Lecture: 31

GO CLASSES

Problem with Segmentation

↳ External fragmentation

→ We require contigious space for one segment.

→ if the segment wants to grow, then we need to reshuffle.

GO CLASSES

Problem with paging

↳ internal fragmentation

(it is very less and managable)

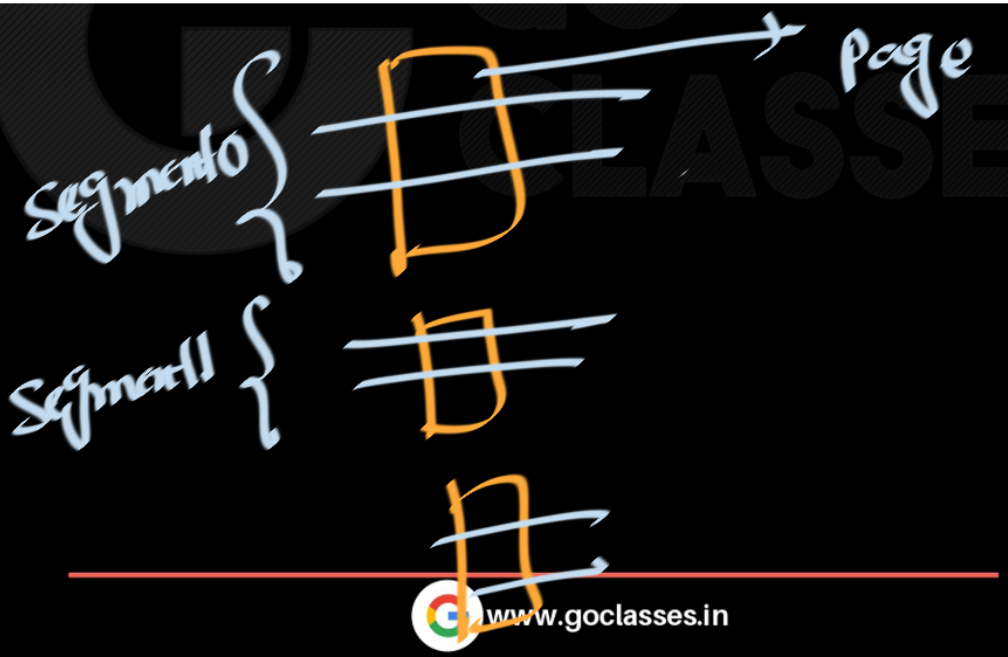
Segmentation with Paging

If the segments are large, it may be inconvenient, or even impossible, to keep them in main memory in their entirety. This leads to the idea of paging them.

we can page the segments itself

In a combined paging/segmentation system, a user's address space is broken up into a number of segments, at the discretion of the programmer. Each segment is, in turn, broken up into a number of fixed-size pages, which are equal in length to a main memory frame. If a segment has length less than that of a page, the segment occupies just one page.

william stallings .pdf



Virtual address

Seg #	Page #	Offset

Say we have 4 bits for Seg No.

