Logical Address to Physical Address

Single Level

In a system there are 3 processes- P1 (10 bytes), P2 (6 bytes) and P3 (8 bytes) with page size of 2 bytes. The size of the main memory is 32 bytes. Page tables of processes are given below.

Page Tables					
Process P1		Process P2		Process P3	
p	f	p	f	p	f
0	7	0	15	0	2
1	12	1	13	1	6
2	1	2	9	2	3
3	5			3	8
4	4				

Find corresponding physical addresses of the following logical addresses:
a. Address 01001 of P1.
b. Address 00100 of P1.
c. Address 00111 of P1.
d. Address 00001 of P2.
e. Address 00100 of P2.
f. Address 10101 of P2.
g. Address 00010 of P3.
h. Address 11011 of P3.
i. Address 00110 of P3.

Answer:

The address is 5 bits, so the address space has 2^5 addresses.

Page size is 2, so offset within each page is 2^1.

Logical Address	Page No. (first 4 bits)	Offset (last 1 bit)	Frame No. (from page table)	Physical Address
01001 (P1)	0100 (4)	1	4	4×2+1 = **9 (01001)**
00100 (P1)	0010 (2)	0	1	1×2+0 = **2 (00010)**
00111 (P1)	0011 (3)	1	5	5×2+1 = **11 (01011)**
00001 (P2)	0000 (0)	1	15	15×2+1 = **31 (11111)**
00100 (P2)	0010 (2)	0	9	9×2+0 = **18 (10010)**
10101 (P2)	1010 (10)	1	–	**Invalid Page No.**
00010 (P3)	0001 (1)	0	6	6×2+0 = **12 (01100)**
11011 (P3)	1101 (13)	1	–	**Invalid Page No.**
00110 (P3)	0011 (3)	0	8	8×2+0 = **16 (10000)**

Logical Address to Physical Address

Single Level - With valid/invalid bit

A process runs in a system with single level paging and it has a logical address space of 8 bits. In the system page size is 16 Bytes and size of the main memory is 512 Bytes. Page table of the process is given below:

PMT		
Page #	Frame #	valid/invalid bit
0	5	v
1		i
2	12	v
3	3	v
4	25	v
5		i
6	0	v
7	18	v
8		i
9	31	v
10	7	v
11		i
12	21	v
13	9	v
14		i
15	14	v

When the CPU is executing the process it generates the following logical addresses: 3, 90, 167 and 241. Map the corresponding physical addresses of these logical addresses.

Answer:

Logical Address (dec)	**Logical Address (8-bit)**	**Page No. (first 4 bits)**	**Offset (last 4 bits)**	**Frame No. (PMT)**	**Physical Address (calc)**	**Physical Address (binary, 9-bit)**
3	00000011	0000 (0)	0011 (3)	5	5×16 + 3 = **83**	**001010011** (83)
90	01011010	0101 (5)	1010 (10)	—	**Invalid**	—
167	10100111	1010 (10)	0111 (7)	7	7×16 + 7 = **119**	**001110111** (119)
241	11110001	1111 (15)	0001 (1)	14	14×16 + 1 = **225**	**011100001** (225)

Logical Address to Physical Address

Multilevel Paging - Two Levels

A process runs in a system with multi level paging and it has a logical address space of 8 bits. In the system page size is 16 Bytes, size of each entry of the page table is 4 Bytes and size of the main memory is 512 Bytes. In order to fit the pages of the process in the main memory the OS applies a two-level paging technique in outer page number bits of the logical address space until the outer most page table can be allocated in a frame of the main memory.

I. Illustrate the logical address space of the process including the necessary outer page bits, inner page bits and offset bits of every step with proper mathematical calculations during the paging mechanism of the system described above.
II. In this system, if the CPU generates logical addresses 179 and 90 then map the corresponding physical addresses of these logical addresses.

Necessary page table information is given below:

Outer page table		
Page #	Frame #	valid/invalid bit
0		i
1	6	v
2	11	v
3		i

Inner page tables					
PMT at frame 6			PMT at frame 11		
Page #	Frame #	valid/invalid bit	Page #	Frame #	valid/invalid bit
0	3	v	0	25	v
1	20	v	1		i
2		i	2	12	v
3	7	v	3	14	v

Answer:

I.
Logical address space = 8 bits
Number of logical addresses = 2^8
Page size = 16 bytes = 2^4

Page no.	Offset
4 bits	4 bits

Size of each page table entry = 4 bytes = 2^2

So, size of the total page table = $2^4 * 2^2 = 2^6$

Since the size of the page table is larger than the page size (2^4), it means that we need to use two-level paging.

Number of pages of the page table = $2^6 / 2^4 = 2^2$

So, the outer page table will have 2^2 entries, and 2 bits will correspond to the outer page bits.

Outer page no.	Inner page no.	Offset
2 bits	2 bits	4 bits

II.

