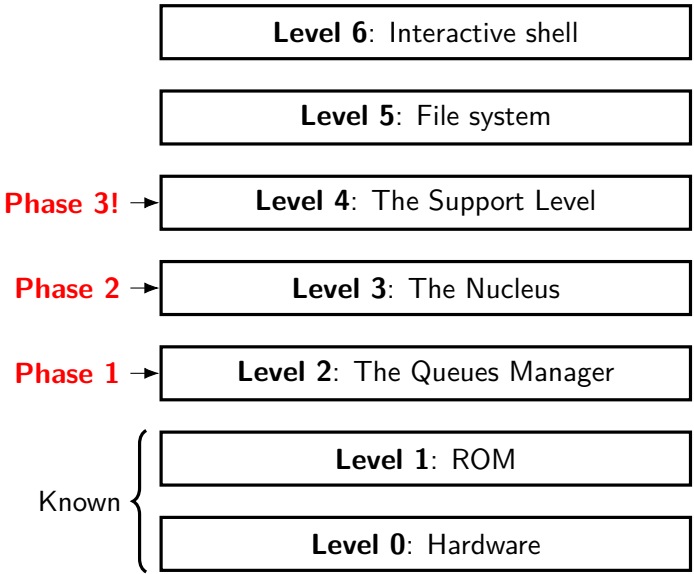
MultiPandOS: Phase 3

Luca Bassi (luca.bassi14@studio.unibo.it)
Luca Orlandello (luca.orlandello@studio.unibo.it)

April 11, 2025

MultiPandOS: six levels of abstraction



MultiPandOS: Phase 3 - Level 4: The Support Level

Level 4, the Support Level, builds on the Nucleus in two key ways to create an environment for the execution of user-processes:

1. Support for address translation/virtual memory.
2. Support for character-oriented I/O devices: terminals and printers.

```
typedef struct support_t {
    int         sup_asid;
    state_t     sup_exceptState[2];
    context_t   sup_exceptContext[2];
    pteEntry_t sup_privatePgTbl[USERPGTBLSIZE];
    unsigned int sup_stackTLB[500];
    unsigned int sup_stackGen[500];
    struct list_head s_list;
} support_t;
```

Goal of this phase

- You will be provided with 8 programs. This programs can be compiled and the 8 executables will be seen as as flash devices.
- You will write the `test` function that should upload the executables and the 8 programs should successfully execute.
- In addition to the scheduling you already have implemented, virtual memory and system calls should be implemented to have an easier interaction with devices.

Important: This slides should not be used as a substitution to the specs. The following is a panoramic view of the specs.

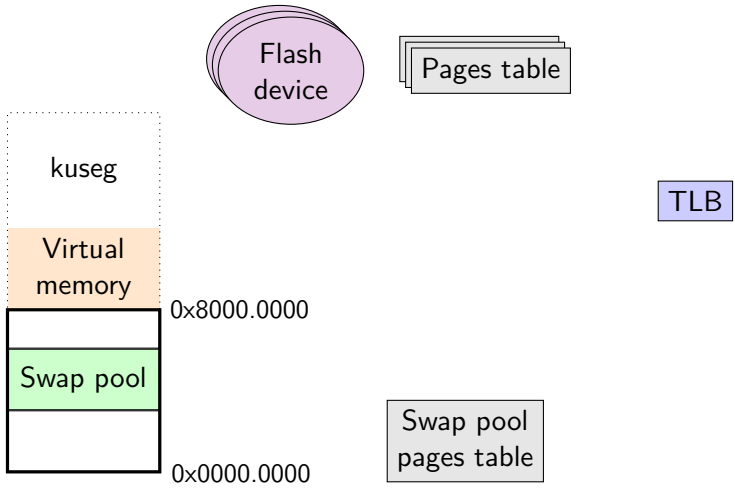
MultiPandOS virtual memory

The U-procs are executed in virtual memory (kuseg) that starts at 0x8000.0000.