= 2^4 = 16 Segments possible

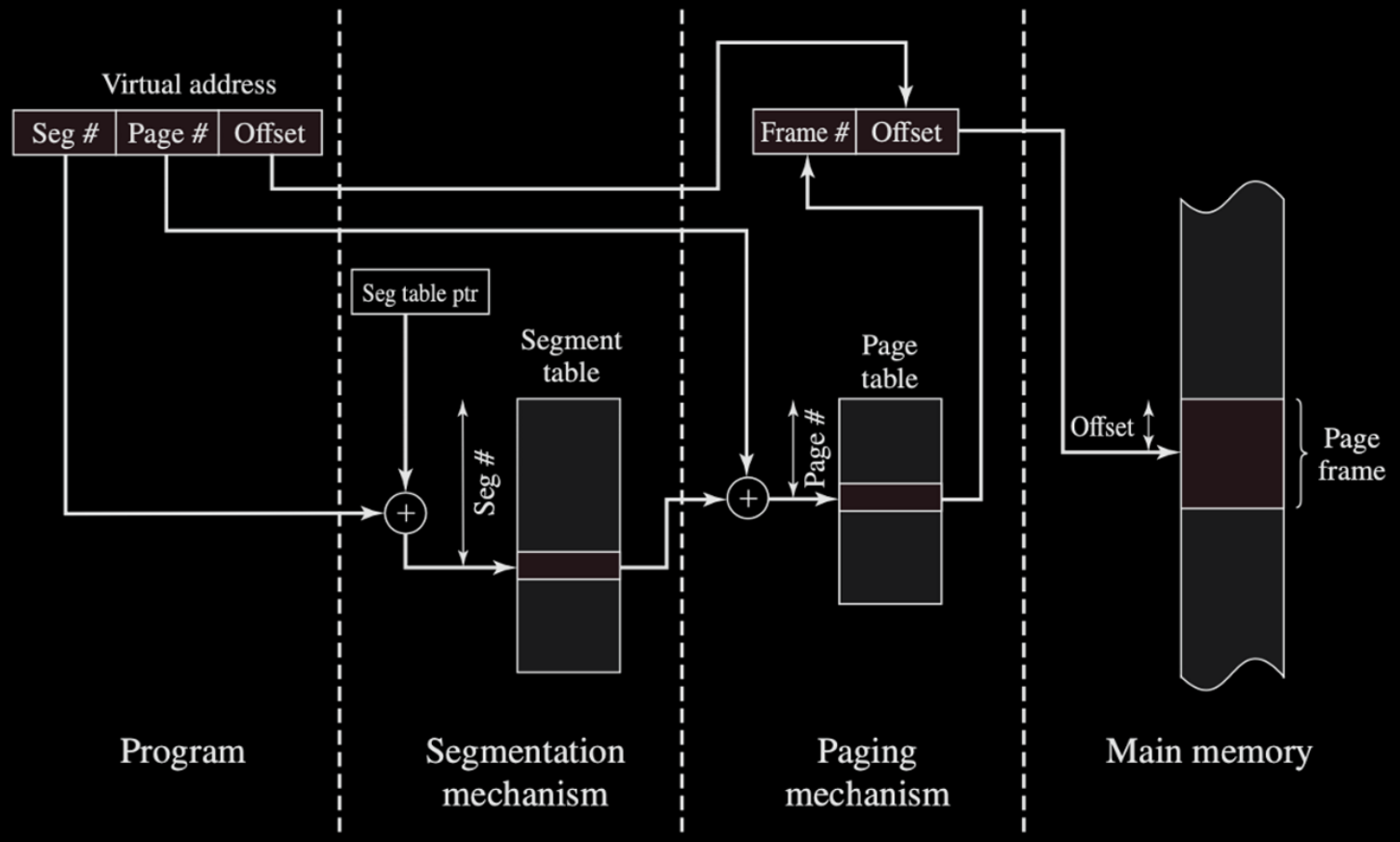
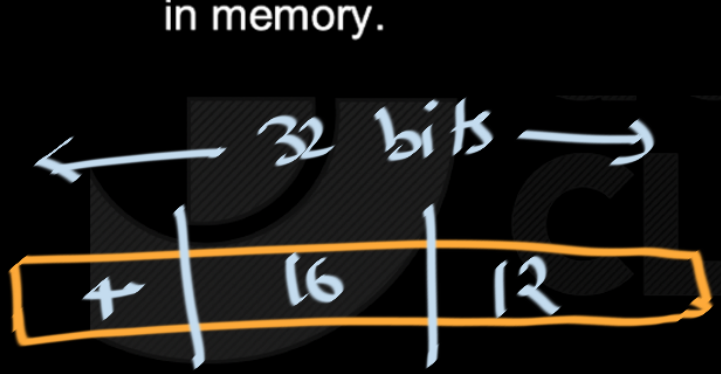


Figure 8.12 Address Translation in a Segmentation/Paging System

Question

1. [20] A processor has a 32 bit address space, and combines both segmentation and paging. 4 bits for the segment, 16 for the page, and 12 for the offset. A PTE is 4 bytes. Describe **in detail** what happens in the MMU and OS (use generic terms not the x86 terms) when:

   (a) [10] A user process does a read of address 0xC0DEDBAD and it is in memory.

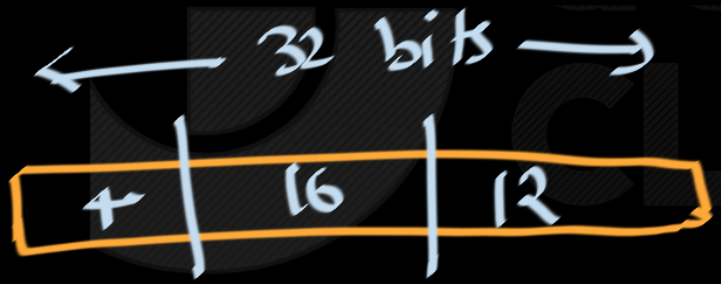


https://www.scs.stanford.edu/23wi-cs212/exams/cs140.sum06.midterm.sol.pdf

Question

1. [20] A processor has a 32 bit address space, and combines both segmentation and paging. 4 bits for the segment, 16 for the page, and 12 for the offset. A PTE is 4 bytes. Describe **in detail** what happens in the MMU and OS (use generic terms not the x86 terms) when:

   (a) [10] A user process does a read of address 0xC0DEDBAD and it is in memory.



