Memory Problems with Solutions

Read terms & points carefully before going through the solution.

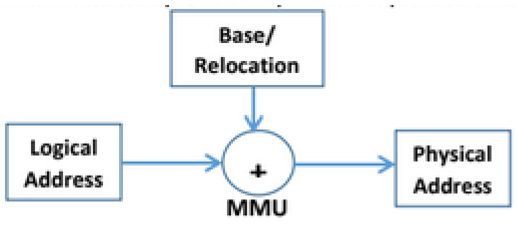
Some terms used in this solution:

p = page
f = frame
d = displacement / offset
d'= displacement / offset in case of paged segmentation
PS = Page Size
PTE = Page Table Entry
PTES = Page Table Entry Size
PTS = Page Table Size
s = Segment
LA = Logical Address
PA = Physical Address
IP = Instruction Pointer
TLB = Translation Look Ahead Buffer
EMA = Effective Memory Access

For page number, we will do integer division instead of complete division & remainder will be offset

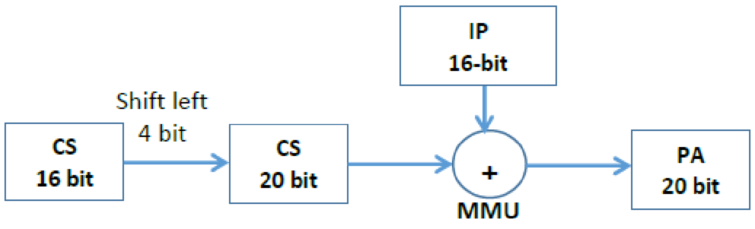
Problem 1

There is a Base / Relocation register, whose contents are added to every address generated by a user process at the time it is sent to memory. Draw its pictorial representation.



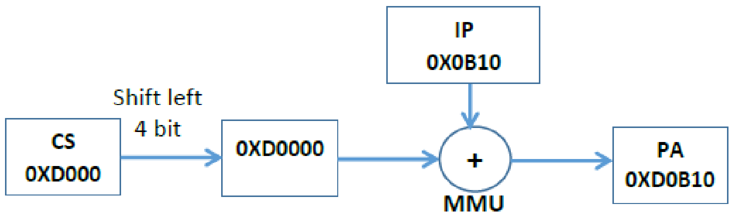
Problem 2

In i8086, the logical address (16 bits) of the next instruction is specified by the value of IP register. The physical address of the instruction is computed by shifting the CS register (16 bits) left by four bits and adding contents of IP to it, thus generating a physical address of 20 bits. Draw its pictorial representation.



Problem 3

In i8086, the contents of IP register are ox0B10 and contents of Code Segment contains oxD000. Compute the Physical address and also give the size of Logical and Physical address space.



Problem 4

Consider a process of size 1 MB. Disk to memory transfer rate is 5 MB per second. Consider the average disk latency to be 8 msec (assume no seek time). Calculate the total cost of swapping

Swapping involves both swap in and swap out, which means 2 MB data transfer.

Transfer time: 2/5 seconds + 8 msec = (0.4 * 1000 + 8) msec = 408 msec

Problem 5

Show how the available memory of 810 KB will accommodate following job sequence with all the four placement algorithms. Job Sequence:

J1 (90K), J2 (45K), J3 (180K), J4 (90K), J5 (135K), J6 (180K), J3 terminates, J5 terminates, J7 (135K), J8 (180K), J7 AND J8 TERMINATE, J9(285K**).**

i. First Fit

J1	J2	J3	J4	J5	J6	Empty
90 K	45 K	180 k	90 K	135 K	180 K	90 K

J3 & J5 terminates

J1	J2	Empty	J4	Empty	J6	Empty
90 K	45 K	180 k	90 K	135 K	180 K	90 K

J1	J2	J7	Empty	J4	Empty	J6	Empty
90 K	45 K	135 k	45 k	90 K	135 K	180 K	90 K

J8 & J9 not entertained due to unavailability of required memory.

ii. Next Fit is same as First Fit in this case

iii. Best Fit

J1	J2	J3	J4	J5	J6	Empty
90 K	45 K	180 k	90 K	135 K	180 K	90 K

J3 & J5 terminates

J1	J2	Empty	J4	Empty	J6	Empty
90 K	45 K	180 k	90 K	135 K	180 K	90 K

J1	J2	J8	J4	J7	J6	Empty
90 K	45 K	180 k	90 K	135 K	180 K	90 K

J1	J2	Empty	J4	Empty	J6	Empty
90 K	45 K	180 k	90 K	135 K	180 K	90 K

J9 not entertained due to unavailability of required memory.

