The Concept of Virtual Memory in Computer Architecture

Virtual memory is a valuable concept in computer architecture that allows you to run large, sophisticated programs on a computer even if it has a relatively small amount of RAM. A computer with virtual memory artfully juggles the conflicting demands of multiple programs within a fixed amount of physical memory. A PC that's low on memory can run the same programs as one with abundant RAM, although more slowly.

Physical vs Virtual Addresses

A computer accesses the contents of its RAM through a system of addresses, which are essentially numbers that locate each byte. Because the amount of memory varies from PC to PC, determining which software will work on a given computer becomes complicated. Virtual memory solves this problem by treating each computer as if it has a large amount of RAM and each program as if it uses the PC exclusively. The operating system, such as Microsoft Windows or Apple's OS X, creates a set of virtual addresses for each program. The OS translates virtual addresses into physical ones, dynamically fitting programs into RAM as it becomes available.

Paging

Virtual memory breaks programs into fixed-size blocks called pages. If a computer has abundant physical memory, the operating system loads all of a program's pages into RAM. If not, the OS fits as much as it can and runs the instructions in those pages. When the computer is done with those pages, it loads the rest of the program into RAM, possibly overwriting earlier pages. Because the operating system automatically manages these details, this frees the software developer to concentrate on program features and not worry about memory issues.

Multiprogramming

Virtual memory with paging lets a computer run many programs at the same time, almost regardless of available RAM. This benefit, called multiprogramming, is a key feature of modern PC operating systems, as they accommodate many utility programs such as printer drivers, network managers and virus scanners at the same time as your applications -- Web browsers, word processors, email and media players.

Paging File

With virtual memory, the computer writes program pages that have not been recently used to an area on the hard drive called a paging file. The file saves the data contained in the pages; if the program needs it again, the operating system reloads it when RAM becomes available. When many programs compete for RAM, the act of swapping pages to the file can slow a computer's processing speed, as it spends more time doing memory management chores and less time getting useful work done. Ideally, a computer will have enough RAM to handle the demands of many programs, minimizing the time the computer spends managing its pages.

Memory Protection

A computer without virtual memory can still run many programs at the same time, although one program might change, accidentally or deliberately, the data in another if its addresses point to the wrong program. Virtual memory prevents this situation because a program never "sees" its physical addresses. The virtual memory manager protects the data in one program from changes by another.

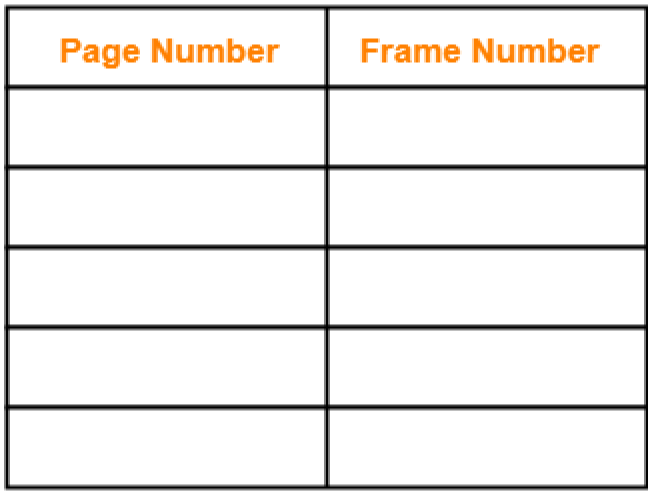
Translation Lookaside Buffer (TLB) in Paging

Translation Lookaside Buffer-

Translation Lookaside Buffer (TLB) is a solution that tries to reduce the effective access time.
Being a hardware, the access time of TLB is very less as compared to the main memory.

Structure-

Translation Lookaside Buffer (TLB) consists of two columns-
Page Number
Frame Number



Page Number	Frame Number

Translation Lookaside Buffer

Translating Logical Address into Physical Address-

In a paging scheme using TLB,
The logical address generated by the CPU is translated into the physical address using following steps-

Step-01:

CPU generates a logical address consisting of two parts-
Page Number
Page Offset

Step-02:

TLB is checked to see if it contains an entry for the referenced page number.
The referenced page number is compared with the TLB entries all at once.

