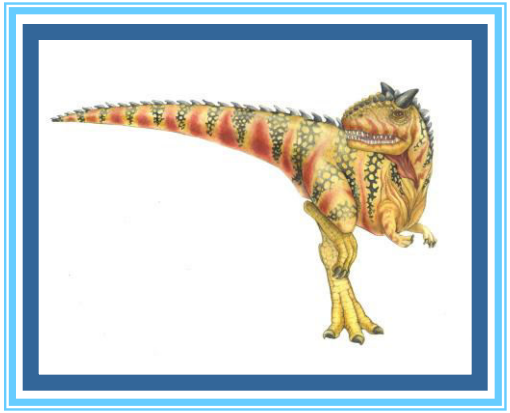
Structure of a Page Table



Structure of Page Table

- ***Logical Address Space:*** *A logical address space, in the context of computer systems, refers to the range of addresses that a program can use to reference data in the memory.*
- *Most modern computer systems support a large logical address space (**2^{32} to 2^{64}**).*
- *In such an environment, the page table itself becomes excessively large. As a result, the page table can not be stored in a single frame in main memory.*

Example

- *Consider a 32-bit logical address space (process size 2^{32}) with page size = 4KB (2^{12}).*
- *Then there will be total and 2^{20} = 1M entries in the page table.*
- *Assuming each entry of 4 bytes then each process may need up to 4MB of physical address space for the page table alone*

Structure of Page Table

Common techniques for structuring the page table are:

- ***Hierarchical paging (Multilevel Paging)***
- ***Hashed page tables***
- ***Inverted page tables.***

Hierarchical paging

- *The need for multilevel paging arises when- The **size of page table is greater than the frame size**. As a result, **the page table can not be stored in a single frame** in main memory.*
- *In multilevel paging,*
  - *The page table having size greater than the frame size is divided into several parts.*
  - *The size of each part is same as frame size except possibly the last part.*
  - *The pages of page table are then stored in different frames of the main memory.*
  - *To keep track of the frames storing the pages of the divided page table, another page table is maintained.*
  - *As a result, the hierarchy of page tables get generated.*
  - ***Multilevel paging is done till the level is reached where the entire page table can be stored in a single frame.***

Hierarchical paging

Logical Address Space – 32 bit

Physical Address = 32 bit

Page Size = 4kb

Page Table Entry Size = 4 Byte

Page Table Size ?