Seg No. = C

Page No. = ODED

offset = BAD

https://www.scs.stanford.edu/23wi-cs212/exams/cs140.sum06.midterm.sol.pdf

Question

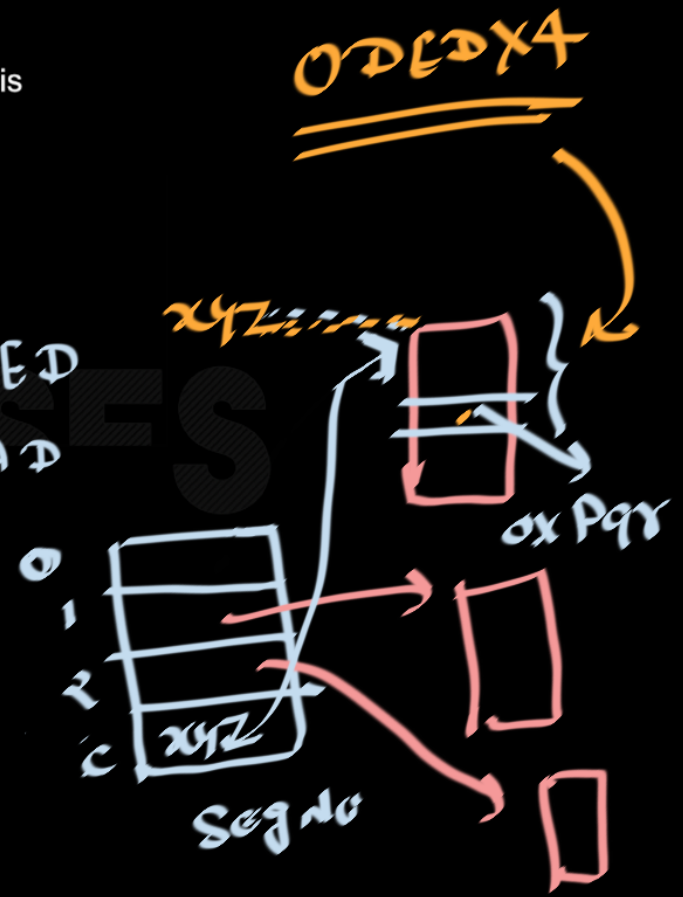
1. [20] A processor has a 32 bit address space, and combines both segmentation and paging. 4 bits for the segment, 16 for the page, and 12 for the offset. A PTE is 4 bytes. Describe **in detail** what happens in the MMU and OS (use generic terms not the x86 terms) when:

   (a) [10] A user process does a read of address 0xC0DEDBAD and it is in memory.

Seg No. = C

Page No. = 0DED

offset = BAD



https://www.scs.stanford.edu/23wi-cs212/exams/cs140.sum06.midterm.sol.pdf

PQR 30

0x PQR BAD

Question

1. [20] A processor has a 32 bit address space, and combines both segmentation and paging. 4 bits for the segment, 16 for the page, and 12 for the offset. A PTE is 4 bytes. Describe **in detail** what happens in the MMU and OS (use generic terms not the x86 terms) when:

   (a) [10] A user process does a read of address 0xC0DEDBAD and it is in memory.

   Segment 0xC, Page 0x0DED, Offset 0xBAD.

   Look at the 0x0DED'th PTE (0x0DED * 4 bytes into the page table) and see what physical page it is in.
   Add 0xBAD to that physical page number shifted 12 bits to the left.
   Access memory.

0x Page BAD

https://www.scs.stanford.edu/23wi-cs212/exams/cs140.sum06.midterm.sol.pdf

Question

Page = 2^{16} B

Consider a processor that uses segmentation and paging.

