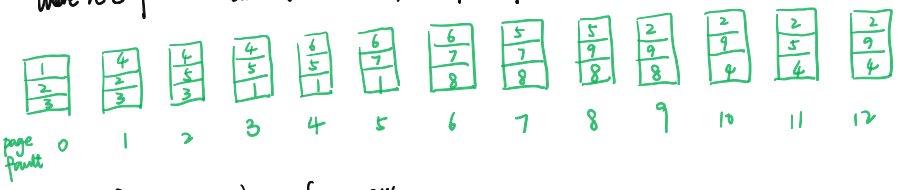
19.IS.

(a)

following Seq. of page refs.

1, 2, 3, 4, 5, 1, 5, 1, 6, 7, 8, 5, 8, 9, 2, 4, 5, 4, 2, 9

there're 3 frames. Illustrate the contents of the frames under LRU page replacement algo. Step by step



page fault: 0 1 2 3 4 5 6 7 8 9 10 11 12

(b) consider following snapshot of a sys.

Allocation.

	A	B	C	D	E
P1	2	3	0	0	2
P2	1	0	3	2	0
P3	0	2	1	3	2
P4	4	3	0	3	3
P5	1	1	3	0	1

Max Request

	A	B	C	D	E
P1	5	3	2	1	6
P2	1	4	9	2	0
P3	0	2	8	6	3
P4	6	10	0	3	3
P5	4	3	5	0	1

(i) what's the contents of the matrix Need denoting the # of additional resources needed by each process?

Max Request − Allocation

	A	B	C	D	E	
P1	3	0	2	1	4	O
P2	0	4	6	0	0	X
P3	0	0	7	3	1	X
P4	2	7	0	0	0	O
P5	3	2	2	0	0	X

(ii) available resources are

A	B	C	D	E
3	4	3	1	1

is the sys. in a safe state? if so, list All the posible safe sequences

Yes, it's a safe state.

1) P5 → 45612 → P2 → 55932 → P3 → 51064 → { P1 → P4 ; P4 → P1 }

(c) Consider a singe-level paging scheme with TLB used to store parte of the page table. Each memory access time is 200×10^{-9} s, TLB access time is 20×10^{-9} s and TLB hit ratio is 0.8. What's the effective memo access time.

$$EAT = (m+e)\alpha + (2m+e)(1-\alpha)$$
$$= [0.8(220) + 0.2(420)] \times 10^{-9}$$
$$= (176 + 84) \times 10^{-9}$$
$$= 260 \times 10^{-9} \text{ s}$$

IS 18.

Q2.

(a) page table: 256 entries.

physical address: 32 bits

2^{16} frames in the physical memory.

(a.1) size of the physical memory?

2^{32}

(a.2) How many bits in the logical address?

8 + 16 = 24

8 ↑ page number, 16 ↖ offset

(a.3) what's the physical address of virtual address 0xD3E2F7?

D3 → 13 × 8 + 3 = 107. (Given) → 0x~~92AE~~ EAC3

13 × 16 + 3 = 211

104 page, ~~92AE~~ EAC3 frame

∴ physical address:

0x~~92AE~~ EAC3 E2F7.

(a.4). physical: 0xA89C78C2, virtual?

A89C → 111 → ~~104+7~~ 96+15

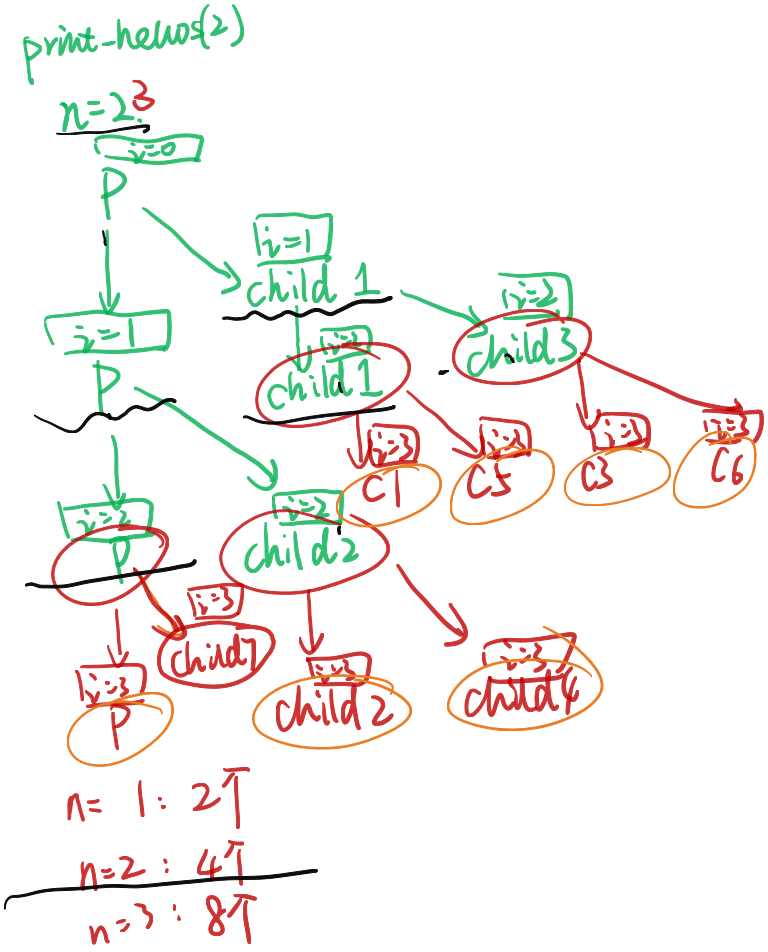
111 page, ~~D7~~ 6F

∴ virtual address:

0x~~D7~~ 6F 78C2

b) consider following C program, How many lines of output does the func. print_hellos print?

```
void print_hellos (int n){
    int i;
    for (i=0; i<n; i++){
        fork();
    }
    printf("hello\n");
    exit(0);
}
int main(){
    int n = 10;
    print_hellos(n);
    return 0;
}
```

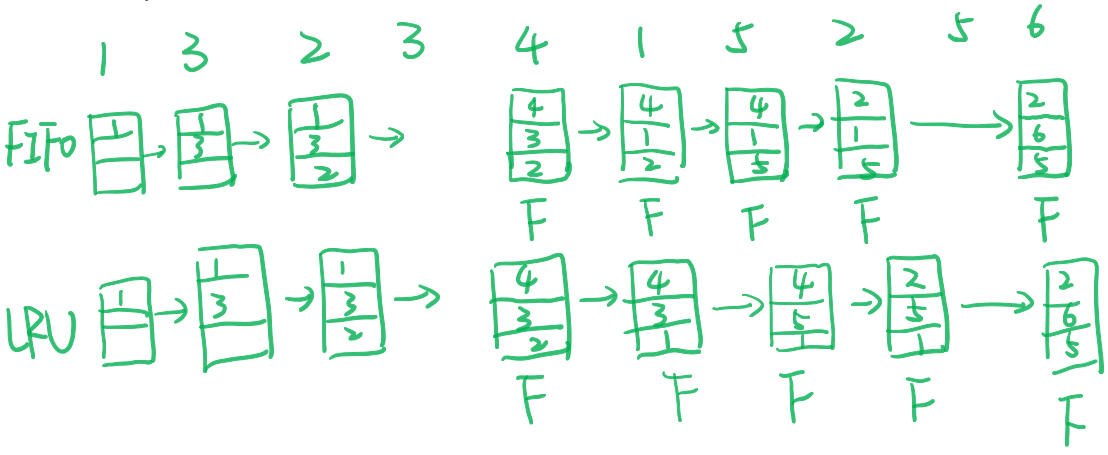


2^{10}

(c) page ref seq:

1, 3, 2, 3, 4, 1, 5, 2, 5, 6

3 frames show FIFO & LRU



IS17

Q2 Producer-Consumer Problem.

Semaphore "full" → 0.
"empty" → N (buffer size)
"mutex" → 1.

Producer Process:

```
do{
    produce an item np;
    wait (empty);
    wait (mutex);
    add np to buffer;
    signal (mutex);
(2) signal (empty);   signal (full)
} while (true);
```

```
wait:
  s.count --.
  if (s.count <0) {
      该进程进入 s.queue 队列
  }
```

```
signal:
  s.count ++.
  if s.count <=0 {
      进入就绪队列.
  }
```

Consumer Process:

```
do{
    wait (full);
    wait (mutex);
    remove an item from buffer to nc;
    signal (mutex)
    signal (empty)
    consume nc;
} while (true);
```

(1) Explain 3 semaphores:

mutex: make sure only one process is modifying the buffer.

empty: number of empty spaces in the buffer (make sure the producer won't add item when full)

full: number of items in the buffer (make sure consumer won't remove from empty store)

(2) There's one error above, point out & correct.

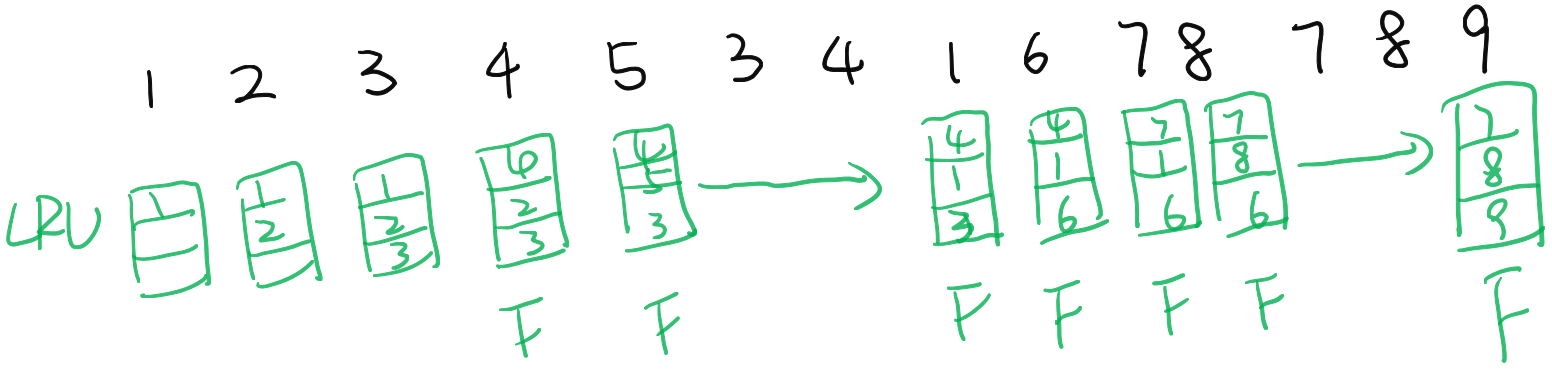
(b) What is the cause of thrashing? How does the sys. detect thrashing?