Process Size = 2^{32}

No. of entries in page Table = $2^{32}/2^{12} = 2^{20}$

Size of one entry = 4 Byte = 2^2 Byte

Total size of page table = $2^{20} * 2^2 = 2^{22} = 4$ Mb

Page Table Size > Page Size

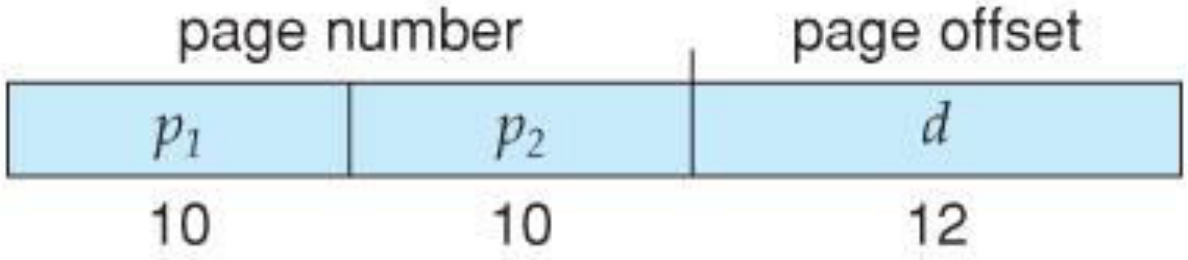
Hierarchical paging

Will divide page Table into equal size of Page

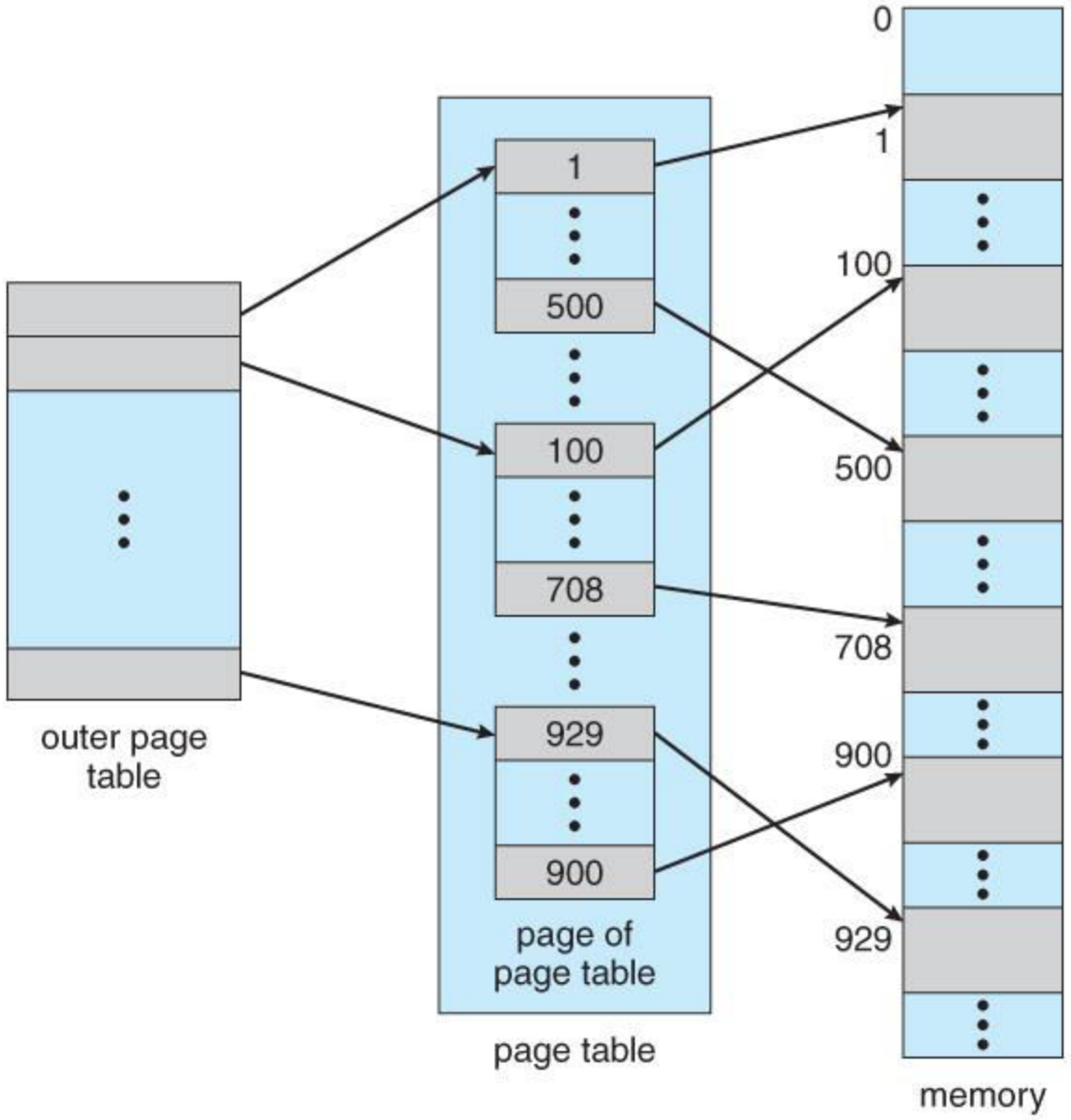
Total Pages of page table = $2^{22}/2^{12} = 2^{10}$

Outer Page table will consist of 2^{10} *Entries*

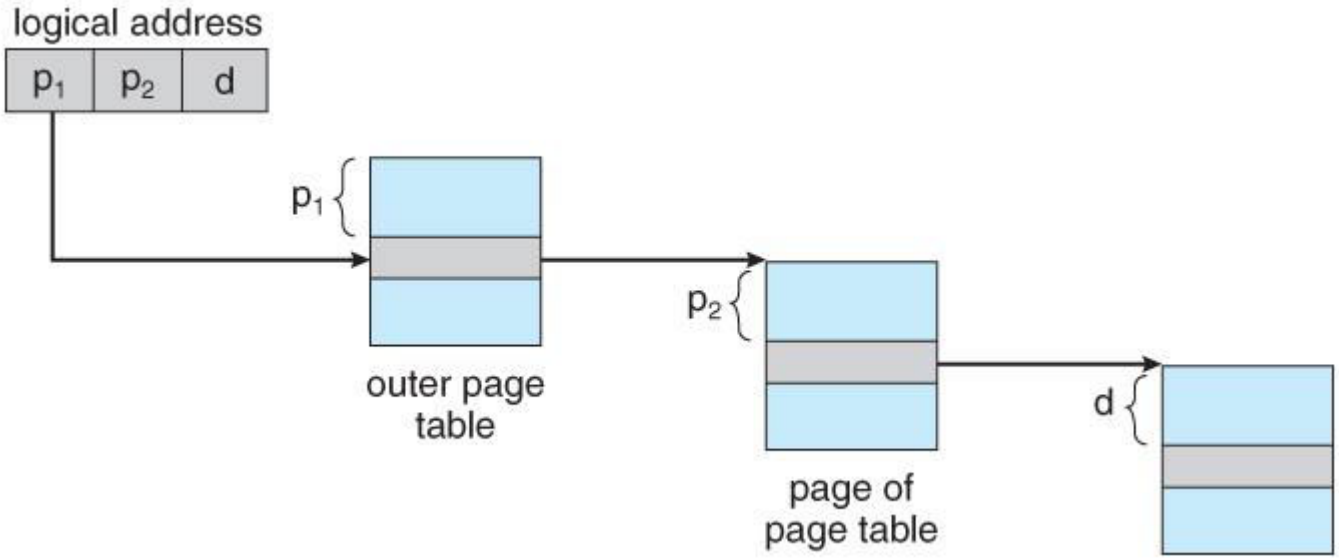
Inner Page table consist of $2^{12} / 2^{2} = 2^{10}$ *Entries*



Hierarchical paging



Hierarchical paging



Example-1

Consider a virtual memory system with physical memory of 8GB, a page size of 8KB, and a 46-bit virtual address. Assume every page table exactly fits into a single page. If page table entry size is 4B then how many levels of page tables would be required?