For the first parts of this question (parts a – b) assume that the processor is using 32-bits for virtual and physical addresses, that the page size is 64 KB, that all addresses (virtual and physical) , and that the system uses 4 bits for segments.

a. **(2 mark(s))** Explain how many **bits** of the virtual address will be used to represent the offset?

b. **(2 mark(s))** What is the **maximum possible size** of a segment in this system in **bytes** (expressed as an equation).

https://cs.uwaterloo.ca/~brecht/courses/350/W13-midterm-sol.pdf

a. **(2 mark(s))** Explain how many **bits** of the virtual address will be used to represent the offset?

b. **(2 mark(s))** What is the **maximum possible size** of a segment in this system in **bytes** (expressed as an equation).

each seg. can have = 2^{12} pages

= $2^{12} \times 2^{16}$ Bytes = 2^{28} Bytes

= 256 MB

a. **(2 mark(s))** Explain how many **bits** of the virtual address will be used to represent the offset?

begin solution

2^{16} = 64 KB, so 16 bits for the offset

end solution

b. **(2 mark(s))** What is the **maximum possible size** of a segment in this system in **bytes** (expressed as an equation).

begin solution

32 virtual address - 4 for segment - 16 bits for page size/offset = 12 bits for a virtual page number and each page is 2^{16} bytes so $2^{12} * 2^{16} = 2^{28}$.

or more simply

32 bit virtual address - 4 bits for the segment = 28 bits. So 2^{28}.

end solution

Question

Consider a multi-level memory management scheme using the following format for virtual addresses:

Virtual seg # (2 bits)	Virtual Page # (6 bits)	Offset (12 bits)

Virtual addresses are translated into physical addresses of the following form:

Physical Page # (8 bits)	Offset (12 bits)

Page table entries (PTE) are 16 bits in the following format, *stored in big-endian form* in memory (i.e. the MSB is first byte in memory):

Physical Page # (8 bits)	Kernel	Nocache	0	0	Dirty	Use	Writeable	Valid

1) How big is a page? Explain.

2) What is the maximum amount of virtual memory supported by this scheme? Explain

3) What is the maximum amount of physical memory supported by this scheme? Explain

4) max seg size =

https://cs162.org/static/exams/fa08-mt1-solutions.pdf

Question

Consider a multi-level memory management scheme using the following format for virtual addresses:

Virtual seg # (2 bits)	Virtual Page # (6 bits)	Offset (12 bits)

Virtual addresses are translated into physical addresses of the following form:

Physical Page # (8 bits)	Offset (12 bits)

Page table entries (PTE) are 16 bits in the following format, *stored in big-endian form* in memory (i.e. the MSB is first byte in memory):

Physical Page # (8 bits)	Kernel	Nocache	0	0	Dirty	Use	Writeable	Valid

1) How big is a page? Explain.

2^{12} B

2) What is the maximum amount of virtual memory supported by this scheme? Explain

2^{20}

3) What is the maximum amount of physical memory supported by this scheme? Explain

2^{20}

4) max seg size = 2^{18} B

5) page size

https://cs162.org/static/exams/fa08-mt1-solutions.pdf

Question

Consider a multi-level memory management scheme using the following format for virtual addresses:

Virtual seg # (2 bits)	Virtual Page # (6 bits)	Offset (12 bits)

Virtual addresses are translated into physical addresses of the following form:

Physical Page # (8 bits)	Offset (12 bits)

Page table entries (PTE) are 16 bits in the following format, *stored in big-endian form* in memory (i.e. the MSB is first byte in memory):

Physical Page # (8 bits)	Kernel	Nocache	0	0	Dirty	Use	Writeable	Valid

1) How big is a page? Explain.

We just have to look at the offset of 12 bits. Since 2^{12}= 4096, thus pages are 4096 bytes in size.

2) What is the maximum amount of virtual memory supported by this scheme? Explain

Since virtual addresses are 20 bits, the maximum amount of virtual memory is 2^{20} =1MB

3) What is the maximum amount of physical memory supported by this scheme? Explain

Since physical addresses are 20 bits, the maximum amount of physical memory is 2^{20}= 1MB

https://cs162.org/static/exams/fa08-mt1-solutions.pdf

GATE CSE 1999 | Question: 19

42

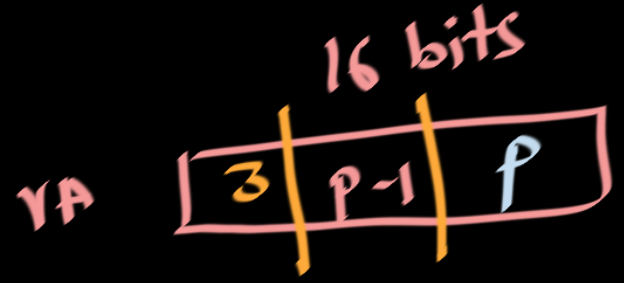
A certain computer system has the segmented paging architecture for virtual memory. The memory is byte addressable. Both virtual and physical address spaces contain 2^{16} bytes each. The virtual address space is divided into 8 non-overlapping equal size segments. The memory management unit (MMU) has a hardware segment table, each entry of which contains the physical address of the page table for the segment. Page tables are stored in the main memory and consists of 2 byte page table entries. = 16 bits

a. What is the minimum page size in bytes so that the page table for a segment requires at most one page to store it? Assume that the page size can only be a power of 2.