Now, two cases are possible-

Case-01: If there is a TLB hit-

If TLB contains an entry for the referenced page number, a TLB hit occurs.
In this case, TLB entry is used to get the corresponding frame number for the referenced page number.

Case-02: If there is a TLB miss-

If TLB does not contain an entry for the referenced page number, a TLB miss occurs.
In this case, page table is used to get the corresponding frame number for the referenced page number.
Then, TLB is updated with the page number and frame number for future references.

Step-03:

After the frame number is obtained, it is combined with the page offset to generate the physical address.
Then, physical address is used to read the required word from the main memory.

PRACTICE PROBLEMS BASED ON TRANSLATION LOOKASIDE BUFFER-

Effective memory access time(EMAT) : TLB is used to reduce effective memory access time as it is a high speed associative cache.

EMAT = h*(c+m) + (1-h)*(c+2m)

where, h = hit ratio of TLB
m = Memory access time
c = TLB access time

Problem-01:

A paging scheme uses a Translation Lookaside buffer (TLB). A TLB access takes 10 ns and a main memory access takes 50 ns. What is the effective access time (in ns) if the TLB hit ratio is 90% and there is no page fault?

54
60
65
75

Solution-

Given-

TLB access time = 10 ns

Main memory access time = 50 ns
TLB Hit ratio = 90% = 0.9

Calculating TLB Miss Ratio-

TLB Miss ratio

= 1 – TLB Hit ratio

= 1 – 0.9

= 0.1

Calculating Effective Access Time-

Substituting values in the above formula, we get-

Effective Access Time

= 0.9 x { 10 ns + 50 ns } + 0.1 x { 10 ns + 2 x 50 ns }

= 0.9 x 60 ns + 0.1 x 110 ns

= 54 ns + 11 ns

= 65 ns

Thus, Option (C) is correct.

Problem-02:

A paging scheme uses a Translation Lookaside buffer (TLB). The effective memory access takes 160 ns and a main memory access takes 100 ns. What is the TLB access time (in ns) if the TLB hit ratio is 60% and there is no page fault?

54
60
20
75

Solution-

Given-

Effective access time = 160 ns
Main memory access time = 100 ns
TLB Hit ratio = 60% = 0.6

Calculating TLB Miss Ratio-

TLB Miss ratio

= 1 – TLB Hit ratio

= 1 – 0.6

= 0.4

Calculating TLB Access Time-

Let TLB access time = T ns.

Substituting values in the above formula, we get-

160 ns = 0.6 x { T + 100 ns } + 0.4 x { T + 2 x 100 ns }

160 = 0.6 x T + 60 + 0.4 x T + 80

160 = T + 140

T = 160 – 140

T = 20

Thus, Option (C) is correct.

Multilevel Paging-

Multilevel paging is a paging scheme where there exists a hierarchy of page tables.

Need –

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.

Working-

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.

Illustration of Multilevel Paging-

Consider a system using paging scheme where-

Logical Address Space = 4 GB
Physical Address Space = 16 TB
Page size = 4 KB

Now, let us find how many levels of page table will be required.

Number of Bits in Physical Address-

Size of main memory

= Physical Address Space

= 16 TB

= 244 B

Thus, Number of bits in physical address = 44 bits

Number of Frames in Main Memory-

Number of frames in main memory

= Size of main memory / Frame size

= 16 TB / 4 KB

= 232 frames

Thus, Number of bits in frame number = 32 bits

Number of Bits in Page Offset-

We have,

Page size

= 4 KB

= 212 B

Thus, Number of bits in page offset = 12 bits

Alternatively,

Number of bits in page offset

= Number of bits in physical address – Number of bits in frame number

= 44 bits – 32 bits

= 12 bits

So, Physical address is-

Number of Pages of Process-

Number of pages the process is divided

= Process size / Page size

= 4 GB / 4 KB

= 220 pages

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 Number of bits in frame number

= 220 x 32 bits

= 220 x 4 bytes

= 4 MB

Now, we can observe-

The size of inner page table is greater than the frame size (4 KB).
Thus, inner page table can not be stored in a single frame.
So, inner page table has to be divided into pages.