Page size = 8KB = 2^{13} B

Virtual address space size = 2^{46} B

Page table entry size = 4B = 2^{2} B

Number of pages or number of entries in page table, = (virtual address space size) / (page size)

= $2^{46}B/2^{13}$ B

= 2^{33}

Example-1

Page Table Size =

$= (\text{number of entries in page table}) * (\text{size of PTE})$

$= 2^{33} * 2^{2}\ B$

$= 2^{35}\ B$

Size of Page Table > Page Size

Number of page tables in last level,

$= 2^{35}\ B / 2^{13}\ B$

$= 2^{22}$

Size of page table [second last level]

$= 2^{22} * 2^{2} B$

$= 2^{24} B$

Size of Page Table > Page Size

Example-1

Number of page tables in second last level

$= 2^{24}B/2^{13}\ B$

$= 2^{11}$

Size of page table [Third last level]

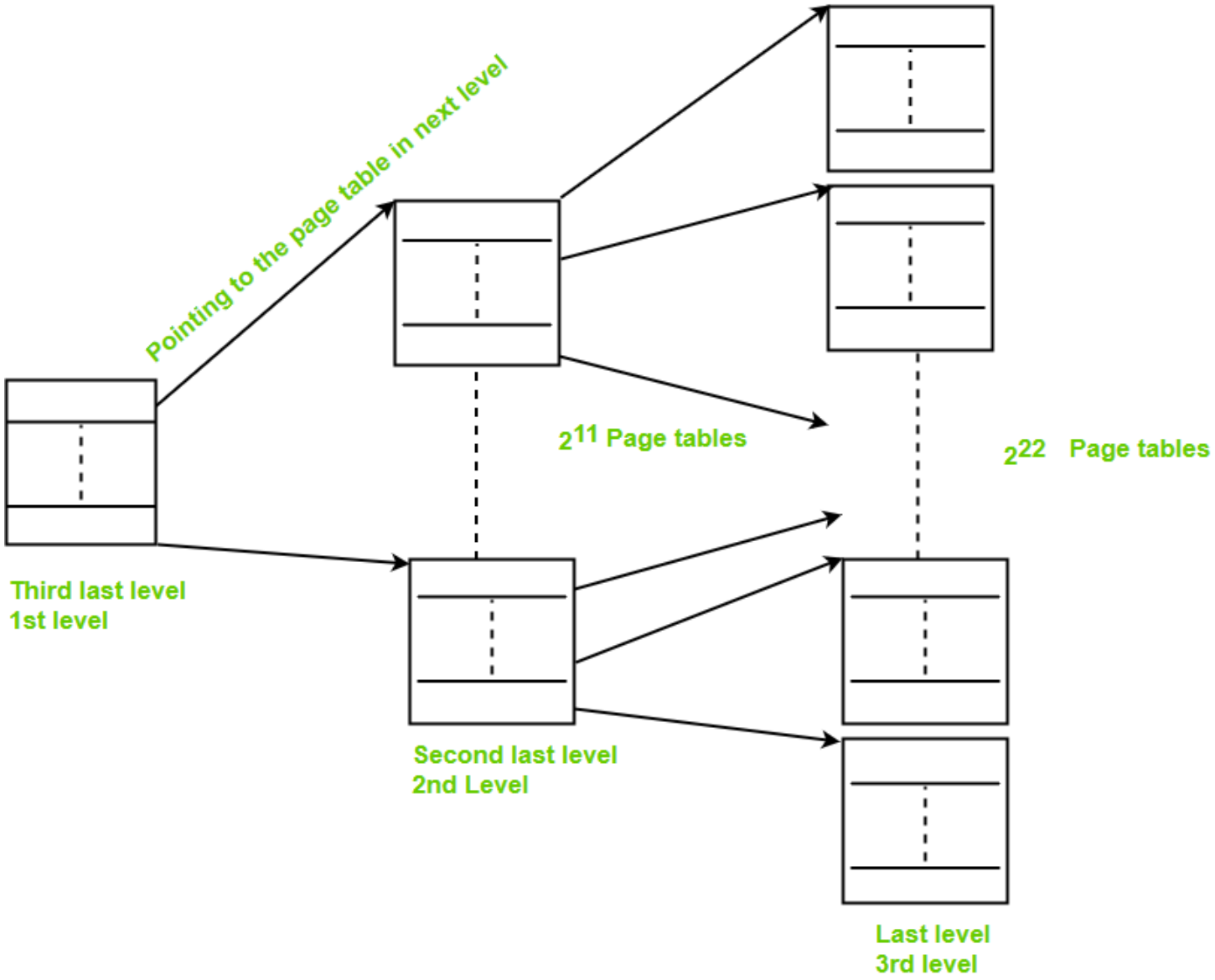
$= 2^{11} * 2^{2}B$

$= 2^{13}B$

Size of Page Table = Page Size

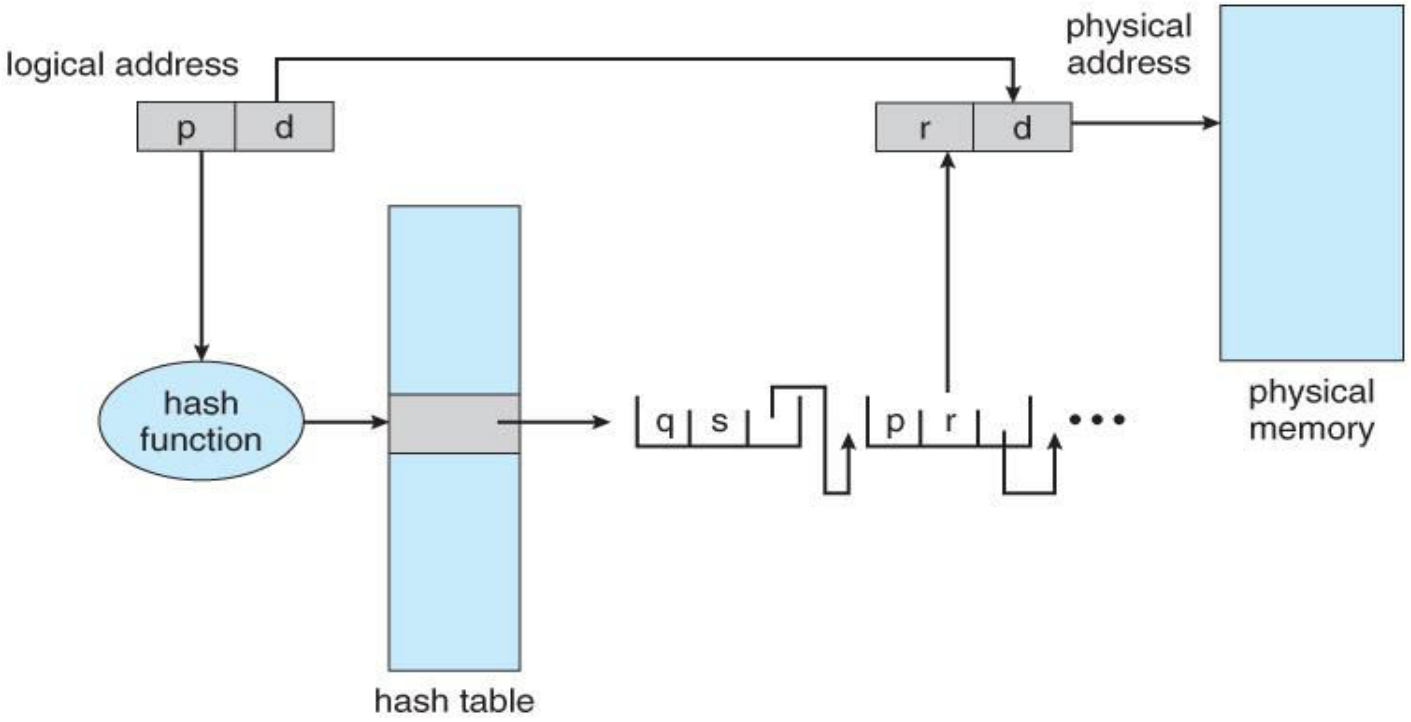
3 levels are required.

Example-1


Pointing to the page table in next level
2^{11} Page tables
2^{22} Page tables
Third last level
1st level
Second last level
2nd Level
Last level
3rd level

Hashed Page Tables

- *Virtual page number is hashed into the hash table.*
- *Each element consists of three entries: i) The virtual page numbers, ii)The value of matched page frames, iii) A pointer to the next element in the linked list.*