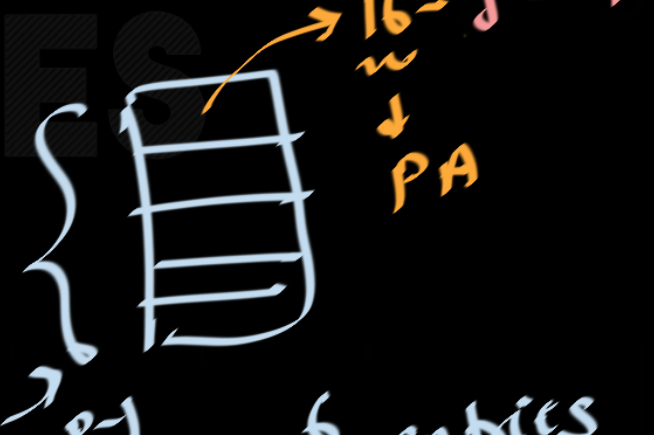
b. Now suppose that the pages size is 512 bytes. It is proposed to provide a TLB (Transaction look-aside buffer) for speeding up address translation. The proposed TLB will be capable of storing page table entries for 16 recently referenced virtual pages, in a fast cache that will use the direct mapping scheme. What is the number of tag bits that will need to be associated with each cache entry?

c. Assume that each page table entry contains (besides other information) 1 valid bit, 3 bits for page protection and 1 dirty bit. How many bits are available in page table entry for storing the aging information for the page? Assume that the page size is 512 bytes.

later



Page size = 2^p



7+1 3+1 = 12 bits

https://gateoverflow.in/1518/gate-cse-1999-question-19

39

Best answer

a. Size of each segment $= \frac{2^{16}}{8} = 2^{13}$

Let the size of page be 2^k bytes

We need a page table entry for each page. For a segment of size 2^{13}, number of pages required will be

2^{13-k} and so we need 2^{13-k} page table entries. Now, the size of these many entries must be less than or equal to the page size, for the page table of a segment to be requiring at most one page. So,

$2^{13-k} \times 2 = 2^k$ (As a page table entry size is 2 bytes)

$k = 7$ bits

So, page size $= 2^7 = 128$ bytes

b. The TLB is placed after the segment table.

Each segment will have $\frac{2^{13}}{2^9} = 2^4$ page table entries

So, all page table entries of a segment will reside in the cache and segment number will differentiate between page table entry of each segment in the TLB cache.

$\text{Total segments} = 8$

Therefore 3 bits of tag is required

c. Number of Pages for a segment $= \frac{2^{16}}{2^9} = 2^7$

Bits needed for page frame identification
$= 7$ bits
$+1$ valid bit
$+3$ page protection bits
$+1$ dirty bit
$= 12$ bits needed for a page table entry

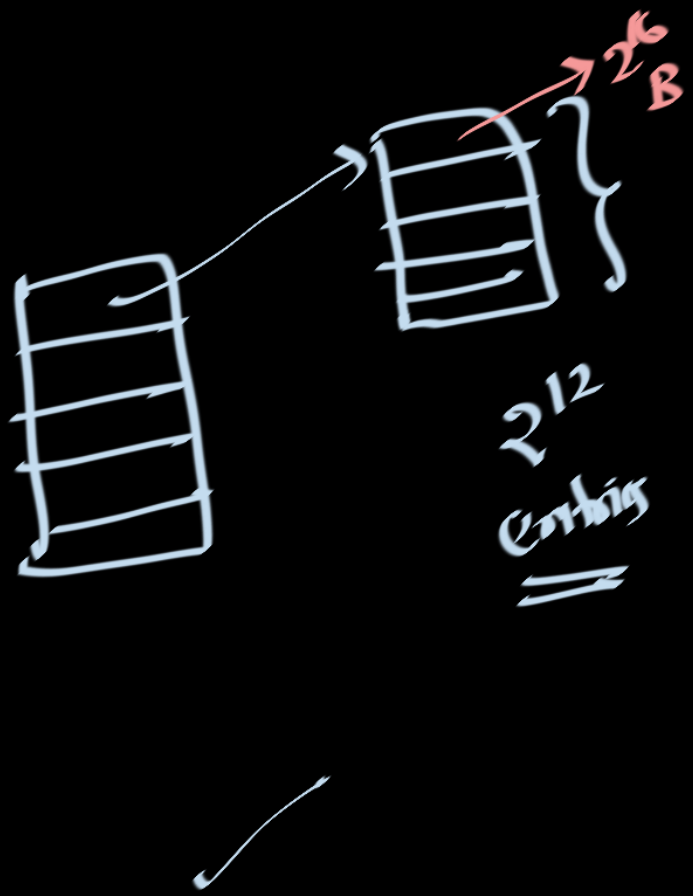
Size of each page table entry $= 2$ bytes $= 16$ bits