Number of Pages of Inner Page Table-

Number of pages the inner page table is divided

= Inner page table size / Page size

= 4 MB / 4 KB

= 210 pages

Now, these 210 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

= Page size / Number of bits in frame number

= 4 KB / 32 bits

= 4 KB / 4 B

= 210

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

One page of inner page table contains 210 entries.

Thus,

Number of bits required to search a particular entry in one page of inner page table = 10 bits

Outer Page Table Size-

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

Outer Page table size

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

= Number of pages the inner page table is divided x Number of bits in frame number

= 210 x 32 bits

= 210 x 4 bytes

= 4 KB

Now, we can observe-

The size of outer page table is same as frame size (4 KB).
Thus, outer page table can be stored in a single frame.
So, for given system, we will have two levels of page table.
Page Table Base Register (PTBR) will store the base address of the outer page table.

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

Outer page table contains 210 entries.

Thus,

Number of bits required to search a particular entry in outer page table = 10 bits

The paging system will look like as shown below-

PRACTICE PROBLEM BASED ON MULTILEVEL PAGING-

Problem-

Consider a system using multilevel paging scheme. The page size is 16 KB. The memory is byte addressable and virtual address is 48 bits long. The page table entry size is 4 bytes.

Find-

How many levels of page table will be required?
Give the divided physical address and virtual address.

Solution-

Given-

Virtual Address = 48 bits
Page size = 16 KB
Page table entry size = 4 bytes

Number of Bits in Frame Number-

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

= 232 frames

Size of Main Memory-

Size of main memory

= Total number of frames x Frame size

= 232 x 16 KB

= 246 B

= 64 TB

Thus, Number of bits in physical address = 46 bits

Number of Bits in Page Offset-

We have,

Page size

= 16 KB

= 214 B

Thus, Number of bits in page offset = 14 bits

Alternatively,

Number of bits in page offset

= Number of bits in physical address – Number of bits in frame number

= 46 bits – 32 bits

= 14 bits

Process Size-

Number of bits in virtual address = 48 bits

Thus,

Process size

= 248 bytes

= 256 TB

Number of Pages of Process-

Number of pages the process is divided

= Process size / Page size

= 256 TB / 16 KB

= 248 B / 214 B

= 234 pages

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

= 234 x 4 bytes

= 236 bytes

= 64 GB

Now, we can observe-

The size of inner page table is greater than the frame size (4 KB).
Thus, inner page table can not be stored in a single frame.
So, inner page table has to be divided into pages.

Number of Pages of Inner Page Table-

Number of pages the inner page table is divided

= Inner page table size / Page size

= 64 GB / 16 KB

= 236 B / 214 B

= 222 pages

Now, these 222 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

= 16 KB / 4 B

= 212 entries

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

One page of inner page table contains 212 entries.

Thus,

Number of bits required to search a particular entry in one page of inner page table = 12 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

= 222 x 4 bytes

= 16 MB

Now, we can observe-

The size of outer page table-1 is greater than the frame size (4 KB).
Thus, outer page table-1 can not be stored in a single frame.
So, outer page table-1 has to be divided into pages.

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

= 16 MB / 16 KB

= 210 pages

Now, these 210 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

= 16 KB / 4 B

= 212 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 212 entries.

Thus,

Number of bits required to search a particular entry in one page of outer page table-1 = 12 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

= 210 x 4 bytes

= 4 KB

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 210 entries.

Thus,

Number of bits required to search a particular entry in outer page table-2 = 10 bits

PRACTICE PROBLEMS BASED ON MULTILEVEL PAGING-

Problem-01:

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-
How many levels of page table will be required?
Give the divided physical address and virtual address.

Solution-

Given-
Virtual Address = 64 bits
Page size = 1 MB
Page table entry size = 4 bytes

Number of Bits in Frame Number-

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

Size of Main Memory-

Size of main memory
= Total number of frames x Frame size
= 2^{32} x 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

Alternatively,
Number of bits in page offset
= Number of bits in physical address – Number of bits in frame number
= 52 bits – 32 bits
= 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

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.