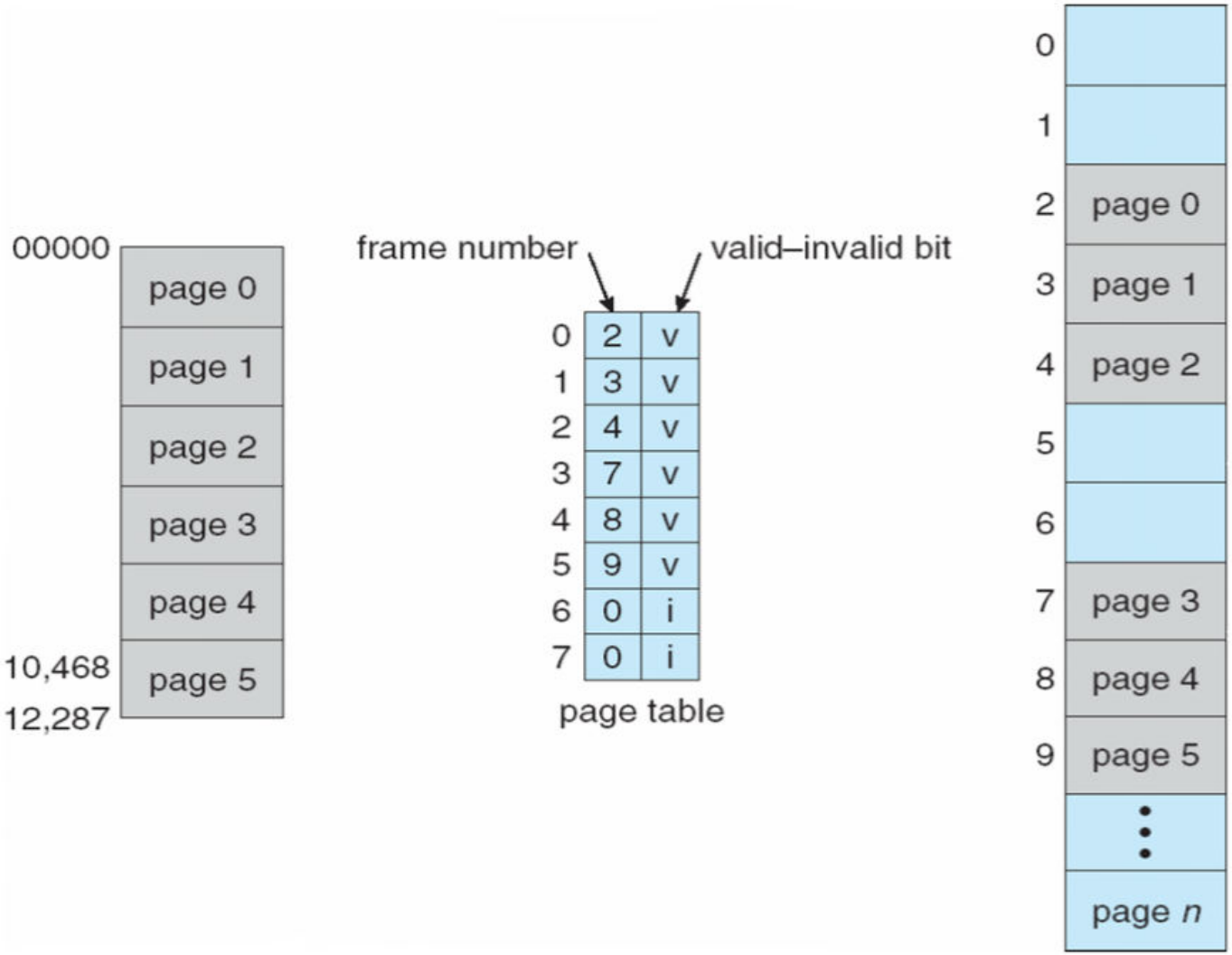
Protection

- ***Memory protection*** *implemented by associating a* ***protection (Read / Write) bit*** *with each frame. These bits are kept in the page table*
- *This bit is concerned with the page protection.*
- *It specifies the permission to perform read and write operation on the page.*
- *If only* ***read operation*** *is allowed to be performed and no writing is allowed, then this bit is set to* ***0****.*
- *If both* ***read and write*** *operation are allowed to be performed, then this bit is set to* ***1****.*

Protection

- ***Valid/invalid*** *bit attached to each entry in the page table:*
  - *When this bit is set to valid, the associated page is in the process's logical address space and is thus a legal (or valid) page.*
  - *When the bit is set to invalid, the page is not in the process's logical address space.*
  - *Illegal addresses are trapped by use of the valid–invalid bit.*

