1)

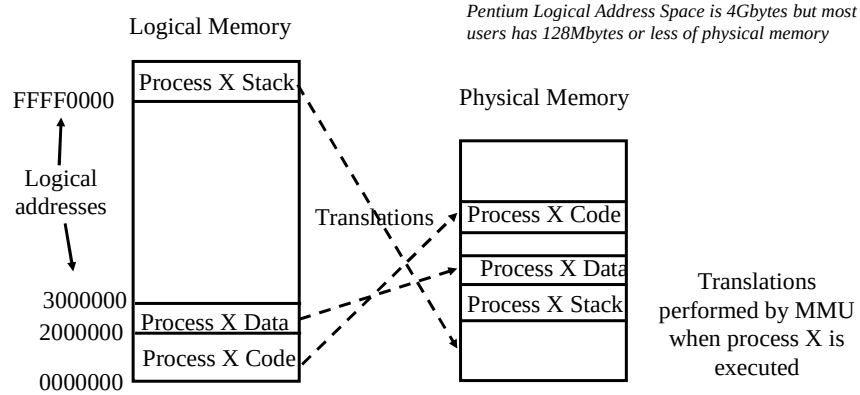
On each read and write cycle address output is checked against Limit register, and the process stopped through by an *exception* if the address is greater.

Otherwise the address form the CPU (the Logical Address) is *translated* to produce the Physical Address of the memory location to be accesses by adding the Base register contents:-.

Logical Address + Base Register --> Physical Address

More Complex
Address Translation Example

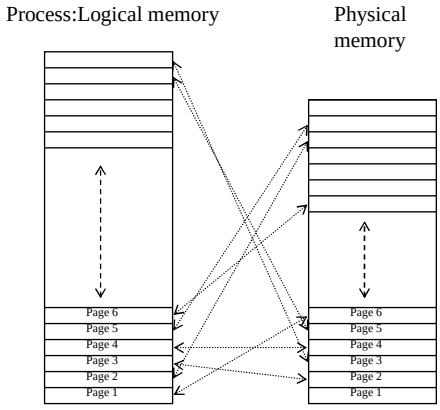
Address range of Logical Memory
is full address range of CPU.
Physical memory is usually smaller.
Pentium Logical Address Space is 4Gbytes but most users has 128Mbytes or less of physical memory



Block	Logical address range	Physical Address Range	Translation
Code	00000000-1FFFFFFF	70000000-9FFFFFFF	add 70000000
Data	20000000-2FFFFFFF	40000000-5FFFFFFF	add 20000000
Stack	FFFF0000-FFFFFFFF	3FFF0000-3FFFFFFF	sub C0000000

Translation information created when program is loaded by O/S from disk into memory

Page-Based Memory Management - The standard memory management system



Program divided into equal size units: ***pages***.

Page size is always small and a power of 2: 256, 512,….

Each page has separate translation to Physical memory.

Translation information for program kept in a "page table."

There is a page table for each process.

Page table can be very large

Translations only exist for Logical pages that are valid for process: few, if any programs use all of Logical Address Space.

Attempt to access Logical address with no translation to physical memory generates an exception to the CPU and the process is Terminated.

Page-based translation: Page Table

in general more complicated than previous slide

Translation information for program kept in page table

There is a page table for each process.

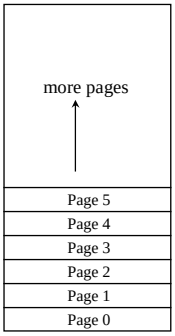
Part of Page Table for **a process** with 1000_{16} byte pages:

Logical page No	Logical Addresses	Physical Addresses
0	0000 - 0FFF	1A000 - 1AFFF
1	1000 - 1FFF	3000 - 3FFF
2	2000 - 2FFF	9000 - 9FFF
3	3000 - 3FFF	123000 - 123FFF
4	4000 - 4FFF	E8000 - E8FFF
5	5000 - 5FFF	4B000 - 4BFFF

From logical address, MMU gets page number and looks up table.

Physical addresses are allocated by O/S as program loaded into memory: their values depend on what space is free at that time. O/S builds table.

