TANU SHREE

CSCE611 OS

Week 3 HOMEWORK

1.) Given, virtual address space = 36 bit
page size = 8K
per page table entry = 4B.

a.) No. of pages = $\frac{\text{Virtual memory size}}{\text{page size}}$

$= \frac{2^{36}\text{ B.}}{2^{13}\text{ B}}$

$= 2^{23}$ pages.

b.) Per page table entry is 4 bytes i.e. 32 bits, so, it can reference 2^{32} pages.
Since, the size of each page is 2^{13} B, the maximum possible physical memory size is

$2^{32} \times 2^{13} = 2^{45}$ B i.e. 32TB.

c.) The virtual memory size is 2^{36} B, so, no. of pages that can be referenced by the page table is

$\frac{2^{36}\text{B}}{2^{13}}$ i.e 2^{23} pages.

Each page entry size is 4 bytes.
So total size of the page table is $2^{23} \times 4 = 2^{25}$ B
i.e. 32MB.

Now, given that process size is 8GB.
32 MB size page table will occupy a lot of process memory space. (hence costly).

∴ 1st level page table will not be good.

2nd level - The virtual address can be divided up as

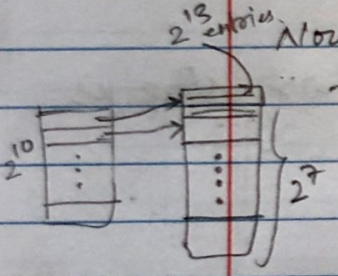
P_1	P_2	
10	13	13

Now, since the process size is 2^{33}B ∴ No of pages to be referenced are 2^{20} pages.

Inner table P_2 can refence 2^{20} pages and since each entry in inner table can be 13 bits i.e 2^{13} entries, we need $\frac{2^{20}}{2^{13}} = 2^7$ such entries in inner table.

Outer table can have 2^{10} entries (outer table is page directory), So, size of outer table which is to be stored in physical memory is $2^{10} \times 4$ bytes = 4kB

4kB occupies low space so it is good.



Now,

total memory in 2nd level:

outer table size + Inner table size

$1 \times 2^{10} \times 4 + 2^7 \times 2^{13} \times 4 \approx 4$MB.

3rd level paging - Suppose we divide virtual address as,

P_1	P_2	P_3	
9	7	7	13

Similarly as above, to address 2^{20} pages in inner table P_3, we need $\frac{2^{20}}{2^7} = 2^{13}$ inner tables (because each inner tables can have 2^7 entries. i.e 2^7 pages)

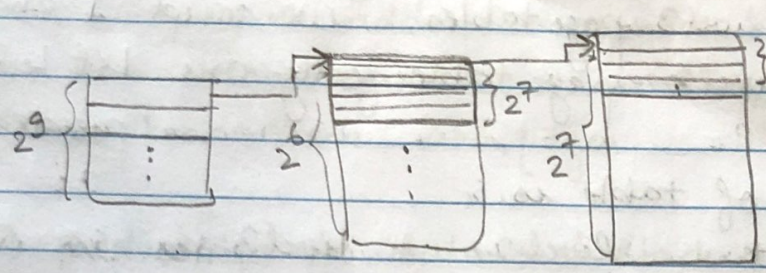
Each entry in 2nd level table P_2, points to 2^{13} 3rd level page table entries. So we need $\frac{2^{13}}{2^7} = 2^6$ such level 2 tables.

And we one level 1 table to point to 2^6 level 2 page tables.

So, total size of tables

level 1 + level 2 + level 3

$1 \times 2^9 \times 4B + 2^6 \times 2^7 \times 4B + 2^{13} \times 2^7 \times 4 \approx 4MB$.



Here we that for both level 2 and level 3 paging, the total memory occupied is 4MB. But level 2 is simple. So level-2 is better choice.

d.) The average size of the table as calculated above is 4MB.

2.) Give, virtual address = 32 bit i.e. 2^{32} B.

divided as,

P_1	P_2	P_3	
10	8	6	8

clearly, level 3 paging is used.

a.) We see that the last 8-bits are the offset of page
∴ the page size is 2^8 bytes i.e. 256 bytes.

b.) Now given, process size = 256K i.e. 2^{18} B.
i.e. the pages required to accessed are

$$\frac{2^{18}}{2^8} = 2^{10} \text{ pages.}$$

Now, here we see that,
level 1 table P_1 points to 2^{10} entries (1024) of P_2
level 2 table P_2 points to 2^8 entries (256) of P_3
level 3 table P_3 has 64 entries.

We have 2^{10} pages to be addressed.
So, we need $\frac{2^{10}}{2^6} = 2^4 = 16$ level 3 page tables P_3.

To access 16 level 3 page tables, we need 1 2nd level table
To access 1 level 2 page table, only one 1st level is needed.