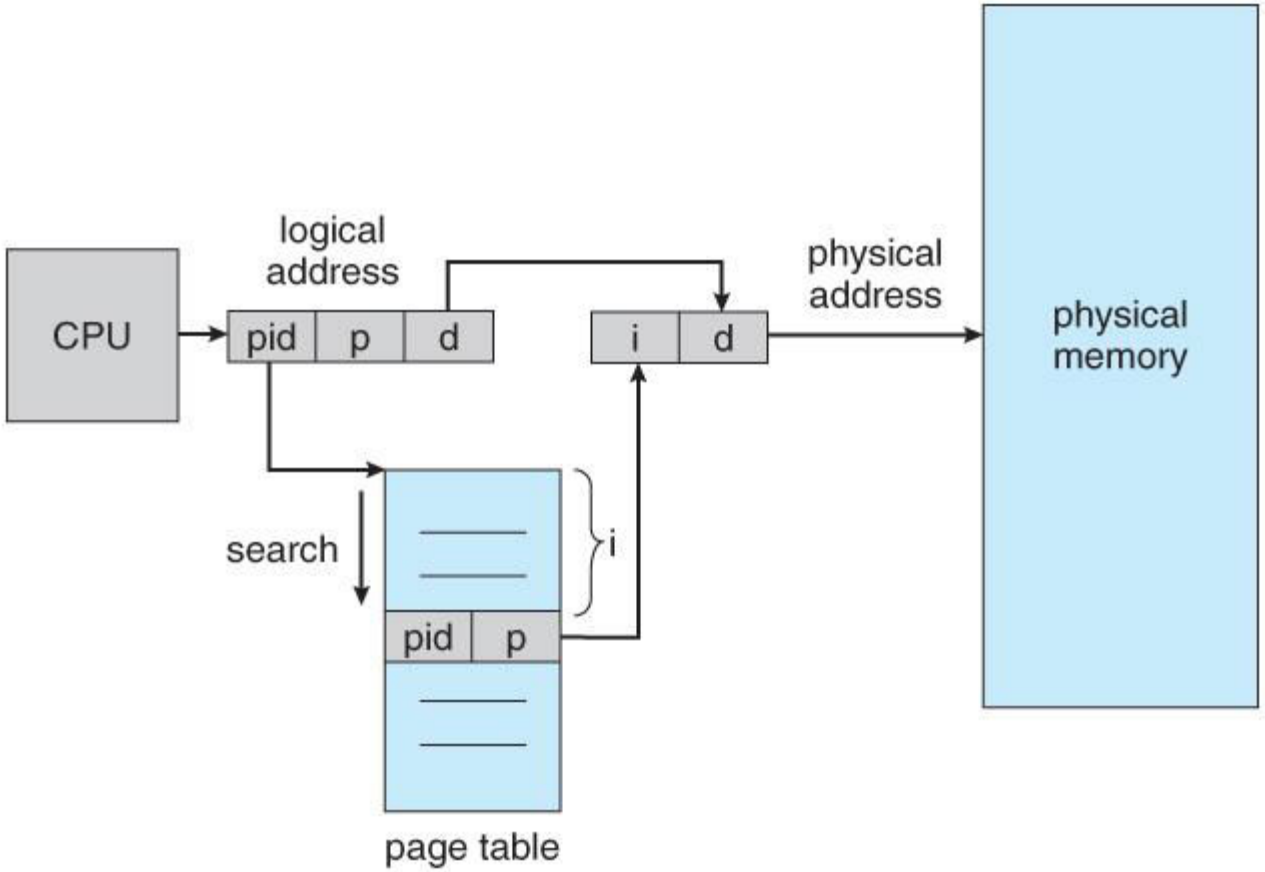
Memory Protection



Inverted Page Table

- *Another approach is to use an inverted page table. Instead of a table listing all of the pages for a particular process, an inverted page table lists all of the pages currently loaded in memory, for all processes. (I.e. there is one entry per frame instead of one entry per page.)*
- *Access to an inverted page table can be slow, as it may be necessary to search the entire table in order to find the desired page (or to discover that it is not there.) Hashing the table can help speedup the search process.*
- *Inverted page tables prohibit the normal method of implementing shared memory, which is to map multiple logical pages to a common physical frame. (Because each frame is now mapped to one and only one process.)*

Inverted Page Table



Problem 2

Consider a system using multilevel paging scheme. The page size is 1 MB. The memory is byte addressable and virtual address is 64 bits long. The page table entry size is 4 bytes.

Find-

1. *How many levels of page table will be required?*
2. *Give the divided physical address and virtual address.*
3. *Find out the size of main memory*

Problem 2

Given-

Virtual Address = 64 bits

Page size = 1 MB

Page table entry size = 4 bytes

We have, Page table entry size = 4 bytes = 32 bits

Thus, Number of bits in frame number = 32 bits

Number of Frames in Main Memory-

We have, Number of bits in frame number = 32 bits

Thus, Number of frames in main memory = 2^{32} frames

Problem 2

Size of Main Memory-

*Size of main memory = Total number of frames * Frame size = 2^{32} * 1 MB = 2^{52} B*

Thus, Number of bits in physical address = 52 bits

Number of Bits in Page Offset-

We have, Page size = 1 MB = 2^{20} B

Thus, Number of bits in page offset = 20 bits

Process Size-

Number of bits in virtual address = 64 bits

Thus,

Process size

= 2^{64} bytes

Number of Pages of Process-

Number of pages the process is divided

= Process size / Page size

= 2^{64} B / 1 MB

= 2^{64} B / 2^{20} B

= 2^{44} pages

Problem 2

Inner Page Table Size-

Inner page table keeps track of the frames storing the pages of process.

Inner page table size

= Number of entries in inner page table x Page table entry size

= Number of pages the process is divided x Page table entry size

= 2^{44} x 4 bytes

= 2^{46} bytes

Now, we can observe-

- The size of inner page table is greater than the frame size (1 MB).
- Thus, inner page table can not be stored in a single frame.
- So, inner page table has to be divided into pages.

Problem 2

Number of Pages of Inner Page Table-

Number of pages the inner page table is divided

= Inner page table size / Page size

= 2^{46} B / 1 MB

= 2^{46} B / 2^{20} B

= 2^{26} pages

Now, these 2^{26} pages of inner page table are stored in different frames of the main memory.

Number of Page Table Entries in One Page of Inner Page Table-

Number of page table entries in one page of inner page table

= Page size / Page table entry size

= 1 MB / 4 B

= 2^{20} B / 2^{2} B

Problem 2

Number of Pages of Inner Page Table-

Number of pages the inner page table is divided

= Inner page table size / Page size

= 2^{46} B / 1 MB

= 2^{46} B / 2^{20} B

= 2^{26} pages

Now, these 2^{26} pages of inner page table are stored in different frames of the main memory.

Number of Page Table Entries in One Page of Inner Page Table-

Number of page table entries in one page of inner page table

= Page size / Page table entry size

= 1 MB / 4 B

= 2^{20} B / 2^{2} B

Problem 2

Number of Bits Required to Search an Entry in One Page of Inner Page Table-

One page of inner page table contains 2^{18} entries.

Thus,

Number of bits required to search a particular entry in one page of inner page table = 18 bits

Outer Page Table-1 Size-

Outer page table-1 is required to keep track of the frames storing the pages of inner page table.

Outer page table-1 size

= Number of entries in outer page table-1 x Page table entry size

= Number of pages the inner page table is divided x Page table entry size

= 2^{26} x 4 bytes

= 2^{28} bytes

= 256 MB

Problem 2

Number of Pages of Outer Page Table-1

Number of pages the outer page table-1 is divided

= Outer page table-1 size / Page size

= 256 MB / 1 MB

= 256 pages

Now, these 256 pages of outer page table-1 are stored in different frames of the main memory.

Number of Page Table Entries in One Page of Outer Page Table-1

Number of page table entries in one page of outer page table-1

= Page size / Page table entry size

Problem 2

= 1 MB / 4 B

= 2^{20} B / 2^{2} B

= 2^{18} entries

Number of Bits Required to Search an Entry in One Page of Outer Page Table-1

One page of outer page table-1 contains 2^{18} entries.

Thus,

Number of bits required to search a particular entry in one page of outer page table-1 = 18 bits

Outer Page Table-2 Size-

Outer page table-2 is required to keep track of the frames storing the pages of outer page table-1.