Logical memory Space
of Program
conceptually divided into pages



Pentium:
Page size - 1000_{16}
Max No. of logical pages for one program is 1000000_{16}

Advantages of Paging

- Can load program into available space in memory without need to coalesce free space into single block of free space.

- Can load several programs into memory at same time and run them even if programs' memory addresses are the same
  [these are logical addresses which be translated to different physical addresses.
  [each program has its own separate page table]

- compiler does not need to know memory addresses in which program will run:
  compiler always use same memory addresses whatever the program
  lets operating system create translation map on loading program.

- parts of logical address space unused by program have no mapping in table

- MMU generates an exception signal to CPU if access made to unused memory and program aborted (bus error) *[part of protection system]*

- page table kept in memory - read from memory by MMU, when process is scheduled to run

What's the use of address translation?

Better use of Memory
Can load programme into available space

.

Simpler compilation
compiler can compile all programs in same way with same logical addresses
[O/S sets up different translation for each programme loaded]

Protection of rest of memory
translations only for valid program addresses
generate exception to terminate programme if
invalid logical address sent to MMU

Prevents user program from accessing I/O interfaces and O/S code& data
[no valid translation provided to I/O addresses or O/S addresses]

Program is prevented from making changes to the page table:
no mapping to memory holding page table

O/S can access all addresses but User processes
can only access a restricted set of adddresses

Virtual Memory an extension to paging

not all of program has to be in memory for program to run
keep some pages of program on disk and load as required to memory.

Part of the page table for a process

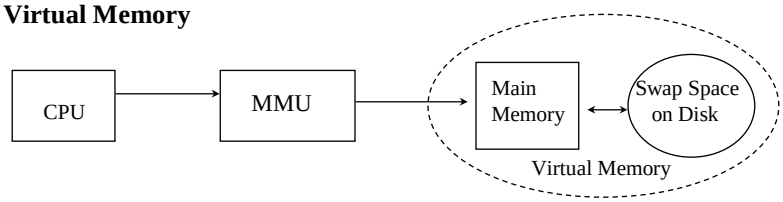
Logical page No	Logical Addresses	Physical Addresses
.		
.		
96_{16}	96000 - 96FFF	19000 -19FFF
97_{16}	97000 - 97FFF	swapped to disk
98_{16}	98000 - 98FFF	not used by program
.		

accesses to page 96_{16} are translated by MMU and memory accessed

an access to page 98_{16} MMU generates an exception to CPU
O/S analyses exception and aborts program.

an access to page 97_{16} MMU generates an exception to CPU
O/S analyses exception and loads page from disk into memory and process re-started.

Virtual Memory



Swap space is an area of a hard disk set aside
for holding pages of memory swapped out to provide space
for loading pages of another program

Virtual memory allows for better use of the available memory:

Large programs can be run on small memories: possibly more slowly.

More programs can have active pages loaded into memory.

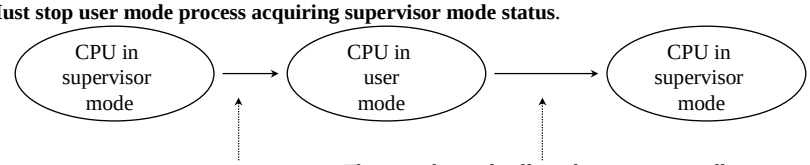
Identification of O/S accesses from User Accesses

A CPU running address translation needs 2 or more modes of operation:-
e.g. a kernel or **supervisor mode** for the operating system
a **user mode** for application processes.

[Pentium has 4 modes or levels - with the Pentium everything is more complex]

Signal output by CPU indicates processor mode:
signal is used by external hardware to enable/disable access to parts of the system.
[MMU uses signal to distinguish between operating system and user process accesses.]

Must stop user mode process acquiring supervisor mode status.



This mode switch is made easy by the CPU.

This switch is only allowed via a system call instruction, which forces a switch to small number of predefined addresses in the O/S.
Process cannot switch the state and keep running.
The O/S is always entered on this switch.

Protection System