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
= 2^{18} entries

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

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

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
= 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

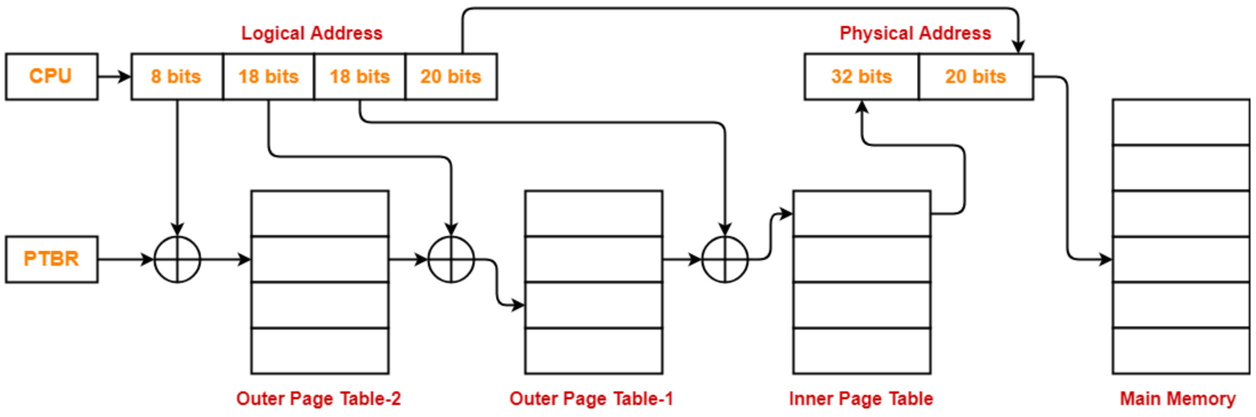
Now, we can observe-
The size of outer page table-2 is less than the frame size (16 KB).
Thus, outer page table-2 can be stored in a single frame.
In fact, outer page table-2 will not completely occupy one frame and some space will remain vacant.

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

The paging system will look like as shown below-



Problem-02:

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-
How many levels of page table will be required?
Give the divided physical address and virtual address.

Solution-

Given-
Virtual Address = 72 bits
Page size = 1 GB
Page table entry size = 4 bytes

Number of Bits in Frame Number-

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

Size of Main Memory-

Size of main memory
= Total number of frames x Frame size
= 2^{32} x 1 GB
= 2^{62} B
Thus, Number of bits in physical address = 62 bits

Number of Bits in Page Offset-

We have,
Page size
= 1 GB
= 2^{30} B
Thus, Number of bits in page offset = 30 bits

Alternatively,
Number of bits in page offset
= Number of bits in physical address – Number of bits in frame number
= 62 bits – 32 bits
= 30 bits

Process Size-

Number of bits in virtual address = 72 bits
Thus,
Process size
= 2^{72} bytes

Number of Pages of Process-

Number of pages the process is divided
= Process size / Page size
= 2^{72} B / 1 GB
= 2^{72} B / 2^{30} B
= 2^{42} pages

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^{42} x 4 bytes
= 2^{44} bytes

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

Number of Pages of Inner Page Table-

Number of pages the inner page table is divided
= Inner page table size / Page size
= 2^{44} B / 1 GB
= 2^{44} B / 2^{30} B
= 2^{14} pages

Now, these 2^{14} 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 GB / 4 B
= 2^{30} B / 2^{2} B
= 2^{28} entries

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

One page of inner page table contains 2^{28} entries.
Thus,
Number of bits required to search a particular entry in one page of inner page table = 28 bits

