Virtual Memory

- ISA has large VA space.
  - Allows user to uniquely identify lots
- Physical Memory smaller
  - Cost issue

Virtual Memory Management
- Access Control
- Translation

The VAX Model

![Diagram of Virtual and Physical Memory with Working Set and Balance Set]
FRAME

PAGES

VAX! 2^9 BYTOS

x86: 2^12 BYTOS

VA

PAGE NO.

PA

PFN

FQ. NO.

TRACK

Cylinder

ROTATION

SEEK

RESIDENT
The Abstraction

1. Is Page No. < P0LR? 
   NO: ACV Fault!

2. Get Correct PTE

Index by Page No.

P0BR

PTE for Page 1

3. V Prot PFN
   The PTE

Check Protection
NO: ACV Fault!

4. Is Page Resident (i.e. V = 1)
   NO: TNV Fault!

5. All Cool: PA
TLB Structure

Example: 16 Entry TLB

Page No. 10 Bits

\[ \text{\because 4 Bits Used To Hash Into TLB.} \]

TLB is a CAM
One Final Example

Let's modify the VAX ISA to make it easier to see what is going on. We will retain the essential elements, but we will reduce all the numbers.

For example: VA will go from 32 bits to 9 bits
Page size will go from 512 bytes to 16 bytes
PA will be 9 bits
PTE will still be 4 bytes.

Page Shows a snapshot of virtual/physical memory. Several things are worth noting:

1. Virtual Memory = 512 bytes: 32 pages possible
   P0 has a max of 8. In our example: 6 pages needed
   P1 " " " " " " " " In our example: 2 pages " "
   SS " " " " " " In our example: 5 pages " "

(PMBR)

2. P0 page table starts at VA = 120 (Note: 2, 0 are hex digits.
   Since there must be 6 entries, the P0 page table
   Consumes all of page 2, half of page 3 or SS.

(SBR)

3. System page table starts at PA = 50 (Note: 5 is octal, 0 is hex.
   Since there must 5 entries, system page table
   Consumes all of frame 5 and 1/4 of frame 6.

4. Note that sys. page table indicates 3 pages resident (Page 0, 34,
   Note that part of P0 page table indicates page 5 in, page 4 out)
VIRTUAL MEMORY

<table>
<thead>
<tr>
<th>PFN</th>
<th>Page 2, P1</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td></td>
</tr>
<tr>
<td>10</td>
<td></td>
</tr>
<tr>
<td>20</td>
<td></td>
</tr>
<tr>
<td>30</td>
<td></td>
</tr>
<tr>
<td>40</td>
<td></td>
</tr>
<tr>
<td>50</td>
<td></td>
</tr>
<tr>
<td>60</td>
<td></td>
</tr>
<tr>
<td>70</td>
<td></td>
</tr>
</tbody>
</table>

PHYSICAL MEMORY

<table>
<thead>
<tr>
<th>PFN</th>
<th>Page 3, SS</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td></td>
</tr>
<tr>
<td>10</td>
<td></td>
</tr>
<tr>
<td>20</td>
<td></td>
</tr>
<tr>
<td>30</td>
<td></td>
</tr>
<tr>
<td>40</td>
<td></td>
</tr>
<tr>
<td>50</td>
<td></td>
</tr>
<tr>
<td>60</td>
<td></td>
</tr>
</tbody>
</table>

FREE BLOCKS:

- PBUR

PAGE TABLE:

- PTE of Page 3 of System Space

- PTE of Page 5 of System Space

- PTE of Page 6 of System Space

- PTE of Page 7 of System Space

- PTE of Page 8 of System Space

- PTE of Page 9 of System Space

- PTE of Page 10 of System Space

- PTE of Page 11 of System Space

- PTE of Page 12 of System Space
Finally, a page table computation:

