University of BRISTOL

Computer Systems B

COMS20012

Introduction to Operating Systems and Security

bristol.ac.uk

University of BRISTOL

OS161

bristol.ac.uk

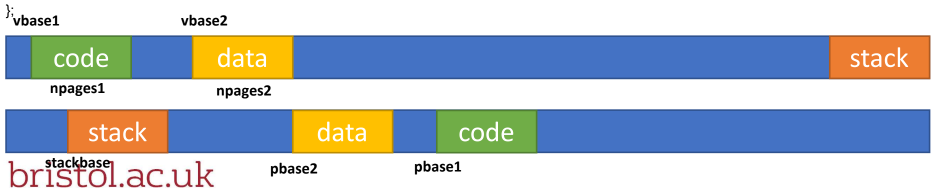
MIPS

- MIPS uses 32bits paged virtual and physical address
- MIPS has software managed TLB
  - Software TLB raises exception on every miss
  - Kernel is free to record virtual to physical mapping however it wants
  - TLB functions are handled by a function called vm_fault
    - kern/arch/mips/vm/dumbvm.c line 146
- vm_fault uses information from addrspace structure to determine virtual to physical mapping to load into the TLB
  - Each process has its own *addrspace* structure
  - Each *addrspace* structure describe where the pages are stored in physical memory
  - *addrspace* does the same job as a page table, but in a much simpler way. OS161 create contiguous segment.

bristol.ac.uk

OS161 address space

```
kern/include/addrspace.h
struct addrspace {
        vaddr_t as_vbase1; /* base virtual address of code segment */
        paddr_t as_pbase1; /* base physical address of code segment */
        size_t as_npages1; /* size (in pages) of code segment */
        vaddr_t as_vbase2; /* base virtual address of data segment */
        paddr_t as_pbase2; /* base physical address of data segment */
        size_t as_npages2; /* size (in pages) of data segment */
        paddr_t as_stackpbase; /* base physical address of stack */
};
```



bristol.ac.uk

dumbvm Address Translation

```
vbase1 = as->as_vbase1;
vtop1 = vbase1 + as->as_npages1 * PAGE_SIZE;
vbase2 = as->as_vbase2;
vtop2 = vbase2 + as->as_npages2 * PAGE_SIZE;
stackbase = USERSTACK - DUMBVM_STACKPAGES * PAGE_SIZE;
stacktop = USERSTACK;

if (faultaddress >= vbase1 && faultaddress < vtop1) {
                paddr = (faultaddress - vbase1) + as->as_pbase1;
} else if (faultaddress >= vbase2 && faultaddress < vtop2) {
                paddr = (faultaddress - vbase2) + as->as_pbase2;
} else if (faultaddress >= stackbase && faultaddress < stacktop) {
                paddr = (faultaddress - stackbase) + as->as_stackpbase;
} else {
                return EFAULT;
}
```

- USERSTACK = 0x8000 0000
- DUMBVM STACKPAGES = 12
- PAGE SIZE = 4KB

kern/arch/mips/vm/dumbvm.c
Line 202

- Line 222 – 239 update TLB

bristol.ac.uk

Initializing address space

- When the kernel creates a process it:
  - Creates an address space
  - Load the program data and code
- OS161 pre-load the programs in RAM
  - Most OS will load on demand
- A program code and data is described in an **executable**
- OS161 uses ELF (executable link format) as other OS (e.g., LINUX)
- OS161 execv system call reinitializes the address space of a process
  - int execv(const char *program, char **args)
- The program parameter should be the name of the ELF executable to be loaded

bristol.ac.uk

ELF files

- ELF files contain address space segment descriptions
  - ELF header describes the segment images
    - the virtual address of the start of the segment
    - the length of the segment in the virtual address space t
    - he location of the segment in the ELF
    - the length of the segment in the ELF
- the ELF file identifies the (virtual) address of the program’s first instruction (the entry point)
- the ELF file also contains lots of other information (e.g., section descriptors, symbol tables) that is useful to compilers, linkers, debuggers, loaders and other tools used to build programs

bristol.ac.uk

OS161

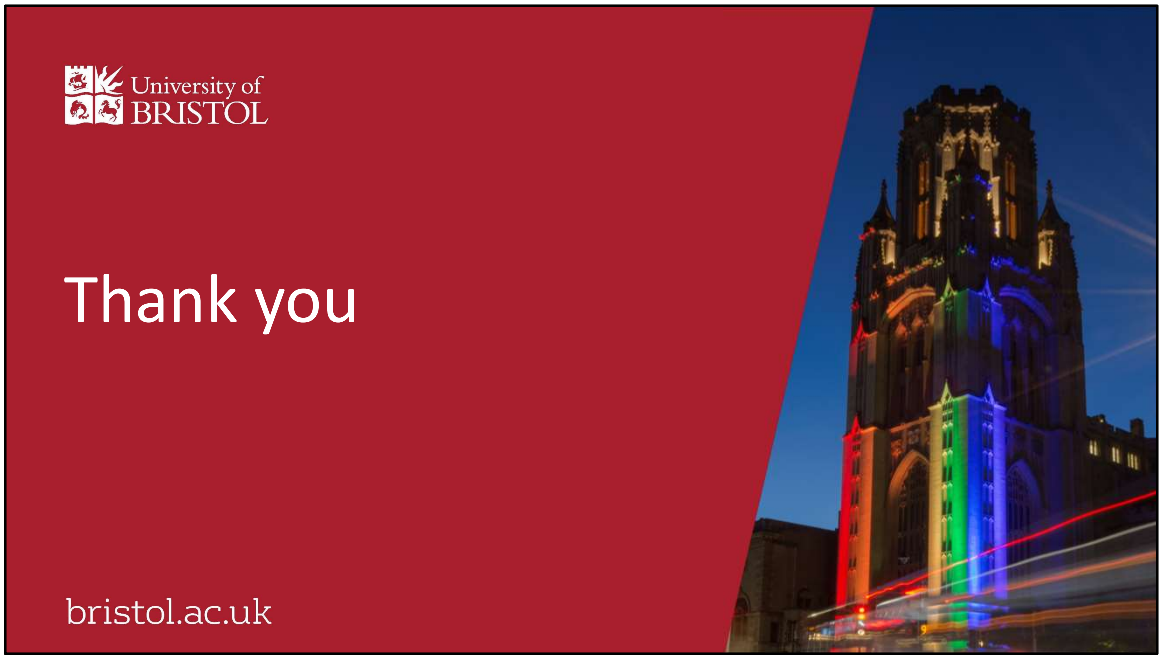
- OS161's dumbvm implementation assumes that an ELF file contains **two segments**
  - a **text segment**, containing the program code any read-only data
  - a **data segment**, containing any other global program data
- the images in the ELF file are an **exact copy** of the binary data to be stored in the address
- dumbvm creates a **stack segment** for each process
  - 12 pages long
  - ending at virtual address 0x7FFFFFFF

bristol.ac.uk

OS161

- If the image is smaller than the segment it is in loaded into, it should be zero filled
- Look through and understand: *kern/syscall/loadelf.c*

bristol.ac.uk



University of BRISTOL

Thank you

bristol.ac.uk