Outer Page Table Size-

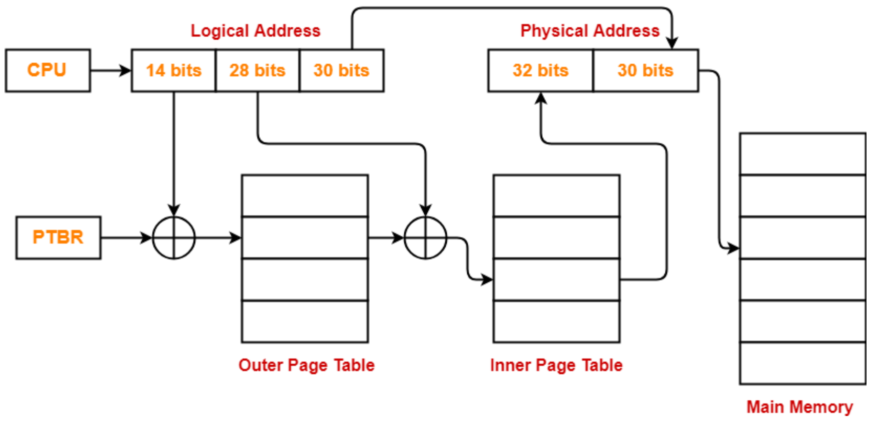
Outer page table is required to keep track of the frames storing the pages of inner page table.
Outer page table size
= Number of entries in outer page table x Page table entry size
= Number of pages the inner page table is divided x Page table entry size
= 2^{14} x 4 bytes
= 2^{16} bytes
= 64 KB

Now, we can observe-
The size of outer page table is less than the frame size (1 GB).
Thus, outer page table can be stored in a single frame.
In fact, outer page table will not completely occupy one frame and some space will remain vacant.
So, for given system, we will have two levels of page table.
Page Table Base Register (PTBR) will store the base address of the outer page table.

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

Outer page table contains 2^{14} entries.
Thus,
Number of bits required to search a particular entry in outer page table = 14 bits

The paging system will look like as shown below-



Problem-03:

Consider a system using multilevel paging scheme. The page size is 256 MB. The memory is byte addressable and virtual address is 72 bits long. The page table entry size is 4 bytes.
Find-
How many levels of page table will be required?
Give the divided physical address and virtual address.

Solution-

Given-
Virtual Address = 72 bits
Page size = 256 MB
Page table entry size = 4 bytes

Number of Bits in Frame Number-

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

Size of Main Memory-

Size of main memory
= Total number of frames x Frame size
= 2^{32} x 256 MB
= 2^{60} B
Thus, Number of bits in physical address = 60 bits

Number of Bits in Page Offset-

We have,
Page size
= 256 MB
= 2^{28} B
Thus, Number of bits in page offset = 28 bits

Alternatively,

Number of bits in page offset
= Number of bits in physical address – Number of bits in frame number
= 60 bits – 32 bits
= 28 bits

Process Size-

Number of bits in virtual address = 72 bits
Thus,
Process size
= 2^{72} bytes

Number of Pages of Process-

Number of pages the process is divided
= Process size / Page size
= 2^{72} B / 256 MB
= 2^{72} B / 2^{28} B
= 2^{44} pages

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 (256 MB).
Thus, inner page table can not be stored in a single frame.
So, inner page table has to be divided into pages.

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 / 256 MB
= 2^{46} B / 2^{28} B
= 2^{18} pages

Now, these 2^{18} 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
= 256 MB / 4 B
= 2^{28} B / 2^{2} B
= 2^{26} entries

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

One page of inner page table contains 2^{26} entries.
Thus,
Number of bits required to search a particular entry in one page of inner page table = 26 bits

Outer Page Table Size-

Outer page table is required to keep track of the frames storing the pages of inner page table.
Outer page table size
= Number of entries in outer page table x Page table entry size
= Number of pages the inner page table is divided x Page table entry size
= 2^{18} x 4 bytes
= 2^{20} bytes
= 1 MB

Now, we can observe-
The size of outer page table is less than the frame size (256 MB).
Thus, outer page table can be stored in a single frame.
In fact, outer page table will not completely occupy one frame and some space will remain vacant.
So, for given system, we will have two levels of page table.

Page Table Base Register (PTBR) will store the base address of the outer page table.

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

Outer page table contains 2^{18} entries.
Thus,
Number of bits required to search a particular entry in outer page table = 18 bits

Assignment Questions(any5)

1. Explain Virtual Memory with advantage a disadvantage
2. Explain Effective access time
3. In virtual memory phenomenon, how can data transfer from RAM to disk storage be measured & does the speed of the system has any effect?
4. Explain System Calls
5. Explain Address Translation
6. Explain Translation Lookaside Buffer with diagram.
7. Explain Parallel vs serial access, interaction with caching
8. What’s difference between CPU Cache and TLB?