1. (a) time to access the page table (200) + time to make the memory reference (200)

= 400 ns

(b) 200\*75% + 400\*25% = 250ns

(c) 200\*99.5% + 400\*0.5% = 219ns

1. (a) 32-10 = 22bits

(b) since the offset is 10bits, so the lower 10 bits are 111111 which is 127

(c) the last 10 bits 11 1111 1111 is 2047

(d) address is 0000 0000 0000 0000 0001 0000 0000 0000

(e) address is 0000 0000 0000 0000 0001 0011 1111 1111

1. (a) and 64K need bit so 44-16 = 28, thus the answer is

(b) since 4 bytes per page, thus one page frame fit 64K/4 = 16000, so we have

44-16(first offset)-14(second offset)=14 bits for page number.(offset field has 16 bits length)

(c) 4G/64K = and pages used by the second level is 4, also the pages used by the first level page tale is 4/2^16\*1, thus the total page frames used is + 4 + 1 = 65541

LRU:

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| 8 | 7 | 2 | 9 | 3 | 1 | 2 | 5 | 8 | 3 | 3 | 4 | 6 | 7 | 1 | 0 | 5 | 7 | 4 | 6 | 2 | 3 |
| 7 | 2 | 9 | 3 | 1 | 2 | 5 | 8 | 3 | 4 | 4 | 6 | 7 | 1 | 0 | 5 | 7 | 4 | 6 | 2 | 3 | 0 |
| 2 | 9 | 3 | 1 | 2 | 5 | 8 | 3 | 4 | 6 | 6 | 7 | 1 | 0 | 5 | 7 | 4 | 6 | 2 | 3 | 0 | 1 |
| 9 | 3 | 1 | 2 | 5 | 8 | 3 | 4 | 6 | 7 | 7 | 1 | 0 | 5 | 7 | 4 | 6 | 2 | 3 | 0 | 1 | 9 |

18 page-faults

FIFO:

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| 8 | 7 | 2 | 2 | 9 | 3 | 3 | 1 | 5 | 8 | 8 | 4 | 6 | 7 | 7 | 1 | 0 | 5 | 4 | 6 | 2 | 3 |  |  |  |
| 7 | 2 | 9 | 9 | 3 | 1 | 1 | 5 | 8 | 4 | 4 | 6 | 7 | 1 | 1 | 0 | 5 | 4 | 6 | 2 | 3 | 0 |  |  |  |
| 2 | 9 | 3 | 3 | 1 | 5 | 5 | 8 | 4 | 6 | 6 | 7 | 1 | 0 | 0 | 5 | 4 | 6 | 2 | 3 | 0 | 1 |  |  |  |
| 9 | 3 | 1 | 1 | 5 | 8 | 8 | 4 | 6 | 7 | 7 | 1 | 0 | 5 | 5 | 4 | 6 | 2 | 3 | 0 | 1 | 9 |  |  |  |

21 page-faults

Optimal:

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| 8 | 8 | 8 | 8 | 8 | 8 | 8 | 4 | 4 | 4 | 4 | 4 | 4 | 4 | 4 | 4 | 6 | 2 | 3 | 3 | 1 | 9 |
| 7 | 7 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 2 | 2 | 2 | 2 | 5 | 5 | 5 | 5 | 5 | 5 | 5 | 5 | 5 | 5 | 5 | 5 | 5 | 5 | 5 | 5 | 5 | 5 |
| 9 | 3 | 3 | 3 | 3 | 3 | 3 | 3 | 6 | 7 | 7 | 7 | 7 | 7 | 7 | 7 | 7 | 7 | 7 | 7 | 7 | 7 |

16 page-faults

1. Diagram

   Description automatically generated
2. TLB miss with no page fault

Virtual address -> TLB lookup -> miss -> page table walk -> page in memory -> update TLB

1. TLB miss with page fault

Virtual address -> TLB lookup -> miss -> page table walk -> page not in memory -> page fault (OS loads page)

1. TLB hit with no page fault

Virtual address -> TLB lookup -> hit -> protection check -> permitted -> physical address (to cache)

1. TLB hit with page fault

Virtual address -> TLB lookup -> hit -> protection check -> denied -> protection fault