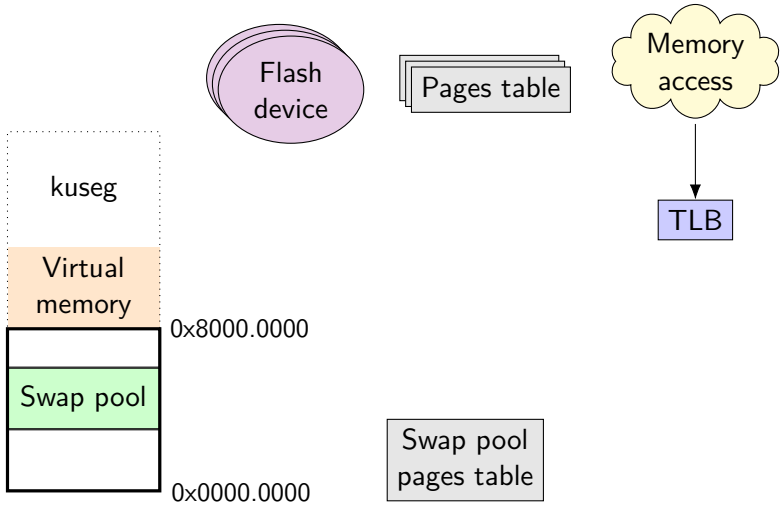
MultiPandOS uses a Translation Lookaside Buffer (TLB) to search for the corresponding physical address.

	EntryHI		EntryLo				
	VPN	ASID	PFN	N	D	V	G
0	0x80000	*i*			1	0	
1	0x80001	*i*			1	0	
⋮	⋮	⋮			⋮	⋮	
30	0x8001E	*i*			1	0	
31	0xBFFFF	*i*			1	0	