iv. Worst Fit is same as First Fit in this case

Problem 6

Show how the available memory of 2560 KB will accommodate following job sequence with all the four placement algorithms. OS Kernel takes 400 KB. Job Sequence:

– P1 (600K), P2 (1000K), P3 (300K), P2 terminates, P4 (700K), P5 (500K)

Draw the pictorial representation using MVT

i. First Fit

P1	P2	P3	Empty
600 K	1000 K	300 k	660 K

P2 terminates

P1	Empty	P3	Empty
600 K	1000 K	300 k	660 K

P1	P4	Empty	P3	P5	Empty
600 K	700 K	300 K	300 k	500 K	160 K

ii. **Next Fit is same as First Fit in this case**

iii. **Best Fit is same as First Fit in this case**

iv. **Worst Fit is same as First Fit in this case**

Problem 7

Assume there is a 1,024-KB segment where memory is allocated using the buddy system. Using Figure 9.27 as a guide, draw a tree illustrating, how the following memory requests are allocated:
Request 240 bytes, 120 bytes, 60 bytes, 130 bytes
Next modify the tree for the following releases of memory. Perform coalescing whenever possible:
Release 240 bytes, 60 bytes, Release 120 bytes

1024

P1 (240)	**Empty**	**Empty**
256	256 B	512

P1 (240)	**P2 (120)**	**Empty**	**Empty**
256	128	128	512

P1 (240)	**P2 (120)**	**P3 (60)**	**Empty**	**Empty**
256	128	64	64	512

P1 (240)	**P2 (120)**	**P3 (60)**	**Empty**	**P4 (130)**	**Empty**
256	128	64	64	256	256

Release 240 bytes

Empty	**P2 (120)**	**P3 (60)**	**Empty**	**P4 (130)**	**Empty**
256	128	64	64	256	256

Release 60 bytes

Empty	**P2 (120)**	**Empty**	**P4 (130)**	**Empty**
256	128	128	256	256

Release 120 bytes

Empty	**Empty**	**P4 (130)**	**Empty**
256	256	256	256
Empty		**P4 (130)**	**Empty**
256		256	256

Problem 8

Show how the available memory of 1MB will be allocated using Buddy Memory Allocation scheme.
(P1:100K), (P2:240K), (P3:64 K), (P4:256 K), P2 terminates, P1 terminates, (P5:75K),
P3 terminates, P4 terminates, P5 terminates

1 MB

P1 (100)	**Empty**	**Empty**	**Empty**
128	128	256 B	512

P1 (100)	**Empty**	**P2 (240)**	**Empty**
128	128	256 B	512

P1 (100)	**P3 (64)**	**Empty**	**P2 (240)**	**Empty**
128	64	64	256 B	512

P1 (100)	**P3 (64)**	**Empty**	**P2 (240)**	**P4 (256)**	**Empty**
128	64	64	256 B	256	256

P2 terminates

P1 (100)	**P3 (64)**	**Empty**	**Empty**	**P4 (256)**	**Empty**
128	64	64	256 B	256	256

P1 terminates

P5 (75)	**P3 (64)**	**Empty**	**Empty**	**P4 (256)**	**Empty**
128	64	64	256 B	256	256

Empty	**P3 (64)**	**Empty**	**Empty**	**P4 (256)**	**Empty**
128	64	64	256 B	256	256

Problem 9

Consider a swapping system in which memory consists of the following hole sizes in memory order: 10, 4, 20, 18, 7, 9, 12 and 5 KBs. Which hole is taken for successive segment requests of:

a. 12 KB
b. 10 KB
c. 9 KB

- for First Fit
- Repeat the question for Best Fit, Worst Fit and Next Fit

Requests	First Fit	Next Fit	Best Fit	Worst Fit
a. 12	20	20	12	20
b. 10	10	18	10	18
c. 9	18	9	9	12

Problem 10

A swapping system eliminates empty slots by compaction. Assuming a random distribution of many empty slots and many data segments. Time to read or write a 32-bit memory word of is 10 nsec. How long does it take to compact 128 MB? For simplicity, assume that word 0 is part of an empty slot and that the highest word in memory contains valid data.

