RISC-V Virtual Memory

1. Introduction

Virtual to physical translation is made by hardware. Each virtual address space has it's own page table. Page table can be thought as a function:

$$f(\text{vitual\_address}) \rightarrow \text{phisical\_address, R/W/X/V/U}$$

This function is used on all interactions with memory. To speed up the conversion a cache called **Translation Lookaside Buffer** can be used.

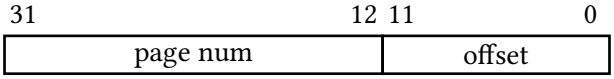
1.1. Permissions:

- **X**: can fetch instruction from page?
- **V**: is valid?
- **U**: is user page? (instead of supervisor)

Users can only access user mode pages. Kernel can only access supervisor mode pages. Kernel can access user mode pages when bit **SUM** of **sstatus** is set. This is necessary to pass arguments and return values from system calls.

1.2. Pages

Page size is 4kB (2^{12} B), pages must align on page boundaries. Page address:



Page table maps virtual page numbers to physical page numbers.

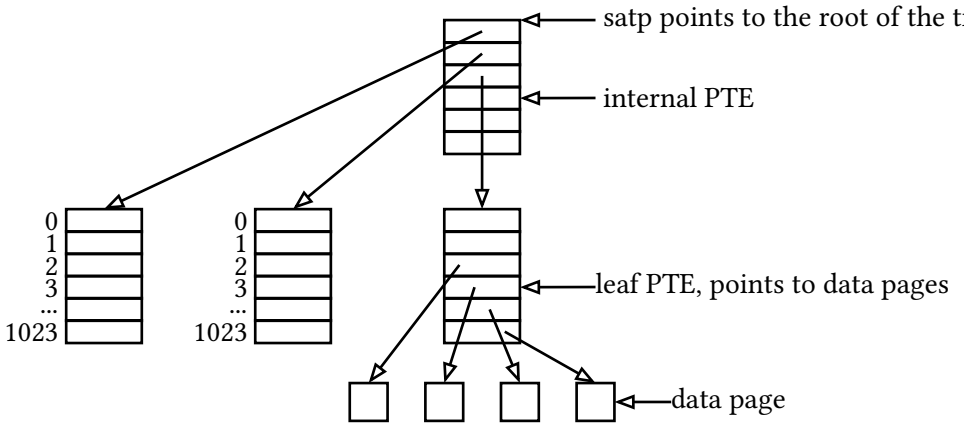
2. Sv32

On RV32: 32-bit virtual address ⟶ 34-bit physical address

Each virtual address space can be 4GB, but the whole system can have 16GB of addressable memory.

To do the mapping radix trees are used (a tree in which each node has n children, in Sv32 $n = 1024$).

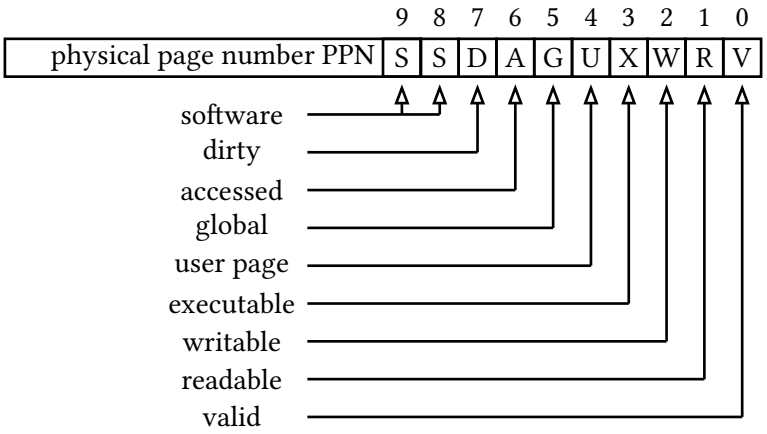
$$1024 = 2^{10}\text{, 2-level tree: } \left(2^{10}\right)^2 = 2^{20} \text{ leaves (number of virt pages)}$$



Each node is an array of 1024 page table entries PTEs. Each PTE points to a node at the next level. Each entry is 4B(32 bit) so the whole node is 4kB, same as a page.

2.1. Page table Entry PTE

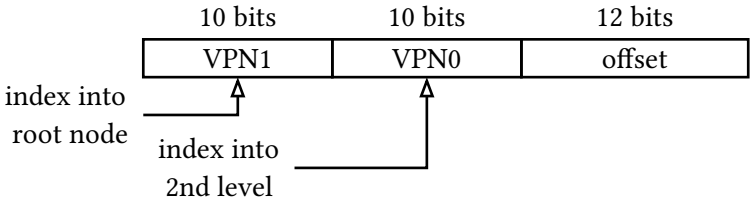
Each node contains a PTE (page table entry).



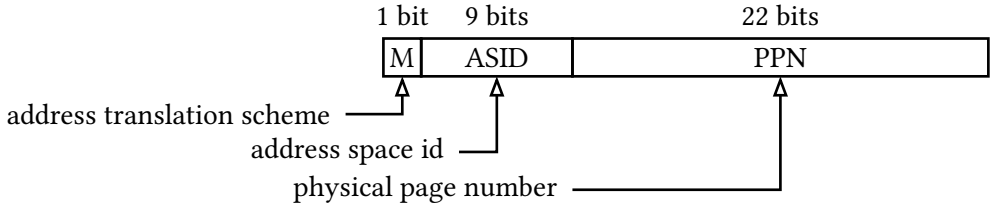
An interior page node entry (that doesn't point to actual data) has bits XWR set to 000.

2.2. Address translation

- Virtual address:



- satp register:



2.2.1. Algorithm

```
address = satp[PPN] << 12 // address of the root node; address is 34 bit var
// address of the root page from satp + offset from VPN1
index = virtual_address[VPN1]·4
pte = memory[address + index]

address = pte[PPN]
index = virtual_address[VPN0]·4
pte = memory[address + index]

// Atomically
set pte[accessed]
if operation is write:
  set pet[dirty]

physical_address = pte[PPN] || virtual_memory[offset]
```

2.3. Exceptions on page tree walk

Access Faults problems accessing physical memory
Page Faults problems accessing virtual memory

1. If the valid bit is not set ⟶ page fault
2. If W is set but R not ⟶ page fault
3. If proper bit for given instruction is not set ⟶ page fault

2.4. Megapages

- If root PTE has bits R or X set it points to a megapage
  - specification requires when W is set R is also set
  - it's sufficient to check bits R and X
- Megapage is 4MB in size, 1024 data pages combined.
- When translating a virtual address into mega page the 22 lsb are used as offset.
- Megapages reduce the pressure on TLB
  - fewer entries are required to cover large amount of memory
  - Kernel is mapped into all address spaces, mapping it with megapages is more efficient

3. Demand Paging