∴ Total size of table is;

level 1 + level 2 + level 3

$1 \times 1024 \times 4 + 1 \times 256 \times 4 + 16 \times 64 \times 4 =$

(Assume 4 bytes per entry)

$= 4096 + 1024 + 4096$

$= 9216$ bytes

c.) Given, in a process,

code segment - 48k
data segment - 600k
stack segment - 64k.

As we know, page size is 2^8 B and it is assumed the page entry size is 4B.

∴ no. of pages for code segment $= \frac{48K}{2^8}$

$= 48 \times 2^2$

$= 192$ pages.

No. of pages for data segment $= \frac{600k}{2^8} \Rightarrow 2400$ pages.

No. of pages of stack segment $= \frac{64K}{2^8} = 256$ pages.

For 48K code segment, we need $\frac{192}{64} = 3$, 3rd level tables to address 192 pages.
For 600K data segment, we need $\frac{2400}{64} = 37.5 \approx 38$ 3rd level tables to address 2400 pages.
For 64K stack segment, we need $\frac{256}{64} = 4$ 3rd level tables to address 256 pages.

To address code, data and stack we need 1, 1st level page table in which each entry has address to code data & stack having 3 page table entries. i.e. 1st level table points to 3 2nd level table.

So, Size of table,

level 1 + level 2 + level 3

$1+1024\times4 + 3\times256\times4 + (3+38+4)\times64\times4$

$= 4096 + 3072 + 11520$

$= 18688$ bytes.

3.) Given, a virtual machine 32 bits.
divided as.

8	4	8	12

Physical address = 44 bits
protection bits per page = 4 bits.

a.) The last 12-bits are page offset. So the size of the page is 2^{12} B i.e. 4KB.

b.)

P_1	P_2	P_3	
8	4	8	12

Using above division, we see that

1st level has $2^8 = 256$ entries each pointing to 2nd level
2nd level has $2^4 = 16$ entries, each pointing to 3rd level table
3rd level has $2^8 = 256$ entries each pointing to a page

We have a process size 64k,
So it contains $\frac{2^{16}}{2^{12}}$ pages i.e. $2^4 = 16$ pages.

To address 16 pages, we need 1 3rd level page table.
∴ 1 entry in 2nd level and one entry in 1st level page.
Now, physical address is 44 bits and page size is 4k ∴ 32 bits for frame no.

32	12

← 44 →

Given that table entry has 4 bit protection bit.
∴ the entry would be 32+4 = 36 bits. Round up to entire word aligned, it is 64 bits or 8 bytes.

∴ The page table size of 3rd level is

$$256 \times 8 = 2048 \text{ bytes.}$$

Now, for level 1 page table, we should not assume that in 2^{nd} level page table pages are aligned. So, full address should be stored.
each page entry is at least 44 bits. Rounding up to 64 bits (8 bytes).
∴ 1st level page table size
$= 256 \times 8 = 2048$

Similarly, 2nd page table also cannot assume page alignment. Hence 44 bits size page entry.

2nd level page table size = $16 \times 8 = 128$ Bytes.

∴ Total size of page table = 1st level + 2nd level + 3rd level
= 2048 + 128 + 2048
= 4224 bytes.

The total size of page table is 4224 bytes and each frame is 4kB ie 4096 bytes, 4224 bytes would need 2 such frame to be stored. 2 pages = $4096 \times 2 = 8192$. 2nd page will not be fully utilized (only 128 out of 4096 utilized). So there will be internal fragmentation & $4096 - 128 = 3968$ bytes will be wasted.

4224
- 4096
128

c.) We know that,
1st level page table has 256 entries
2nd level page table has 16 entries
3rd level page table has 256 entries.

For code segment, we have $\frac{48k}{4k} = 12$ page. To access 12 pages we need 1 3rd level page table.

For data segment, we have $\frac{600k}{4k} = 150$ pages and to access 150 pages we need 1 3rd level page table

For stack segment we have $\frac{64k}{4k} = 16$ pages and to access 16 page we need 1 3rd level page table.

we have 1 entry in 2nd level page table, and 3 entries in 1st level page table to point code, data and stack.

∴ We need 3 2nd level page table & 1 1st level page table.

To calculate size of page table we consider protection bit in the page entry too.

∴ 3rd level page table size = (1+1+1) × 256 × 8
= 6144 bytes

(8 byte page entry because total bits are 32+4 = 36
To round up to next word, we use 64 bits i.e. 8 byte)

2nd level page table size = 3 × 16 × 8
= 384 bytes

1st level page table size = 1 × 256 × 8
= 2048 bytes

∴ Total size = 6144 + 384 + 2048
= 8576 bytes.

Here, the page table size is 8579 bytes. each frame size is 4096 bytes. So, it would need 3 frames to get stored. 2 frames are fully utilized by 3rd is not.

8576 - 8192 = 384 bytes are stored in 3rd frame.

∴ 4096 - 384 = 3172 bytes gets wasted in the 3rd frame.

So, internal fragmentation occurs and 3172 bytes get wasted.