Time read/ write: 32 bit / 10 nsec = 3.2 bit / nsec
Compact Size = 128 MB
Compact Time = Read + Write = 2 Read / Write

$$= 2 * \frac{128x2^{20}x8 \ \ (bits)}{3.2} = 2 * \frac{128x2^{20}x80}{32} = 2 * 4 * 80 * 1000 * 1000$$

$$= 640 \ \ x \ \ 1000 \ \ x \ \ 1000 \quad nsec = 64 \ \ x \ \ 10^{6} \ \ x \ \ 10^{-9 \ \ secs} = 64x10^{-3 \ \ secs}$$

$$= 0.064 \ \ sec \ \ approximately, because \ \ 2 \ \ power \ \ 10 \ \ is \ \ taken \ \ as \ \ 1000$$

Problem 11

Given five memory partitions of 100, 500, 200, 300 and 600 KB (in order). How would each of the First Fit, Best Fit and Worst Fit algorithms place processes of 212 Kb, 417 Kb, 112 KB, and 426 KB (in order)? Which algorithm makes the most efficient use of memory?

Requests	First Fit	Best Fit	Worst Fit
a. 212	500	300	600
b. 417	600	500	500
c. 112	200	200	300
d. 426	x	600	x

Best fit is most efficient because this scheme has handled all the requests; whereas; first fit & worst fit fails to handle last request of 426.

Problem 12

Consider a logical address space of 16 pages each of 1024 words (each word of 2 Bytes) mapped into a physical memory of 32 frames. Give the Logical and Physical address format. Also give the total Logical and Physical address space. Compute the required page table size for this situation.

p = number of bit to refer 16 pages or $\log_2 16$ = 4 bits
Page size = 1024 words = 2048 bytes = 2^{11} bytes => d = 11 bits
f = number of bits to refer 32 pages = 5 bits
Logical Address = (4, 11)
Physical Address = (5, 11)
PTS = number of entries in page table x PTES
= 2^4x 1 bytes (5 bits required for frames, which is rounded up to 1 byte)
= 16 bytes

Problem 13

A system has 48-bits logical address & a main memory of 64 GBs. Page size is 4096 bytes. Compute the number of pages and frames that exist in the system. Also give L.A & P.A format.

Main Memory = 64 GB = 2_6x2_{30} bytes = 2_{36} bytes

PA = 36 bits
Page size = 4096 bytes = 2^{12} bytes => d = 12 bits
p = LA – d = 48 – 12 = 36 bits
f = PA – d = 36 - 12 = 24 bits
# of pages = 2^{36}
# of frame = 2^{24}
Logical Address = (36, 12)
Physical Address = (24, 12)

Problem 14

Consider a system with logical address 32 bits, Page Size 4K, Main Memory 512MB. Compute the total process address space and maximum number of pages in a process address space. Also give the logical and physical address format. Also give the page table size for this situation.

Main Memory = 512 MB = 29x220 bytes = 229 bytes
PA = 29 bits
Page size = 4 K = 2^{12} bytes => d = 12 bits
p = LA – d = 32 - 12 = 20 bits
f = PA – d = 29 – 12 = 17 bits
# of pages = 2^{20}
# of frame = 2^{17}
Logical Address = (20, 12)
Physical Address = (17, 12)
PTS = # of entries in page table x PTES
= 2^{20} x 3 bytes (f=17 bits required 3 bytes, 16 bits =2 bytes) = 3 MB

Problem 15

Consider a LA space of 8 pages of 1024 words mapped into memory of 32 frames. How many bits are there in the LA? How many bits are there in PA?

p = number of bit to refer 8 pages = 3 bits
Page size = 1024 words = 2048 bytes = 2^{11} bytes => d = 11 bits
f = number of bit to refer 32 frames = 5 bits
LA = p + d = 3 + 11 = 14 bits
PA = f + d = 5 + 11 = 16 bits

Problem 16

In a system with a logical address space of 64 pages, each of 512 bytes mapped into physical memory of 1024 frames. Compute lengths (in bits) of p, d, f, logical and physical address format.

p = number of bit to refer 64 pages = 6 bits
f = number of bit to refer 1024 pages = 10 bits
Page Size = 512 bytes = 2^9 bytes => 9 bits for d
Logical Address = (6, 9)
Physical Address = (10, 9)

Problem 17

A system has 48-bits logical address, physical address space is 32 bits and page size is 4 KB. Determine the lengths of p, d, f, logical and physical address formats, maximum number of pages per process and maximum number of frames in the system, page table entry size (PTES) and the size of the page table

Page Size = 4 kb = 2^{12} bytes => 12 bits for d
p = 48 – 12 = 36
f = 32 – 12 = 20
Logical Address = (36, 12)
Physical Address = (20, 12)
# of pages = 2^{36}
# of frames = 2^{24}

