

1 Precheck

This section is designed as a conceptual check for you to determine if you conceptually understand and have any misconceptions about this topic. Please answer true/false to the following questions, and include an explanation:

- 1.1 Responsibilities of the OS include loading programs, handling services, combining programs together for efficiency.

- 1.2 The purpose of supervisor mode is to isolate certain instructions and routines from user programs.

- 1.3 An operating system uses context switches to allow for multiple processes to run simultaneously across multiple CPUs.

- 1.4 Having virtual memory helps protect a system.

- 1.5 The virtual address space is limited by the amount of memory in the system.

- 1.6 The virtual and physical page number must be the same size.

- 1.7 If a page table entry can not be found in the TLB, then a page fault has occurred.

1.8 For I/O, polling is better than interrupts when I/O events occur regularly at a fast rate.

1.9 Memory-mapped IO only works with polling.

2 Addressing

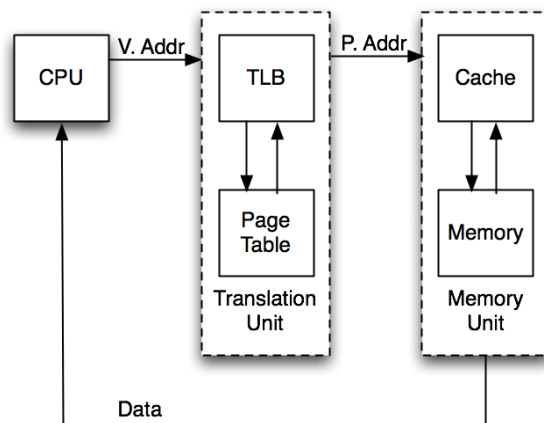
Virtual Address (VA) What your program uses

Virtual Page Number (VPN)	Page Offset
---------------------------	-------------

Physical Address (PA) What actually determines where in memory to go

Physical Page Number (PPN)	Page Offset
----------------------------	-------------

For example, with 4 KiB pages and byte addresses, there are 12 page offset bits since $4 \text{ KiB} = 2^{12} \text{ B} = 4096 \text{ B}$.



Pages

A chunk of memory or disk with a set size. Addresses in the same virtual page map to addresses in the same physical page. The page table determines the mapping.

Valid	Dirty	Permission Bits	PPN
— Page entry (VPN: 0) —			
— Page entry (VPN: 1) —			

Each stored row of the page table is called a **page table entry**. There are 2^{VPN} bits such entries in a page table. Say you have a VPN of 5 and you want to use the page table to find what physical page it maps to; you'll check the 5th (0-indexed) page table entry. If the valid bit is 1, then that means that the entry is valid (in other words, the physical page corresponding to that virtual page is in main memory as opposed to being only on disk) and therefore you can get the PPN from the entry and access that physical page in main memory.

The page table is stored in memory: the OS sets a register (the Page Table Base Register) telling the hardware the address of the first entry of the page table. If you write to a page in memory, the processor updates the “dirty” bit in the page table entry corresponding to that page, which lets the OS know that updating that page on disk is necessary (remember: main memory contains a subset of what’s on disk). This is a similar concept as having a dirty bit for each cache block in a write-back cache. Each process gets its own illusion of full memory to work with, and therefore its own page table.

Protection Fault The page table entry for a virtual page has permission bits that prohibit the requested operation. This is how a segmentation fault occurs.

Page Fault The page table entry for a virtual page has its valid bit set to false. This means that the entry is not in memory. For simplicity, we will assume the address causing the page fault is a valid request, and maps to a page that was swapped from memory to disk. Since the requested address is valid, the operating system checks if the page exists on disk. If so, we transfer the page to memory (evicting another page if necessary), and add the mapping to the page table **and** the translation lookaside buffer (TLB).

Translation Lookaside Buffer

A cache for the page table. Each block is a single page table entry. If an entry is not in the TLB or the valid bit = 0, it’s a TLB miss. Typically fully associative:

TLB Valid	Tag (VPN)	Page Table Entry		
		Page Dirty	Permission Bits	PPN
		— <i>TLB entry</i> —		
		— <i>TLB entry</i> —		

2.1 What are three specific benefits of using virtual memory?

2.2 What should happen to the TLB when a new value is loaded into the page table address register (i.e. we are switching page tables to those for another process)?

3 VM Access Patterns

3.1 A processor has 16-bit addresses, 256 byte pages, and an 8-entry fully associative TLB with LRU replacement (the LRU field is 3 bits and encodes the order in which

pages were accessed, 0 being the most recent). Suppose RAM (main memory) has 12 bit addresses. At some time instant, the TLB for the current process is the initial state given in the table below, and we have one free physical page = 0x7. Suppose the least recently used physical page is 0x9. Assume that all current page table entries are in the initial TLB. Assume also that all pages can be read from and written to. Fill in the final state of the TLB according to the following access pattern, and also write out the physical addresses corresponding to each location accessed. Indicate TLB hit or miss (if it is a TLB miss, indicate whether there is a page fault). Update the page table as needed.

Access Pattern

- | | |
|----------------------------|----------------------------|
| 1. 0x11f0 (Read) | 4. 0x2332 (Read) |
| 2. 0x1301 (Write) | 5. 0x20ff (Read) |
| 3. 0x20ae (Write) | 6. 0x3415 (Write) |

Initial TLB

VPN	PPN	Valid	Dirty	LRU
0x01	0x1	1	1	0
0x00	0x0	0	0	7
0x10	0x3	1	1	1
0x20	0x2	1	0	5
0x22	0xb	0	0	7
0x11	0x4	1	0	4
0xac	0x5	1	1	2
0xff	0xf	1	0	3

Initial Page Table

This page table does is simplified and includes the status bits valid and dirty. LRU is omitted for simplicity.

VPN	PPN	Valid	Dirty
0x00	0x0	0	0
0x01	0x1	1	1
...
0x13	0xc	0	0
...
0x23	0x8	1	0
...
0x34	0xd	0	0
...
0xff	0xf	1	0

How many bits are the offset, VPN, and PPN?

4 Polling & Interrupts

4.1 Fill out this table that compares polling and interrupts.

Operation	Definition	Pro/Good for	Con
Polling			
Interrupts			

5 Memory Mapped I/O

5.1 For this question, the following addresses correspond to registers in some I/O devices and not regular user memory.

- `0xFFFF0000`—Receiver Control: LSB is the ready bit, there may be other bits set that we don't need right now.
- `0xFFFF0004`—Receiver Data: Received data stored at lowest byte.
- `0xFFFF0008`—Transmitter Control: LSB is the ready bit, there may be other bits set that we don't need right now.
- `0xFFFF000C`—Transmitter Data: Transmitted data stored at lowest byte.

Recall that receiver will only have data for us when the corresponding ready bit is 1, and that we can only write data to the transmitter when its ready bit is 1. Write RISC-V code that reads byte from the receiver and writes that byte to the transmitter (busy-waiting if necessary).