Logical (dec)	Logical (8-bit)	Outer (2 bits)	Inner (2 bits)	Offset (4 bits)	Frame (from inner PT)	Physical (dec) = frame×16 +offset	Physical (9-bit)
179	10110011	10 (2)	11 (3)	0011 (3)	outer 2 → inner PT at frame 11 → inner 3 → frame **14**	14×16 + 3 = **227**	**011100011**
90	01011010	01 (1)	01 (1)	1010 (10)	outer 1 → inner PT at frame 6 → inner 1 → frame **20**	20×16 + 10 = **330**	**101001010**

Effective Access Time (with TLB)

Effective Access Time (EAT) = $\alpha \times (\epsilon + t) + (1 - \alpha) \times (\epsilon + 2t)$

Question
During TLB search, the associative lookup time (ε) is 3ns and hit ratio (α) is 70%. For each time memory access, 80ns is needed. Calculate the effective access time.

Answer:
Given:

- $\epsilon = 3$ ns
- $\alpha = 70\% = 0.7$
- $t = 80$ ns

Substitute:

$EAT = 0.7 \times (3 + 80) + 0.3 \times (3 + 2 \times 80)$

$EAT = 0.7 \times 83 + 0.3 \times 163$

$EAT = 58.1 + 48.9 = 107$ ns

Answer: 107 ns

Question
During TLB search, the associative lookup time (ε) is 5ns and hit ratio (α) is 65.5%. For each time memory access, 200ns is needed. Calculate the effective access time.

Answer:
Given:

- $\epsilon = 5$ ns
- $\alpha = 65.5\% = 0.655$
- $t = 200$ ns

Substitute:

$EAT = 0.655 \times (5 + 200) + (1 - 0.655) \times (5 + 2 \times 200)$

$EAT = 0.655 \times 205 + 0.345 \times 405$

$EAT = 134.275 + 139.725 = 274$ ns

Answer: 274 ns

Frame Replacement

In a system, there are 4 frames in the main memory. In a particular scenario main memory needs to accommodate 16 pages according to the order of the given reference string.

[1 5 4 3 0 4 7 1 2 9 1 2 7 3 1 7]

Apply FIFO and LRU page replacement algorithms in order to accommodate the pages in the main memory and find out page hit ratio and page fault ratio of every algorithm. Lastly, logically explain which algorithm performs better for the given scenario.

Answer:

FIFO algorithm

Time	Ref	Frame 1	Frame 2	Frame 3	Frame 4	Result
1	1	1*	-	-	-	Fault
2	5	1*	5	-	-	Fault
3	4	1*	5	4	-	Fault
4	3	1*	5	4	3	Fault
5	0	0	5*	4	3	Fault
6	4	0	5*	4	3	Hit
7	7	0	7	4*	3	Fault
8	1	0	7	1	3*	Fault
9	2	0*	7	1	2	Fault
10	9	9	7*	1	2	Fault
11	1	9	7*	1	2	Hit
12	2	9	7*	1	2	Hit
13	7	9	7*	1	2	Hit
14	3	9	3	1*	2	Fault
15	1	9	3	1*	2	Hit
16	7	9	3	7	2*	Fault

FIFO Totals:
Hits = 5, Faults = 11
Hit Ratio = 31.25%, Fault Ratio = 68.75%

LRU Algorithm

Time	Ref	Frame 1	Frame 2	Frame 3	Frame 4	Result
1	1	1 (0) *	-	-	-	Fault
2	5	1 (1) *	5 (0)	-	-	Fault
3	4	1 (2) *	5 (1)	4 (0)	-	Fault
4	3	1 (3) *	5 (2)	4 (1)	3 (0)	Fault
5	0	0 (0)	5 (3) *	4 (2)	3 (1)	Fault
6	4	0 (1)	5 (4) *	4 (0)	3 (2)	Hit
7	7	0 (2)	7 (0)	4 (1)	3 (3) *	Fault
8	1	0 (3) *	7 (1)	4 (2)	1 (0)	Fault
9	2	2 (0)	7 (2)	4 (3)*	1 (1)	Fault
10	9	2 (1)	7 (3) *	9 (0)	1 (2)	Fault
11	1	2 (2)	7 (4) *	9 (1)	1 (0)	Hit
12	2	2 (0)	7 (5) *	9 (2)	1 (1)	Hit
13	7	2 (1)	7 (0)	9 (3) *	1 (2)	Hit
14	3	2 (2)	7 (1)	3 (0)	1 (3) *	Fault
15	1	2 (3) *	7 (2)	3 (1)	1 (0)	Hit
16	7	2 (4) *	7 (0)	3 (2)	1 (1)	Hit

LRU Totals:
Hits = 6, Faults = 10
Hit Ratio = 37.5%, Fault Ratio = 62.5%

So, the hit ratio is better in LRU.