PTES = 20 bits or 3 bytes

PTS = 2^{36} * 3 = 2^{6} * 2^{30} *3 = 64 * 3 GB = 192 GB

Problem 18

Consider a system that allows maximum 2 Mega pages per process with 2 KB page size. Determine the length of the logical address only

Page Size = 2 kb = 2^{11} bytes => 11 bits for d

p = # of bits required to access 2 Mega pages = 21

LA = 21 + 11 = 32 bits

Problem 19

Consider a system with 24-bits physical address space that supports a frame size of 512 Bytes. Calculate the page table entry size (PTES) and the length of physical address.

Frame Size = 512 bytes = 2^{9} bytes => 9 bits for d

f = 24 – 9 = 15 bits

PTES = 15 bits or 2 bytes

PA = 2^{24} bytes = 16 MB

Problem 20

For each of the following logical addresses (given in decimal), compute the page number and offset within the page; if the page size is 4 KB

20000

32768

60000

Page Size = 4 kb = 2^{12} bytes

Logical Address	Page No	Offset
20000	20000 / 2^{12} = 4	20000 - (4096 x 4) = 3616
32768	32768 / 2^{12} = 8	32768 - (4096 x 8) = 0
60000	60000/ 2^{12} = 14	60000 - (4096 x 14) = 2656

Repeat for an 8 KB page

Logical Address	Page No	Offset
20000	20000 / 2^{13} = 2	20000 - (8192 x 2) = 3616
32768	32768 / 2^{13} = 4	32768 - (8192 x 4) = 0
60000	60000/ 2^{13} = 7	60000 - (8192 x 7) = 2656

Problem 21

A machine has a 32-bits address space and an 8 KB page. The page table is entirely in hardware, with one 32-bits word per entry. When a process starts, the page table is copied to the hardware from memory, at one word every 100 nsec. If each process runs for 100 msec (including the time to load the page table), what fraction of the CPU time is devoted to loading the page tables?

Page Size = 8 KB = 2^{13} bytes => 13 bits for d

p = 32 – 13 = 19 bits

PTES = 32 bits = 4 bytes

Copy speed = 4 bytes / 100 nsec = 1 bytes / 25 nsec

PTS = 219 x 4 bytes = 2 MB

Process running time = 100 msec

Time to load PT = 2^{21} bytes x 25 nsec = $\frac{221*25}{100000}$= 50 msec approximately

Problem 22

A machine has a 48 bit virtual addresses and 32 bit physical addresses. Page size is 8 KB. How many entries are needed for the page table?

Page Size = 8 KB = 2^{13} bytes => 13 bits for d

p = 48 – 13 = 35 bits

# of entries in PT = 2^{35}

Problem 23

Let 14000 is a logical address, in which page does it exist if the page size is 1 KB?

Page # = 14000 / 1024 = 13

Problem 24

In a system with 34-bits logical address and 32-bits physical address and a 16 KB page size. How many entries will be there in the page table?

Page Size = 16 KB = 2^{14} bytes => 14 bits for d
p = 34 – 14 = 20 bits
# of entries in PT = 2^{20}

Problem 25

Consider a virtual address 40808. Compute the virtual page number and offset for a 4 KB page

Page # = 40808 / 4096 = 9
Offset = 40808 – (4096 x 9) = 3944

Problem 26

Consider a system with memory access time of 100 nsec. Page table is implemented using associative memory. The TLB access time is 20 ns. Hit ratio is 80%. Calculate the Effective memory access time. Calculate the Effective memory access time if there is no TLB, i.e. the entire page table is kept in memory.

With TLB:

EMA Time = 0.8 x (20 + 100) + 0.2 x (20 + 200 + 100) = 0.8 x 120 + 0.2 x 220 = 94 + 44 = 140 nsec

Without TLB:

EMA Time = 200 nsec (Two times memory access, one time for page table and second time for memory)

Problem 27

Repeat above example with a hit ratio of 95% and compare

With TLB:

EMA Time = 0.95 x (20 + 100) + 0.05 x (20 + 100) = 0.95 x 120 + 0.05 x 220 = 114 + 11 = 125 nsec

Without TLB:

EMA Time = 200 nsec

Problem 28

Consider a paging system with the page table stored in memory.

a. If a memory reference takes 200 nanoseconds, how long does a paged memory reference take?