Number of bits left for aging $= 16 - 12 = 4$ bits

Question

Consider an architecture combining segmentation and paging with 32-bit addresses divided into fields as follows:

4-bit seg id	12 bit page number	16 bit offset into page



SegId	Page Table Ptr	Page Table Size
0	0x20000	0x4
1	0x30000	0x4
2	Not Valid	-----
3	0x10000	0x4
4	Not Valid	-----
…	Not Valid	-----
15	Not Valid	-----

https://people.eecs.berkeley.edu/~kubitron/courses/cs162-F08/exams/sp00mt2.pdf

Address	Value
0x10000	0x10
	0xA
	0xB
	0x5
…	…
0x20000	0x3
	0x7
	0x5
	0x9
…	…
0x30000	0x6
	0x4
	0x8
	0x11
…	…

Given the contents of physical memory shown on the left and the segment table above, what is the phyical address do the following virtual addresses map to (if a virtual address is invalid, state why)

0x00001234 =

0x10001234 =

0x11111234 =

0x20021234 =

0x30031234 =

4-bit seg id	12 bit page number	16 bit offset into page

SegId	Page Table Ptr	Page Table Size
0	0x20000	0x4
1	0x30000	0x4
2	Not Valid	-----
3	0x10000	0x4
4	Not Valid	-----
…	Not Valid	-----
15	Not Valid	-----

Address	Value
0x10000	0x10
	0xA
	0xB
	0x5
…	…
0x20000	0x3
	0x7
	0x5
	0x9
…	…
0x30000	0x6
	0x4
	0x8
	0x11
…	…

main memory

frame no

Given the contents of physical memory shown on the left and the segment table above, what is the phyical address do the following virtual addresses map to (if a virtual address is invalid, state why)

0x00001234 =

PTE = 1B

0 000 1234

4-bit seg id	12 bit page number	16 bit offset into page

0x3 1234

SegId	Page Table Ptr	Page Table Size
0	0x20000	0x4
1	0x30000	0x4
2	Not Valid	-----
3	0x10000	0x4
4	Not Valid	-----
…	Not Valid	-----
15	Not Valid	-----

Address	Value
0x10000	0x10
	0xA
	0xB
	0x5
…	…
0x20000	0x3
	0x7
	0x5
	0x9
…	…
0x30000	0x6
	0x4
	0x8
	0x11
…	…

main memory

Given the contents of physical memory shown on the left and the segment table above, what is the phyical address do the following virtual addresses map to (if a virtual address is invalid, state why)

0x30031234 =

PTE = 1B

3 003 1234

4-bit seg id	12 bit page number	16 bit offset into page

frame No. = 0x5

PA = 0x51234

SegId	Page Table Ptr	Page Table Size
0	0x20000	0x4
1	0x30000	0x4
2	Not Valid	-----
3	0x10000	0x4
4	Not Valid	-----
…	Not Valid	-----
15	Not Valid	-----

0x31234

0x61234

invalid - virtual page num. exceeds size of segment

segment 2 does is not valid

0x51234

(valid: 2 points for right address;

invalid: 1 pt for saying invalid; 1 pt for reason)

Question

Consider a processor that uses segmentation and paging (i.e., this is not a MIPS processor). Below is the segment table being used for the currently executing process and below that are pages tables for several processes in the system. Note that some processes may not use all of the available segments. Recall that VPN is the virtual page number, PFN is the physical frame number.

