Memory Protection:

Paging

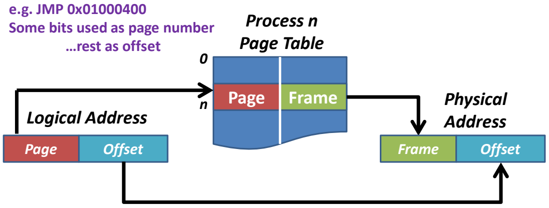
Dr Andrew Scott

a.scott@lancaster.ac.uk

1

Fixed Sized Memory Allocation: *Paging*

Logical address in paging looks like traditional linear address e.g. JMP 0x01000400
Some bits used as page number
...rest as offset

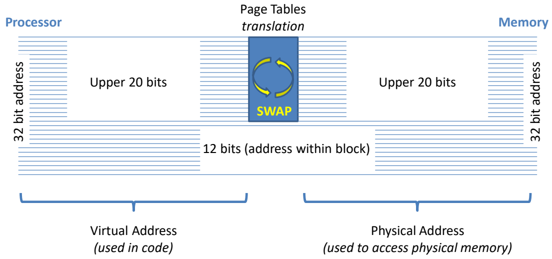


Note: We can allow logical address range to be larger than physical address range, as long as we never use more blocks than we have in real memory

So assume all entries in table will be valid as a real system would need to handle 'missing'/ non-mapped pages

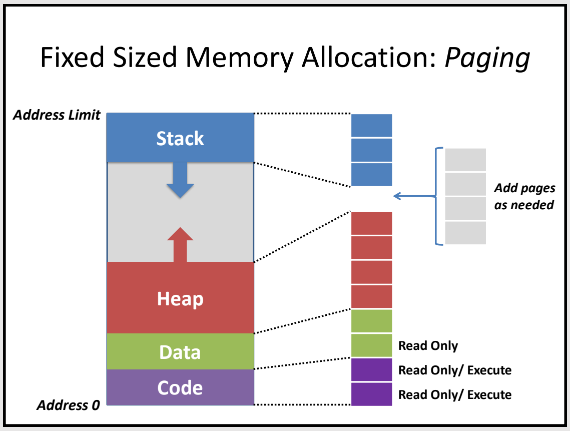
2

Address Bus: *Processor to Memory*



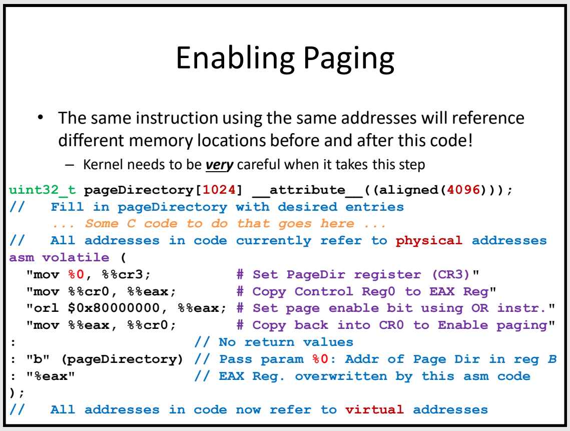
Varies with address bus width... e.g. 32 vs. 64 bit

3



Fixed Sized Memory Allocation: *Paging*

4

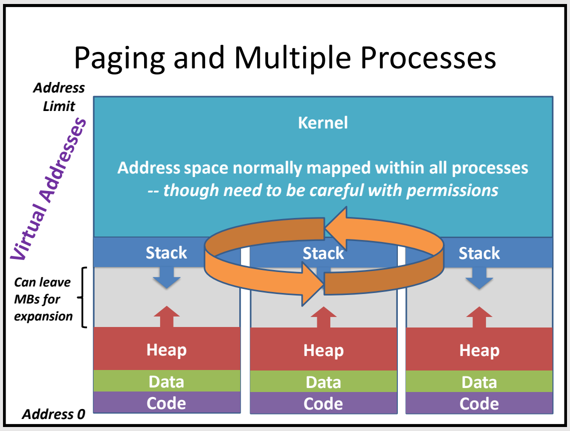


Enabling Paging

- The same instruction using the same addresses will reference different memory locations before and after this code!
  - Kernel needs to be ***very*** careful when it takes this step

```
uint32_t pageDirectory[1024] __attribute__((aligned(4096)));
//   Fill in pageDirectory with desired entries
     ... Some C code to do that goes here ...
//   All addresses in code currently refer to physical addresses
asm volatile (
  "mov %0, %%cr3;           # Set PageDir register (CR3)"
  "mov %%cr0, %%eax;        # Copy Control Reg0 to EAX Reg"
  "orl $0x80000000, %%eax; # Set page enable bit using OR instr."
  "mov %%eax, %%cr0;        # Copy back into CR0 to Enable paging"
:                      // No return values
: "b" (pageDirectory) // Pass param %0: Addr of Page Dir in reg B
: "%eax"               // EAX Reg. overwritten by this asm code
);
//   All addresses in code now refer to virtual addresses
```

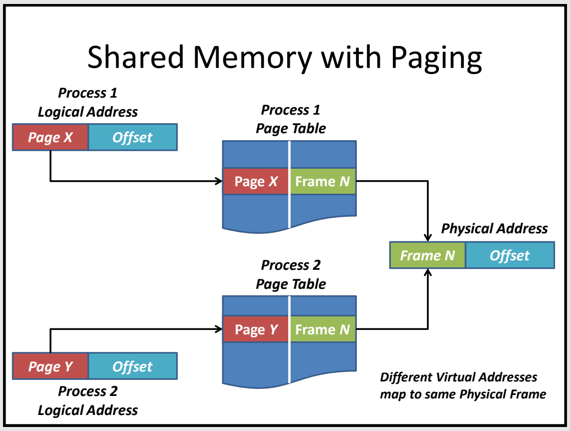
5



Paging and Multiple Processes

Address Limit
Virtual Addresses
Kernel
Address space normally mapped within all processes
-- though need to be careful with permissions
Stack
Stack
Stack
Can leave MBs for expansion
Heap
Heap
Heap
Data
Data
Data
Code
Code
Code
Address 0

6



Shared Memory with Paging

Different Virtual Addresses map to same Physical Frame

7