Paged memory reference takes 400 nsec, 200 nsec for page table and 200 nsec for memory.

b. If we add associative registers, and 75 percent of all page-table references are found in the associative registers, what is the effective memory reference time? (Assume that finding a page-table entry in the associative registers takes zero time, if the entry is there.)

EMA = 0.75 x (200) + 0.25 x (400) = 150 + 100 = 250 nsec (Here time to access associative registers is ignored.

Problem 29

If the hit ratio to a TLB is 80%, and it takes 15 nanoseconds to search the TLB and 150 nanoseconds to access the main memory, then what must be the effective memory access time in nanoseconds?

EMA = 0.8 x (15 + 150) + 0.2 x (15 + 150 + 150) = 0.8 x 165 + 0.2 x 315 = 132 + 63 = 195 nsec

Problem 30

If the hit ratio to register is 30% and hit ratio to TLB is 50%, and it takes 1 and 10 nanoseconds to search the register and the TLB respectively and 150 nanoseconds to access the main memory, then what must be the Effective Memory Access Time in nanoseconds?

EMA = 0.3 x (1 + 150) + 0.5 x (1 + 10 + 150) + 0.2 x (1 + 10 + 300) = 0.3 x 151 + 0.5 x 161 + 0.2 x 311
= 45.3 + 80.5 + 62.2 = 188 nsec

Problem 31

- Consider a system with 80% hit ratio, 50 nsec time to search the associative registers, 750 nsec time to access main memory. Find the time to access a page:

a. When the page number is found in associative memory **EMA = 50 + 750 = 800 nsec**
b. When the page number is not found in associative memory **EMA = 50 + 750 + 750 = 1550 nsec**
c. Find the effective memory access time **EMA = 0.8 x (800) + 0.2 x (1550) = 640 + 310 = 950 nsec**

Problem 32

A computer with a 32-bit address uses a two level page table. Virtual addresses are split into a 9-bit top level page table field, an 11 bit second level page table field and an offset. How large are the pages and how many are there in the address space?

d = 32 – (9 + 11) = 12
Page Size = 2^{12} = 4 KB
# of pages in address space = 2^{9+11} = 2^{20} pages

Problem 33

Suppose that a 32-bit virtual address is broken up into four fields: a, b, c and d. The first three are used for a three level page table system. The fourth field, d, is the offset. Does the number of pages depend on the sizes of all four fields? If not, which one's matter and which ones do not?

# of pages depends on first three fields (a, b & c) and does not depend on d.

Problem 34

A computer has 32-bit logical addresses and 4 KB pages. How many entries are needed in the page table if traditional (one level) paging is used?
How many page table entries are needed for two level paging, with 10 bits in each part?

p = 32 – 12 = 20 bits
Entries in page table = 2^{20}
How many page table entries are needed for two level paging, with 10 bits in each part?
= $2^{10} + 2^{10} \times 2^{10} = 2^{10} + 2^{20}$ entries [2^{10} for outer page table & $2^{10} \times 2^{10}$ in inner page tables]

Problem 35

Consider a system with 20-bit logical address and a page size of8 KB and a **single** level page table. Let the page table base register (PTBR) contains 14000 and PTES is 4 bytes. What will be the starting address of the last entry of the page table? Also determine the maximum number of pages per process supported by the system. What will be the format of the logical address? Can you calculate the number of frames with the provided information?

Page Size = 8 kb = 2^{13} bytes => 13 bits for d
p = 20 – 13 = 7 bits
# of pages 27 = 128 pages
Starting address of last entry = 14000 + 127 x 4 = 14508
# of frames = 2^{32}, because entry size is 4 bytes means 32 bits used to refer pages (assuming no extra bits)

Problem 36

Repeat above problem if logical address is of 32 bit and a **two level page table** is used.

Page Size = 8 kb = 2^{13} bytes => 13 bits for d
p = 32 – 13 = 19 bits
Page Table Size = no of pages x PTES = 2^{19}x4=2^{21} bytes
p1 = 21-13=8
p2 = 13
# of pages = 2^{19} = 512 K pages
Starting address of last entry of outer page = 14000 + 255 x 4 = 14000+1020=15020

Problem 37

Consider a system with 36-bits logical address that supports 4 KB page size. Available physical memory is 64 MB. Calculate p, d, f, PTES, minimum number of levels of the page table per process, format of logical and physical addresses.