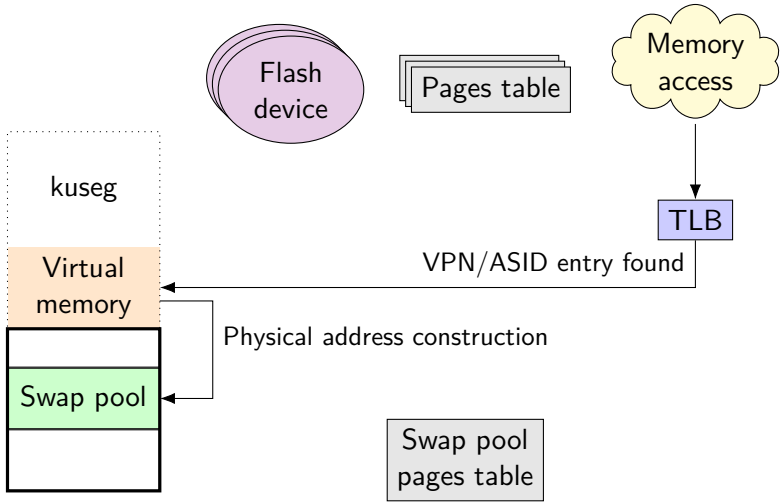
MultiPandOS virtual memory



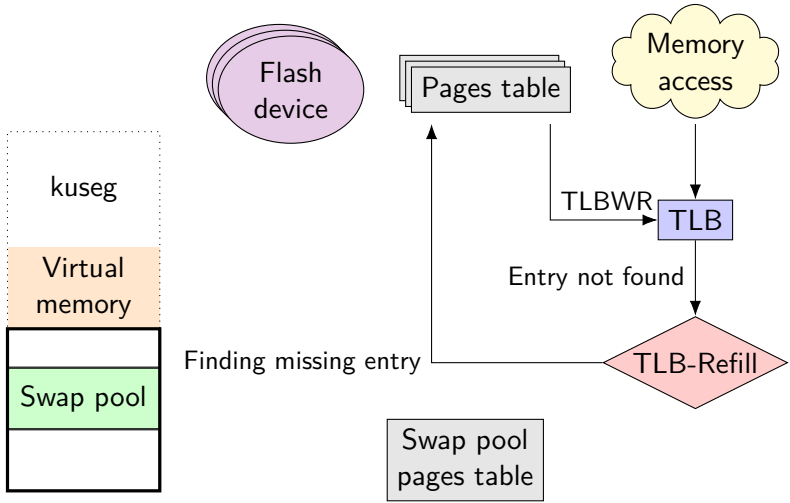
MultiPandOS virtual memory



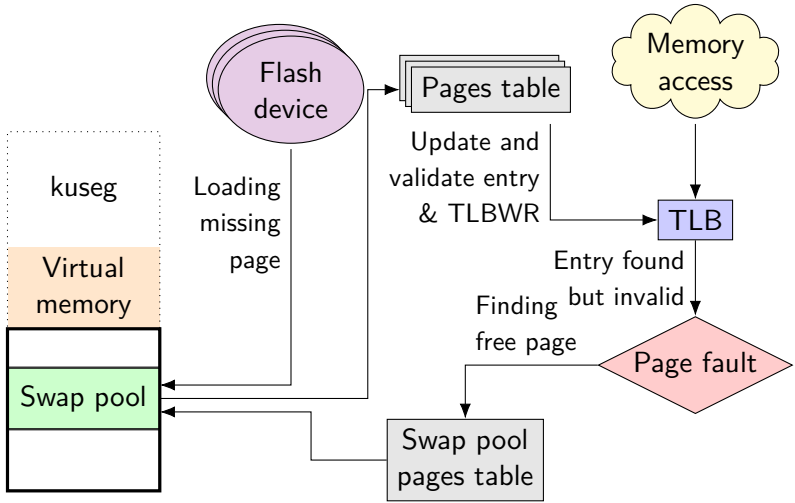
MultiPandOS virtual memory



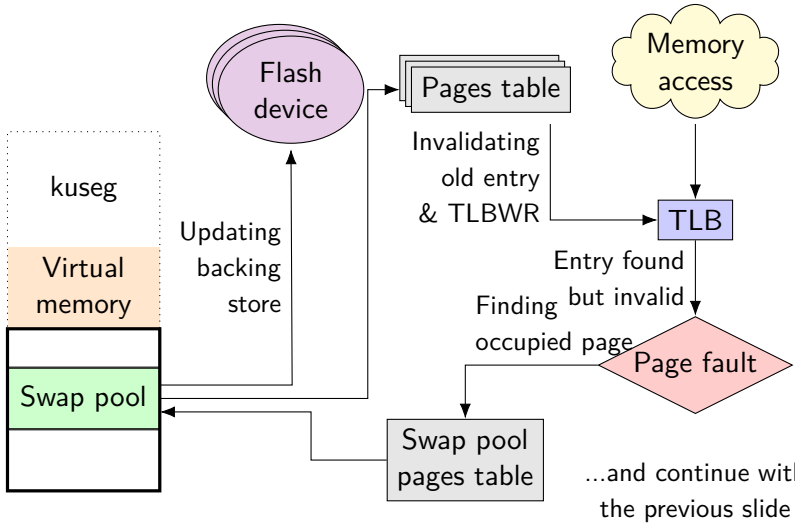
MultiPandOS virtual memory



MultiPandOS virtual memory



MultiPandOS virtual memory



Process initialization

The test process must initialize:

- Phase 3/Level 4's data structures (Swap Pool table, Swap Pool semaphore and device semaphores)
- Initialize and launch between 1 and 8 U-procs

After it must wait the termination of the 8 U-procs.

There's no need to modify Phase 2/Level 3, because Phase 3/Level 4 builds on top of it. Obviously, you can modify the code of the previous phases to adapt your needs.

The Support Level General Exception Handler

The Support Level general exception handler will process all passed up non-TLB exceptions:

- All SYSCALL exceptions numbered 1 and above (positive number)
- All Program Trap exceptions namely all exception causes except of those for SYSCALL exceptions and those related to TLB exceptions

The SYSCALL Exception Handler

The nucleus directly handles all NSYS SYSCALL exceptions (those having negative identifiers).

For all other SYSCALL exceptions the nucleus either treats the exception as a NSYS2 (terminate) or "passes up" the handling of the exception if the offending process was provided a non-NULL value for its Support Structure pointer when it was created.

The SYSCALL Exception Handler

Terminate This services causes the executing U-proc to cease to exist. The SYS2 service is essentially a user-mode "wrapper" for the kernel-mode restricted NSYS2 service.

WritePrinter This service causes the requesting U-proc to be suspended until a line of output (string of characters) has been transmitted to the printer device associated with the U-proc.

WriteTerminal This service causes the requesting U-proc to be suspended until a line of output (string of characters) has been transmitted to the terminal device associated with the U-proc.

ReadTerminal This service causes the requesting U-proc to be suspended until a line of input (string of characters) has been transmitted from the terminal device associated with the U-proc.

The Program Trap Exception Handler

The Support Level's Program Trap exception handler is to terminate the process performing the same operations as a SYS2 request.

Testing

There is a provided set of possible U-proc programs that will "exercise" your code.

These programs will generate page faults in addition to issuing SYSCALLs and purposefully causing Program Traps.

The supplied U-proc programs also come with their own `Makefile` configured to compile, link (using the U-proc linker script, `crtsi.o`), create a corresponding flash device (a `.uriscv` file), and preload the U-proc's load image on to a flash device.

Submission

The deadline is set for **Sunday June 8, 2025 at 23:59** or **Sunday July 6, 2025 at 23:59** or **Sunday August 25, 2025 at 23:59**.

Upload a single `phase3.tar.gz` in the folder associated to your group with:

- All the source code with a `CMakeLists.txt`
- Documentation
- README and AUTHOR files

Please comment your code!

We will send you an email with the hash of the archive, you should check that everything is correct.

You will receive another email with the score out of 10.