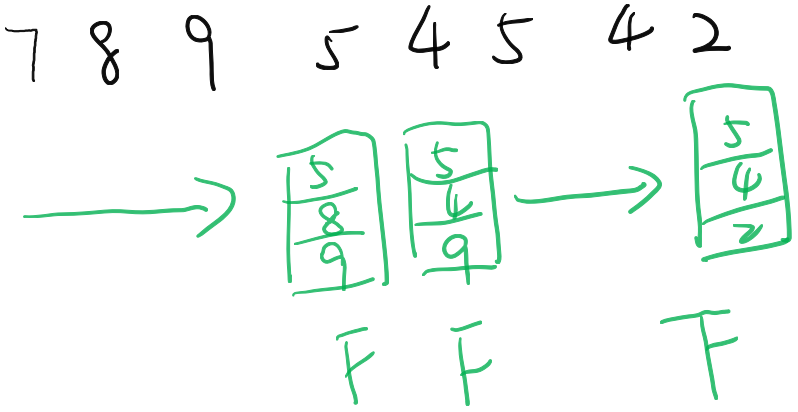
The system spends most of its time swapping process pieces rather than executing instructions because the main memory is almost full and the sys. has to retreive the data it has just throwed out.

detect: Too many page faults?

evaluate the level of CPU utilization.

(c) page refs:





# of page faults:

10

IS 2016.

Q3

(a). same as 19(b)

(b) Consider following sys:

(1) Byte - addressable Memory

(2) 24 - bit logical address

(3) Max seg. length 2M bytes

(4) Page size 128 bytes.

Using segmentation with one-level paging scheme. the logical address can be partitioned into 3 parts. show its.

(3) $2M \rightarrow 2^{21}$

∴ Max seg number : 2^3

(4) $128 \rightarrow 2^7$

∴ Max page number : $2^{21} \div 2^7 = 2^{14}$

∴ offset : 2^7

∴ logical address:

3 bits	14 bits	7 bits
seg #	page #	offset

|——— 24 bits in total ———|

(c) Given logical address <page, offset>, compute corresponding physical address

(page: decimal)

size of page and frame are 1024 bytes.

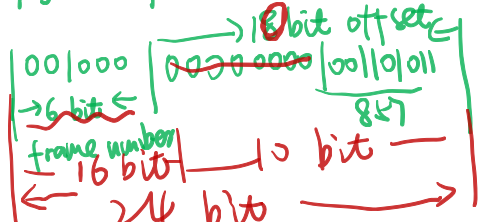
byte addressable.

(1) <0, 85>

(2) <5, 989>

(1) page 0 → frame 8. (Given.) offset : 1024 bytes = 2^{10} bytes.

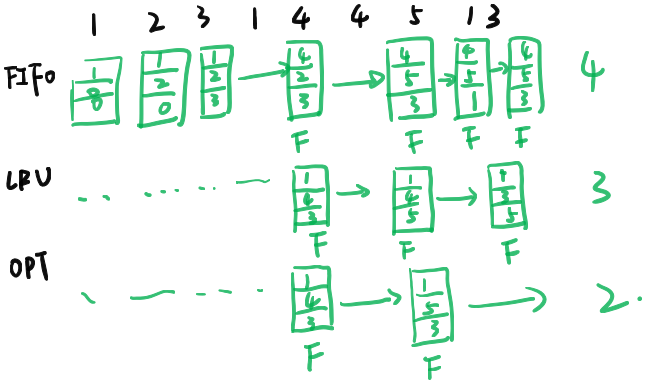
~~2^{18} bits.~~



(2) 5 → 22

IS 15

Q3 (a). Page Ref.



(b) Memory access time: 200×10^{-9} s

TLB access time. 10×10^{-9} s

hit ratio: 0.9

effective memory access time:

$$0.9 \times (10+200) \times 10^{-9}\text{s} + 0.1 \times (10+400) \times 10^{-9}\text{ s}$$

$$= (210 \times 0.9 + 410 \times 0.1) \times 10^{-9}\text{s}$$

$$= 230 \times 10^{-9}\text{ s}.$$

(c) List 2 advantages of seg. and paging sys. over a pure seg. scheme.

① less fragments, use the memory more effectively

② easier to manage the memory with fixed length of data

☆ page fault + 前面 table 为空的部分.