Page Size = 4 kb = 2^{12} bytes => 12 bits for d
p = 36 – 12 = 24 bits
Physical memory = 64 MB = $2^{6}\,2^{20}$ bytes
f = 26 – 12 = 14
PTES = 16 bits or 2 bytes (f=14, add two extra bits to round up)
Page Table Size = no of pages x PTES = 2^{24}x2=2^{25} bytes = 32 MB
Number of pages required for page table= $\frac{32MB}{4\ kb} = 8k$
Number of pages required for Inner page table= $\frac{8\ k}{2\ K} = 4$

Outer most page table requires 4 entries.
Each middle level page table requires 2K entries.
Minimum number of levels = 3 level paging required

Problem 38

Consider a system with 36-bits logical address space that supports 4 KB page size. Available physical memory is 64 GB. OS running on this system supports 32 bits PIDs. Calculate p, d, f, PTES in an inverted page table, size of the inverted page table, and the format of logical and physical addresses.

Page Size = 4 kb = 2^{12} bytes => 12 bits for d
p = 36 – 12 = 24 bits
Physical memory = 64 MB = $2^{6}\,2^{20}$ bytes
f = 26 – 12 = 14
PTES = 24 bits (for p) + 1 or 2 bytes for process-id = 4 or 5 bytes (Bytes for process-id is assumed)
Page Table Size = no of frames x PTES = 2^{14}x5 bytes=5*2^{4} KB = 80 KB
Logical Address = (24, 12)
Physical Address = (14, 12)

Problem 39

In a system with 2048 MB RAM and a 4 KB page size, how many entries will be there in an inverted page table?

For inverted page table, we have to find number of frames
Number of entries = Number of frames = $\frac{2048\ MB}{4\ KB} = 512\ \ K\ \ entries = 512 * 1024\ \ entires$

Problem 40

In a 64-bit machine, with 256 MB RAM and a 4 KB page size, how many entries will there be in the page table if it is inverted?

For inverted page table, we have to find number of frames
Number of entries = Number of frames = $\frac{256\ MB}{4\ KB} = 64\ \ K\ \ entries = 64 * 1024\ \ entires$

Problem 41

Consider a system that has a Process ID of 32 bits and uses inverted page table for address translation. The 34-bits logical address is viewed as a 22-bits page identifier (p) and 12 bits offset (d). Size of physical address is 32 bits. Compute page size, PTES, number of pages, and size of inverted page table.

Number of entries = Number of frames = $\frac{2^{32}}{2^{12}} = 2^{20}\ \ entries$
PTES = 22 bits for page number + 4 bytes (32 bit for process id) = 7 bytes (54 bit are rounded up to 56)
Size of inverted page table = $7 * 2^{20}$ **bytes**
Number of pages = 2^{22}

Problem 42: Consider the given segment table, what are the Physical addresses for the following logical addresses? (2,399), (4,0), (4,1000), (3,1300), (6,297)

Consider the given segment table, what are the Physical addresses for the following logical addresses:
(2,399) = 4300 + 399 = 4699
(4,0) = 4700 + 0 = 4700

(4,1000) = Illegal Address, offset is greater than length
(3,1300) = Illegal Address, offset is greater than length
(6,297) = Illegal Address, segment number is greater than total segments

Problem 43

Consider the given segment table, what are the Physical addresses for the following logical addresses? (0,430), (1,10), (2,500), (3,400), (4,112)

(0,430) = 219 + 430 = 649
(1,10) = 2300 + 10 = 2310
(2,500) = Illegal Address, offset is greater than length
(3,400) = 1327 + 400 = 1727
(4,112) = Illegal Address, offset is greater than length

Problem 44

Consider the given segment table, what are the Physical addresses for the following logical addresses? (0,0), (2,120), (6,10), (5,3399), (4,1200), (0,99)

Consider the given segment table, what are the Physical addresses for the following logical addresses:
(0,0) = 12000 + 0 = 12000
(2,120) = 13300 + 120 = 13420
(6,10) = Illegal Address, segment number is greater than total segments
(5,3399) = Illegal Address, offset is greater than length
(4,1200) = 18008 + 1200 = 19208
(0,99) = 12000+ 99 = 12099

Problem 45

Consider the above segment table, if segment table base register (STBR) contains 36500 and segment table entry size (STES) is 64 bits then what will be size of segment table? Also compute the address of the last entry?

Size of segment table: 8 (64 bits) * 6 = 48 bytes
Address of the last entry = 36500 + 8 * 5 = 36500 + 40 = 36540

Problem 46

Compare between Segmentation and Base/Bound Address Translation Techniques

1. Segmentation supports generalized base and bound with support of multiple segments at once
2. Protection: Different segments can have different protections
3. Flexibility: Can separately grow both stack and heap. Enables sharing of code and other segments if needed

Problem 47

What is the main difference between paging of memory and segmentation of memory? List two ways in which this difference affects the address translation hardware required with each scheme.