Outer page table-2 size

= Number of entries in outer page table-2 x Page table entry size

= Number of pages the outer page table-1 is divided x Page table entry size

= 256 x 4 bytes

= 1 KB

Problem 2

Now, we can observe-

- The size of outer page table-2 is less than the frame size (16 KB).
- Thus, outer page table-2 can be stored in a single frame.
- In fact, outer page table-2 will not completely occupy one frame and some space will remain vacant.
- So, for given system, we will have three levels of page table.
- Page Table Base Register (PTBR) will store the base address of the outer page table-2.

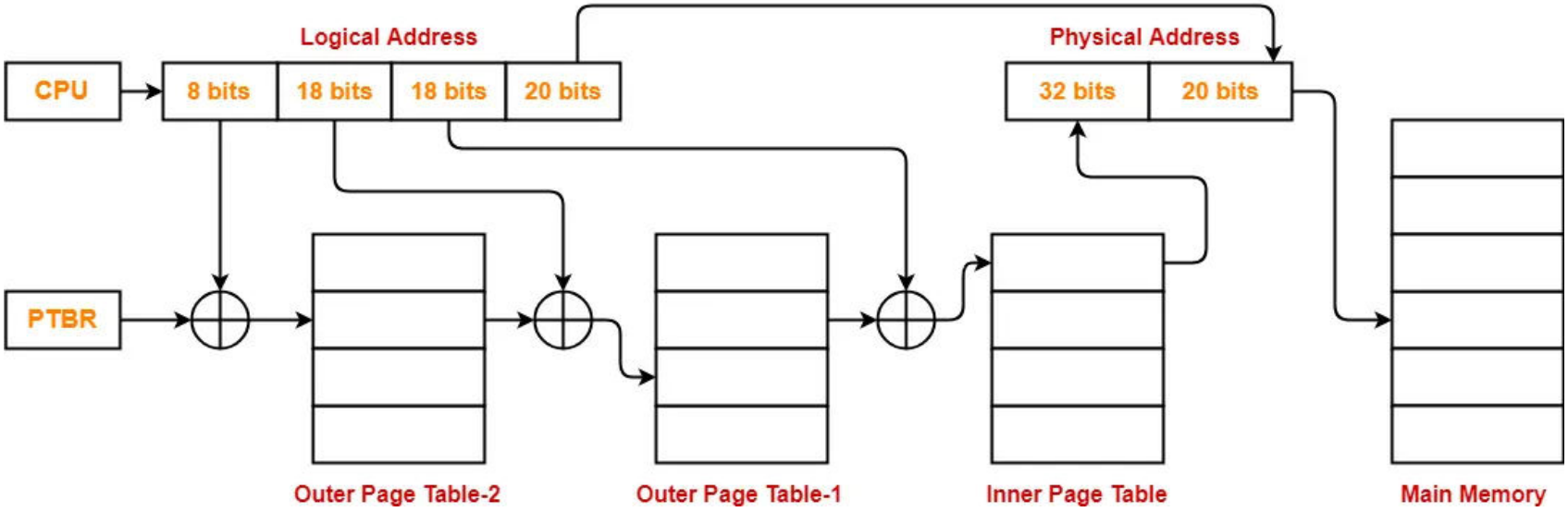
Number of Bits Required to Search an Entry in Outer Page Table-2

Outer page table-2 contains $256 = 2^8$ entries.

Thus,

Number of bits required to search a particular entry in outer page table-2 = 8 bits

Problem 2


Logical Address
Physical Address
CPU
8 bits
18 bits
18 bits
20 bits
32 bits
20 bits
PTBR
Outer Page Table-2
Outer Page Table-1
Inner Page Table
Main Memory