Segment	PT base addr
4	70700000
3	70200000
2	70500000
1	70300000
0	70100000

VPN	PFN
3	516
2	37
1	7731
0	6341

Base addr: 70100000

VPN	PFN
3	641
2	753
1	2577
0	517

Base addr: 70200000

VPN	PFN
3	65
2	77
1	567
0	672

Base addr: 70300000

VPN	PFN
3	5177
2	34
1	563
0	1641

Base addr: 70500000

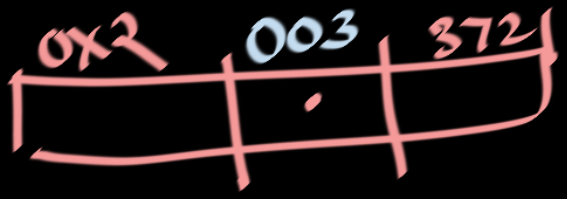
VPN	PFN
3	1311
2	12
1	711
0	23

Base addr: 70700000

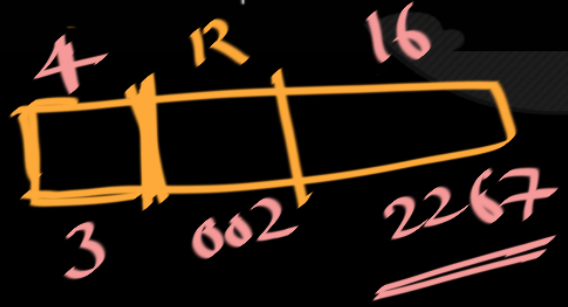
https://cs.uwaterloo.ca/~brecht/courses/350/W13-midterm-sol.pdf

GO CLASSES

0x5177 '3721



Segment	PT base addr
4	70700000
3	70200000
2	70500000
1	70300000
0	70100000



0x30022267

VPN	PFN
3	516
2	37
1	7731
0	6341

Base addr: 70100000

VPN	PFN
3	641
2	753
1	2577
0	517

Base addr: 70200000

VPN	PFN
3	65
2	77
1	567
0	672

Base addr: 70300000

VPN	PFN
3	5177
2	34
1	563
0	1641

Base addr: 70500000

VPN	PFN
3	1311
2	12
1	711
0	23

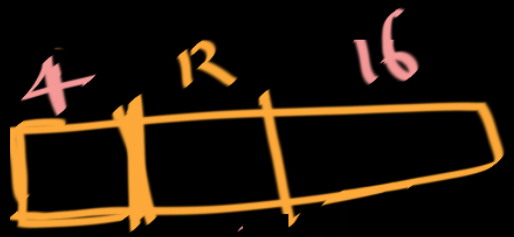
Base addr: 70700000

75322267 ← PA

Convert the **virtual address** 0x30022267 into a 32-bit physical addresses (also expressed in hexadecimal). Show your work and if the address can not be translated, explain why.

Good

Convert the **physical address** 0x51773721 into a 32-bit virtual addresses (also expressed in hexadecimal). Show your work and if the address can not be translated, explain why.



Convert the **virtual address** 0x30022267 into a 32-bit physical addresses (also expressed in hexadecimal). Show your work and if the address can not be translated, explain why.

begin solution

The first 4 bits are 0 so the segment is 3.
and the page table address is 0x70200000.
12 bits are used for the VPN so the VPN is 2.
The PFN for that VPN is 753
The last 16 bits are the offset which is 2267.
So the physical address is
0x0753 2267.

end solution

Convert the **physical address** 0x51773721 into a 32-bit virtual addresses (also expressed in hexadecimal). Show your work and if the address can not be translated, explain why.

begin solution

16 bits are used for the offset so 16 bits are used for the frame number.
So the PFN = 5177.
Now find an entry in one of the page tables for the executing process
with PFN = 5177.
The page table with base address 70500000 (segment 2) has PFN = 5177 in VPN 3.
So the VPN is 3 and the the segment is 2.
So the virtual address is 0x20033721.

end solution

GATE IT 2006 | Question: 56

62

For each of the four processes P_1, P_2, P_3, and P_4. The total size in kilobytes (KB) and the number of segments are given below.

Process	Total size (in KB)	Number of segments
P_1	195	4
P_2	254	5
P_3	45	3
P_4	364	8

The page size is 1 KB. The size of an entry in the page table is 4 bytes. The size of an entry in the segment table is 8 bytes. The maximum size of a segment is 256 KB. The paging method for memory management uses two-level paging, and its storage overhead is P. The storage overhead for the segmentation method is S. The storage overhead for the segmentation and paging method is T. What is the relation among the overheads for the different methods of memory management in the concurrent execution of the above four processes?