Problem 48

Consider MULTICS on a GE 345 processor, with Logical Address of 34 bits and page size of 1 KB. **s** is of 18 bits and **d** is of 16 bits.

What is the largest segment size?
What is the maximum number of segments per process?
Give maximum number of pages per segment.
Give the LA format including the no of bits for p and d'

Consider MULTICS on a GE 345 processor, with Logical Address of 34 bits and page size of 1 KB. s is of 18 bits and d is of 16 bits:

– What is the largest segment size?
$2^d = 2^{16}$ = 64 KB
– What is the maximum number of segments per process?
$2^s = 2^{18}$ = 256 K
– Give maximum number of pages per segment:
$2^p = 2^6$ = 64
– Give the LA format including the no of bits for p and d'

(18, 6, 10)   d' = 1 KB = 2^{10} = 10 bits   p = 16 – 10 = 6

Problem 49

Consider a process in MULTICS with its segment # 15 having 5096 bytes. The page size is 1 KB. The process generates a Logical Address of (15, 3921).

Is it a legal address? If yes, why?

How many pages does the segment have?

What page does the logical address refer to, and what is its offset?

What is the P.A if page#3 (i.e. fourth page) is in frame 12?

Yes, it is legal address, because segment number is less than total segments and offset (3921) is less than segment size (5096 bytes)

• How many pages does the segment have?

Segments: 5096 (segment size) / 1024 (page size) = 5096 / 1024 = 5 pages (because 1024 * 5 = 5120)

• What page does the logical address refer to, and what is its offset?

3921 – 1024 x 3 = 3921 – 3072 = 849

Page No: 4 Offset = 849

This means logical address 3921 exist on page 4, with offset 849

• What is the P.A if page#3 (i.e. fourth page) is in frame 12?

Starting address of frame 12 is: 1024 x 11 = 10240 + 1024 = 11264

Physical address = 11264 + 849 = 12113

[Consider Segment Table for next 3 questions]

Problem 50

Consider the given segment table. How many page tables will be constructed for the process whose segments are shown in the segment table?

Number of required page tables = 6 (one-page table for each segment)

Segment #	Length	Base
0	100	12000
1	1200	12100
2	190	13300
3	444	15500
4	19308	18008
5	3400	5000

Problem 51

Consider the given segment table. If the system implements paged-segmentation with the page size of 2 KB, then compute the page number and offset for the logical address of (4, 12765). Also compute the number of pages in segment 4 and the address of the 3rd entry in the page table (Assume PTES is 4 Bytes)

12765 – 2048 x 6 = 12765 – 12288 = 477

Page number = 6 (Address 12765 lies on page 7)

Offset = 477

Number of pages in segment 4 = 1 page because length of segment 4 is smaller than page size

Address of the 3rd Entry = Starting address of page table + size of first two entries = 15500 + 8 = 15508

Problem 52

Consider the above segment table, if segment table base register (STBR) contains 36500 and segment table entry size (STES) is 64 bits then what will be size of segment table? Also compute the address of the last entry?

Segment table size: number of page tables * segment table entry size = 6 * 8 bytes = 48 bytes

Address of last entry: 36500 + 40 = 36540

Problem 53

Consider a logical address in paged segmentation (16, 7865), where 16 is segment number and 7865 is offset. Suppose segment 16 has length of 15690 bytes. Page size is 1 KB. Calculate following:

Number of pages in segment number 16

In which page the above said offset will reside?

How would you represent the above address in <s, p, d'> format?

Let the page p is stored at frame 30, what would be the physical address?

7865 > 7168 (1024 x 7)

Hence offset 7865 lies in page 8 that is page number 7

How would you represent the above address in <s, p, d'> format?

Page Offset= 7865 – 7168 = 697

**Address <16, 7, 697>

Let the page p is stored at frame 30, what would be the physical address?
Starting address of frame 30 = 29 x 1024 = 29696
Physical Address = 29696 + 697 = 30393

Problem 54

Consider a system with Memory Access Time of 100 Nano second. Total Page fault service time is 25 milliseconds. Calculate Effective Access Time. Discuss how much the system will be slowed down if one out of 1000 accesses causes a page fault.