Problem 3

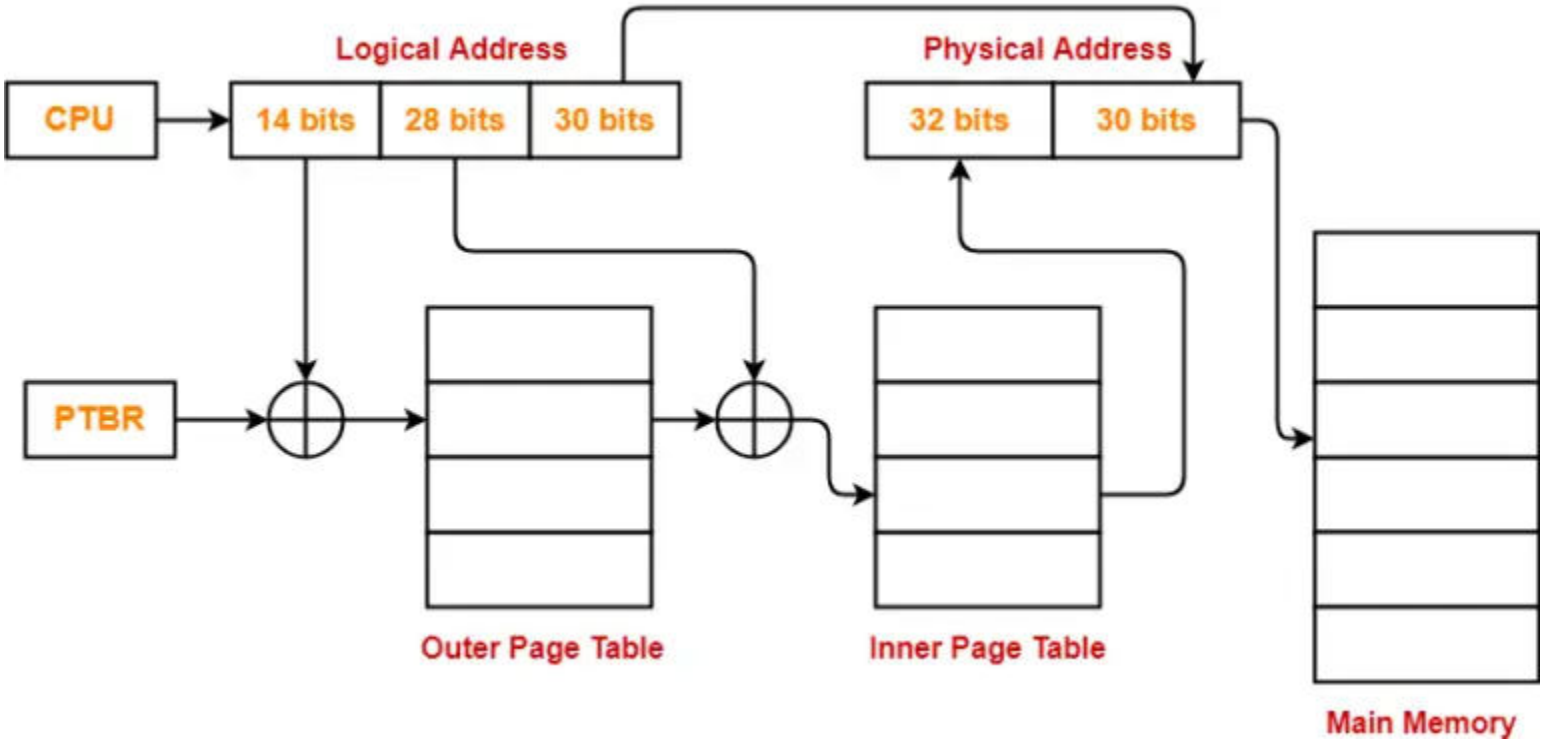
Consider a system using multilevel paging scheme. The page size is 1 GB. The memory is byte addressable and virtual address is 72 bits long. The page table entry size is 4 bytes.

Find-

1.How many levels of page table will be required?

2.Give the divided physical address and virtual address.

Problem 3


Logical Address
Physical Address
CPU
14 bits
28 bits
30 bits
32 bits
30 bits
PTBR
Outer Page Table
Inner Page Table
Main Memory

Problem 4

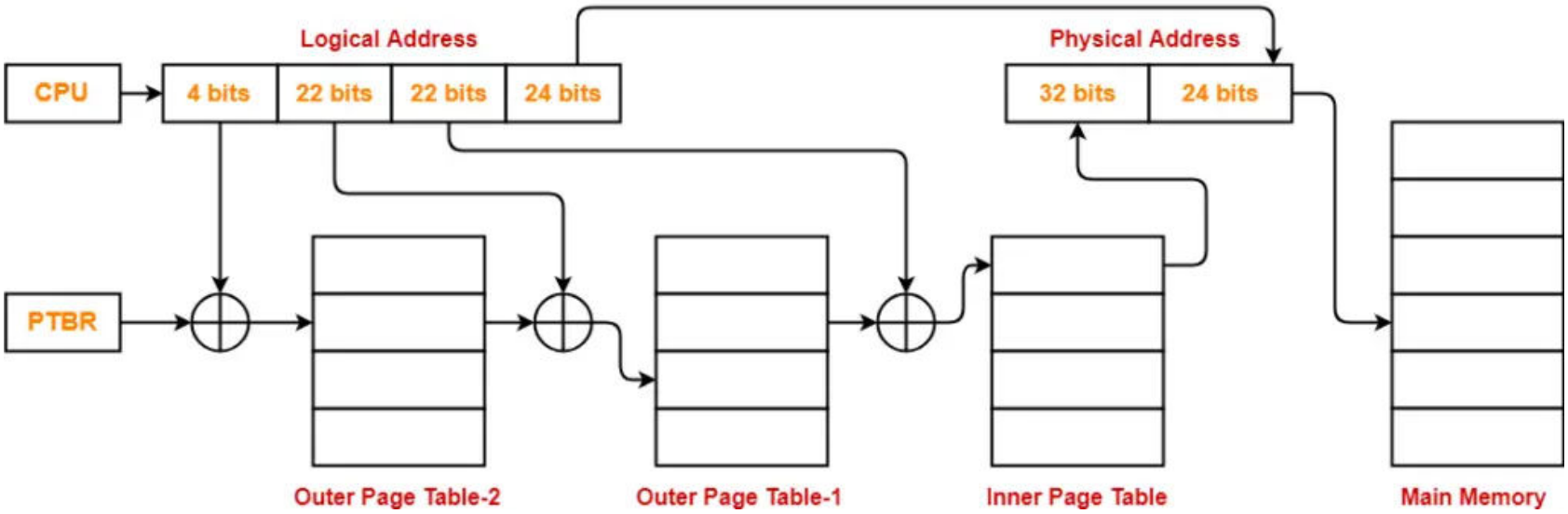
Consider a system using multilevel paging scheme. The page size is 16 MB. The memory is byte addressable and virtual address is 72 bits long. The page table entry size is 4 bytes.

Find-

1.How many levels of page table will be required?

2.Give the divided physical address and virtual address.

Problem 4


Logical Address
Physical Address
CPU
4 bits
22 bits
22 bits
24 bits
32 bits
24 bits
PTBR
Outer Page Table-2
Outer Page Table-1
Inner Page Table
Main Memory