Paged Virtual Memory

- memory maps
- address translations, page tables, TLBs
- page faults
- paging and demand loading
- page replacement algorithms

memory map for process 1 in homework 1
What happens when you open two terminals are run two copies of homework. Why each of the copies run in the same address (virtual address space)

```
Virtual Address space      Physical address space
VA                        PA
```

VA is mapped to the physical address mapping is done through h/w support in the CPU using a page table.

Virtual Address

On a system with virtual memory, any address used by the CPU instruction

eg: when u call a function
Physical Address

Any address that is external to the CPU

e.g. a computer having 2ab memory.
by two slots of 1ab memory.

\[
\begin{array}{c}
2a \\
1a \\
\end{array}
\]

\[32\text{bits}\]
\[\text{mem}\]

\[\Rightarrow\] a particular physical address always
refers to the same memory location.

\[\Rightarrow\] Consider,
32-bit VA, PA
4K pages

VPN \hspace{1cm} \text{virtual} \hspace{1cm} \text{page #}

\[31\]
\[\ldots\]
\[12\]
\[0\]

20 bits \hspace{1cm} \map \hspace{1cm} \text{map} \hspace{1cm} \map \hspace{1cm} \text{physical page #}

\[\text{physical page #}\]
\[\text{PPN}\]

Mapping can not be done with software alone
-h/w support is needed.
MMU ← Memory Management Unit responsible for mapping

- mechanism that translates VPN to PPN
- OS supplies the policies for this translation

Map: Stored in memory accessed by mmu

most straightforward way to do this

- create a large table

\[
\begin{array}{c}
\text{VPN} \\
2^{20}
\end{array}
\]

\[
\begin{array}{c}
\text{phys page} \\
\require{cancel}
\text{1M entries} \\
\times 4\text{ bytes} = 4\text{MB}
\end{array}
\]

index the table by the VPN and directly read out the phys page #.

→ disadvantage: uses a lot of memory.
To overcome this disadvantage, we use multiple tables.

Diagram:

- 0
- 20
- 1023
- 1023
- Phys Pg
- 12

Tree:

- 1024ary tree
- Page directory
- PD[A31:22]
- PT[A21:12]
- Page table(s)
CR3 \rightarrow \text{stores the root address of the page table in Pentium}

\text{Example}

\begin{align*}
\text{Virtual} & \quad \text{Phys} \\
\begin{array}{c}
\text{P} = \text{PT}[0] \\
P = 00001000 \\
\text{VPN} \rightarrow \text{PPN}
\end{array} & \quad \\
\begin{array}{c}
\text{PC} = 09000030 \\
\text{fetch:} \\
\text{VPN} \rightarrow \text{PPN} \\
024 \ 000 \\
00000 + 4 \times 024 \\
\text{VPN} \rightarrow \text{PPN} \\
09000 \rightarrow 00002
\end{array}
\end{align*}
Representation of the steps involved in fetching an address by the MMU.

MMU

PD = array[1024] of ptr to PT
PT = array[1024] of ptr to P

address = a31-22 : 10
a21-12 : 10
offset : 12

fetch(addr):

PT = PDT[addr a31-22]
P = PT[addr a21-12]
val = P[offset]
This is inefficient, therefore OS maintains a TLB

**TLB - Translation Lookaside Buffer**

Caches the VPN $\rightarrow$ PPN mapping

If the mapping is present, directly access the address.

If not fetch using the above procedure, also, cache this mapping in the TLB

\[ V \rightarrow \text{TLB ?} \]

Page Fault:

- CPU
- \( \rightarrow \) VA
- \( \rightarrow \) PD [A31:22]
  - invalid

Interrupt: page fault

If you get a page fault on an instruction, CPU has to return to the OS to resolve the page fault and then return.

OS can either
1) kill the process : seg fault
or 2) resolve the page fault and return to the instr
OS can resolve by zero fill by paging from disk.

OS: page table

image of vm

paging from disk

/bin/ls

set up the mapping

page
Page Table entry (PTE)

20 bits: PPN

12 bits

P → Present bit

0: generates page fault

1: valid entry

R/W → Read/Write

0: Read only

U/S → User/Supervisor mode

A, D \{ Accessed and Dirty bits

-provided so that OS can track usage of memory

VA P

<table>
<thead>
<tr>
<th>file</th>
</tr>
</thead>
<tbody>
<tr>
<td>file</td>
</tr>
</tbody>
</table>

page

Page replacement: → similar to replacing entries in a cache or a TLB

given access pattern \( a_1, a_2, \ldots, a_N \)

which physical pages \( P_1, \ldots, P_m \) \( m \leq N \)

and physical pages \( P_1, \ldots, P_m \) \( m \leq N \)

which \( P_i \) does \( a_j \) go in?