$T_{effective} = (1- p) * 100 + p (25 \times 10^6)$
Given p = 1/1000
$T_{effective} = (1- 1/1000) * 100 + 1/1000 * (25 \times 10^6)$
$T_{effective} = 99.9 + 25000$
$T_{effective} = 25099.9$ nsec = 25000 nsec approximately

So the system is slowed down by a factor of 250 times, by just one-page fault out of 1000 page accesses.

Problem 55:

Consider a system with MAT of 200 nanoseconds. Average Page fault service time is 8 ms. Compute Effective Access Time and discuss how much the system will be slowed down if one out of 1000 accesses causes a page fault. Also compute the page fault rate (p), if we want a slow down by less than 10%

EAT = (No page fault) * mat + (page fault) * (page fault service time)
EAT = 0.999 * 200 NS + 0.001 * 8 MS
$= 0.999 * 200 + 0.001 * 8 * 10^6$
= 199.8 + 8000 = 8199.8 or 8200 NS
Let p = 1/10
$Teffective = (1- p) * 200 + p (8 * 10^6)$
$200 * 1.1 > (1 – p) * 200 + 8000000 * p$
$220 > 200 – 200 * p + 8000000 * p$
$220 - 200 > - 200 * p + 8000000 * p$
$20 > 7999800 * p$
$20 / 7999800 > p$
$p < 1 / 399990$
This means we can allow only one-page fault every 399990 instruction.

Problem 56:

Compute total page faults for the given page Trace for 4 frames using following algorithms:

- FIFO
- OPTIMAL
- LRU

Page Trace/Reference String is: 2, 3, 4, 5, 6, 4, 5, 2, 7, 8, 9, 4, 9, 0, 8, 9, 1, 6, 5, 6, 5, 3

Total page faults = 14 (FIFO is used as default, when any page can be replaced)

FIFO	2	3	4	5	6	4	5	2	7	8	9	4	9	8	9	1	6	5	6	5	3
	2	2	2	2	6	6	6	6	6	6	9	9	9	9	9	9	9	5	5	5	5
		3	3	3	3	3	3	2	2	2	2	4	4	4	4	4	4	4	4	4	3
			4	4	4	4	4	4	7	7	7	7	7	7	7	1	1	1	1	1	1
				5	5	5	5	5	5	8	8	8	8	8	8	8	6	6	6	6	6
Page Faults	1	2	3	4	5	5	5	6	7	8	9	10	10	10	10	11	12	13	13	13	14

Total page faults = 11 (Last row in green color is showing page faults)

OPTIMA	2	3	4	5	6	4	5	2	7	8	9	4	9	8	9	1	6	5	6	5	3
	2	2	2	2	2	2	2	2	7	8	8	8	8	8	8	1	1	5	5	5	3
		3	3	3	6	6	6	6	6	6	6	6	6	6	6	6	6	6	6	6	6
			4	4	4	4	4	4	4	4	4	4	4	4	4	4	4	4	4	4	4
				5	5	5	5	5	5	5	9	9	9	9	9	9	9	9	9	9	9
Page Faults	1	2	3	4	5	5	5	5	6	7	8	8	8	8	8	9	9	10	10	10	11

Total page faults = 14 (Last row in green color is showing page faults for 3 frames)

LRU	2	3	4	5	6	4	5	2	7	8	9	4	9	8	9	1	6	5	6	5	3
	2	2	2	2	6	6	6	6	7	7	7	7	7	7	7	1	1	1	1	1	1
		3	3	3	3	3	3	2	2	2	2	4	4	4	4	4	6	6	6	6	6
			4	4	4	4	4	4	4	8	8	8	8	8	8	8	8	5	5	5	5
				5	5	5	5	5	5	5	9	9	9	9	9	9	9	9	9	9	3
Page Faults	1	2	3	4	5	5	5	6	7	8	9	10	10	10	10	11	12	13	13	13	14

Problem 57 & 58 are similar.

Problem 59 & 60 are excluded from course.

Problem 61: Consider a system with 64 frames. At a particular instant of time there are three processes P1(10 pages), P2(40 pages) and P3(127 pages). Allocate frames to processes using Fixed and proportionate distribution?

Frame allocation using fixed: Allocate 21 frames to each of the three processes i.e. 64/3 = 21 (Integer Division)
Frame allocation using proportion:
Total pages = 10 + 40 + 127 = 177 pages
Frame allocation to P1 = 64 / 177 x 10 = 4
Frame allocation to P2 = 64 / 177 x 40 = 14
Frame allocation to P3 = 64 / 177 x 127 = 46