A. $P < S < T$
B. $S < P < T$
C. $S < T < P$
D. $T < S < P$

H·W

https://gateoverflow.in/3600/gate-it-2006-question-56

102

Best answer

For 2-level paging.

Page size is $1KB$. So, no. of pages required for $P_1 = 195$. An entry in page table is of size 4 bytes and assuming an inner level page table takes the size of a page (this information is not given in question), we can have up to 256 entries in a second level page table and we require only 195 for P_1. Thus only 1 second level page table is enough. So, memory overhead $= 1KB$ (for first level) (again assumed as page size as not explicitly told in question) $+1KB$ for second level $= 2KB$.

For P_2 and P_3 also, we get $2KB$ each and for P_4 we get $1 + 2 = 3KB$ as it requires 1 first level page table and 2 second level page tables $(364 > 256)$. So, total overhead for their concurrent execution $= 2 \times 3 + 3 = 9KB$.

Thus $P = 9KB$.

For Segmentation method

Ref: http://web.cs.wpi.edu/~cs3013/b02/week6-segmentation/week6-segmentation.html

P_1 uses 4 segments $\rightarrow$ 4 entries in segment table $= 4 \times 8 = 32$ bytes.

Similarly, for P_2, P_3 and P_4 we get 5×8, 3×8 and 8×8 bytes respectively and the total overhead will be $32 + 40 + 24 + 64 = 160$ bytes.

So, $S = 160B$.

For Segmentation with Paging

For Segmentation with Paging

Here we segment first and then page. So, we need the page table size. We are given maximum size of a segment is $256\ KB$ and page size is $1KB$ and thus we require 256 entries in the page table. So, total size of page table $= 256 \times 4 = 1024$ bytes (exactly 1 page size).

So, now for P_1 we require 1 segment table of size 32 bytes plus 4 page table of size $1KB$ for the 4 segments. Similarly,

$P_2 - 40$ bytes and $5\ KB$
$P_3 - 24$ bytes and $3\ KB$
$P_4 - 64$ bytes and $8\ KB$.

Thus total overhead $= 160$ bytes
$4\ KB + 5\ KB + 3\ KB + 8\ KB = 20480 + 160 = 20640$ bytes.

So, $T = 20640B$.

So, answer will be (**B**)- $S < P < T$.

answered Dec 21, 2015 • selected Nov 24, 2022 by **gatecse**

Question

Assume that a task is divided into four equal-sized segments and that the system builds an eight-entry page descriptor table for each segment. Thus, the system has a combination of segmentation and paging. Assume also that the page size is 2 Kbytes.

a. What is the maximum size of each segment?

b What is the maximum logical address space for the task?

c. Assume that an element in physical location 00021ABC is accessed by this task. What is the format of the logical address that the task generates for it? What is the maximum physical address space for the system?

https://www.scribd.com/presentation/104797549/Combined-Paging-and-Segmentation#

SOLUTION

A. (8 entries in the page table) x 2K = 16K.

B. (16 K) x 4 segments per task = 64K.

C. The physical address is 32 bits wide total, so the frame number must be 21 bits wide. Thus 00021ABC is represented in binary as:

Frame	Offset
0000 0000 0000 0010 0001 1	010 1011 1100

D. The maximum physical address space is 2^{32} = 4 GB

Question

- This question refers to an architecture using segmentation with paging. In this architecture, the 32-bit virtual address is divided into fields as follows:

  4 bit segment number
  12 bit page number
  16 bit offset

  Find the physical address corresponding to each of the following virtual addresses (answer "bad virtual address" if the virtual address is invalid)..

1. 00000000
2. 20022002
3. 10015555

	Segment table
0	Page table A
1	Page table B
x	(rest invalid)

	Page table A
0	CAFE
1	DEAD
2	BEEF
3	BA11
X	

	Page table B
0	F000
1	D8BF
X	(rest invalid)

SOLUTION:

1. CAFE0000.
2. Bad virtual address.
3. D8BF5555.

Question

Addresses in Segmented Paging: Example

- Given a memory size of 256 addressable words,
- a page table indexing 8 pages,
- a page size of 32 words, and
- 8 logical segments

- How many bits is a physical address?
- How many bits is a virtual address?
- How many bits for the seg #, page #, offset?
- How many segment table entries do we need?
- How many page table entries do we need?

https://lass.cs.umass.edu/~shenoy/courses/fall13/lectures/Lec13.pdf