First Half: Lecture 4

* Paged Virtual Memory:

- memory maps
- address translation, page tables, TLB's
- page faults
- paging & demand loading
- page replacement algorithms

* About Homework 1:

proc1

```
stack

0x9000000

code

FFFFFFFFFF
```

"q2 prog" was created to run 4K in proc1 address space & not in proc2 address space.

```
80k

stack

BE090000

12k

09000000

16k

data

0B0C4000

400k

code

08048000

00000000
```

Mapping between virtual & physical Address space.
If we try to execute something in virtual space which is not mapped to any physical addr then it won't work.

Virtual Address: (Internal to CPU)
- Used by CPU for program execn.
- e.g. PC value, SP value

Physical Address: (External to CPU)
- addr that go to motherboard & to memory thru pins. It never changes during any timespan whereas virtual addr. may refer somewhere by just changing its mapping

* 32 bit Virtual & Physical addr. Pentium
4K pages

![Diagram]

- Identify page in memory
- Physical page #
- Offset within the page
- MMU (Memory Management Unit)

responsible for translating virtual page # to physical page #.
Memory Map: stored in memory & accessed by MMU.

Disadvantage: Uses a lot of memory (e.g., MBs).

If we use 64-bit address this table will go in hundreds of GBs.

Solution: Use multilevel map.
Page Directory \( \rightarrow \) 1024-ary tree \( \rightarrow \) Page Tables

\[ PT \[ A_{21:12} \] \]

\[ PT = PD \left( A_{31:22} \right) \]

set \( PC = 09000030 \)

**Virtual**

2000
1000
0900 0000

**Physical**

fetch:
VPN : 09000 \( \rightarrow \) PPN

\[ 0000 \ 0000 \ 0000 \ 0000 \]

A31:22 \( \rightarrow \) A21:12

\[ PD[0243] \]

\[ 00000 + 4 \times 024 = 0000090 \]

\[ P = PT[0] \]

\[ 00001 \]

\[ = 00001000 \]

\[ \rightarrow 00002 \]

(Goes across memory bus)
MMU: 

\[ PD = \text{array [1024]} \text{ of } \text{ptr to } PT \]

\[ PT = \text{array [1024]} \text{ of } \text{ptr to } P \]

\[ \text{address = a 31-12 : 10} \]
\[ a 21-12 : 10 \]
\[ \text{offset : 12} \]

So to be precise following holds:

\[ \text{fetch (addr) :=} \]
\[ PT = PD\{\text{addr, a 31-22}\} \]
\[ P = PT\{\text{addr, a 21-12}\} \]
\[ \text{val} = P\{\text{offset}\} \]

So if \( PC = 00000030 \) flow over buses look like:

\[ \begin{align*}
0000 0000 &\rightarrow PD\{0,24\} \\
0000 0001 &\rightarrow PT\{0\} \\
0000 0002 &\rightarrow P\{0,30\}
\end{align*} \]

\[ \text{value} \]

But this is inefficient & so to this is TLBs meaning Translation Lookaside Buffer.

So to translate from virtual to physical addr. first look TLB, so TLBs hold cached values which in turn can save few exchanges in above scheme.
Y \rightarrow \boxed{TLB} \rightarrow Y N \rightarrow \text{continue the process normally.}

Skip first 2 exchanges

\text{value (if it is mov 00001014, eax) diff page reference}

Then process this whole flow again.

* TLBs are hardware implemented mapping tables so holds same space no matter how many entries are there at any point of time

* \text{Seg Faults:}

\text{VA} \rightarrow \text{PD[A31:22]}

\rightarrow \text{Invalid}

\rightarrow \text{Page fault \rightarrow handled by OS \rightarrow seg fault}

\text{Resolve the Page Fault}

\text{It can be resolved by zero fill process or paging from disk.}

* \text{OS / VM \rightarrow page table}

\text{In case of page faults, the VMA can be referred to resolve them as shown here}