We wish to execute: LD R1, X

Where X has VA: 001011000

VA of X: 001011000

---

Page 5

---

Because PTE contains 4 bytes

---

PE3R → +1 to index into PES page table

---

10100000

---

VA of PTE of Page 5, PES space

Which is on Page 3 of virtual system space

---

Page 3

---

SBR → +1 to index into system page table

---

1010000

---

PA of PTE of Page 3, System space

---

SBR → [1]

---

← PTE of Page 3 of System Space. It occupies PFN 1

---

VA of PTE of Page 5 of PES space [1010111000]

Is mapped to PA [001011000]
So, if we look atPFN 1

\[
\begin{array}{c|c}
0 & 1 \\
1 & 2 \\
\end{array}
\]

We can read the PTE of Page 5, P₀ Space. We see Page 5, P₀ Space occupies PFN 2

The VA of X \(100'101'1000\) therefore maps to 

PA \(010'1000\)

We look in that location and find 17, which we load into R1.
IA-32

Activity: Intel calls it Linear Address
Segment Register

Seegment Registers

GDT

GDTR

LDT

LDTR

2^{13} \times 8 \text{ bytes each}

Segment Descriptor

<table>
<thead>
<tr>
<th>Field</th>
<th>Bits</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>BASE [31:20]</td>
<td>11</td>
<td>Base address (GDT)</td>
</tr>
<tr>
<td>LIM [15:4]</td>
<td>8</td>
<td>Limit</td>
</tr>
<tr>
<td>P</td>
<td>3</td>
<td>Access privilege</td>
</tr>
<tr>
<td>Type</td>
<td>3</td>
<td>Segment type</td>
</tr>
<tr>
<td>LIM [14:0]</td>
<td>10</td>
<td>Limit (LDT)</td>
</tr>
<tr>
<td>BASE [15:0]</td>
<td>10</td>
<td>Base address (LDT)</td>
</tr>
</tbody>
</table>

G: Granularity  1 byte /4K bytes
P: Segment Present
PL: Privilege Level
TYPE: Segment type
Segmentation AND Paging

Program

MOV A, M

Linear Address Space

BASE

LIMIT

PA

Physical Memory

A is logical Address

Linear Address

10

10

12

PAGE DIRECTORY

PAGE TABLE

FRAME OF PHYSICAL MEM.

Virtual translation enabled

PTE

PFN | PTE

DIRTY | PRIV | PRESENT

S | U | R | W

S | U | R | W

Linear Address

BYTE ON PAGE
= BYTE ON FRAME

31

21

11

00
Task State Segment (TSS)

<table>
<thead>
<tr>
<th>I/O Map</th>
<th>0</th>
<th>T</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>LDT</td>
<td></td>
</tr>
<tr>
<td>0</td>
<td>DS</td>
<td></td>
</tr>
<tr>
<td>:</td>
<td>:</td>
<td></td>
</tr>
<tr>
<td>EDI</td>
<td></td>
<td></td>
</tr>
<tr>
<td>EAX</td>
<td></td>
<td></td>
</tr>
<tr>
<td>CPL, AC1</td>
<td></td>
<td></td>
</tr>
<tr>
<td>EBP</td>
<td></td>
<td></td>
</tr>
<tr>
<td>CR3</td>
<td></td>
<td></td>
</tr>
<tr>
<td>SS 2</td>
<td></td>
<td></td>
</tr>
<tr>
<td>ESP L</td>
<td></td>
<td></td>
</tr>
<tr>
<td>SS 1</td>
<td></td>
<td></td>
</tr>
<tr>
<td>ESP 1</td>
<td></td>
<td></td>
</tr>
<tr>
<td>SS 0</td>
<td></td>
<td></td>
</tr>
<tr>
<td>ESP 0</td>
<td></td>
<td></td>
</tr>
<tr>
<td>SS 1</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Back Link</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
X86 Extension to 3 Page Sizes

Linear Address

UNUSED

CR3

A

1G PAGE

B

C

2M PAGE

D

4K PAGE

C'

PFN
IBM SEGMENTATION

\[ 2^n \]

\[ n \gg m \]