FINAL 22/12/2023

E) 1) Cuantas 'a' imprime el siguiente programa si se llama con:

a) Se llama con ./a.out ./a.out ./a.out , El programa p0

b) Se llama el programa p1 con ./a.out; luego con ./a.out 1; luego con ./a.out 1 2;

2) Con un esquema (9,9,9,12) a -> (44, 12) Teniendo tablas de 512 entradas:
a) Calcular la cantidad maxima de memoria que pueden ocupar las page tables segun se use:
- Solo la tabla L0
- Las tablas L0 y L1
- L0, L1, y L2

b) Implementar un buffer circular rapido de 4KiB en la direccion virtual 0x2000 mapeando dos pags iguales, una al lado de la otra, es decir que 0x2000 y 0x3000 esten mapeadas a la misma pagina. Modificar las tablas de paginas para lograr esto.

Ej 3) El siguiente codigo assembly es un multiprograma Concurrent Vector Writting de INTS (Cada elemento es de 4bytes) en legv8. Usa dos componentes paralelas p0 y p1, cada instruccion en assembler es atomica. El resultado es en el vector(o array?) que esta en la direccion 0x1000:

a) Describa los resultados posibles si p0 y p1 son DOS PROCESOS.
b) Describa los resultados posibles si p0 y p1 son DOS HILOS.
c) Describa los resultados posibles si p0 y p1 son DOS HILOS, UN SOLO NUCLEO, y el context switch NO GUARDA los registros de los hilos.

```
        0x1000 = {2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2} 14 dos

P0:                                 P1:
    movz x0, #0x1000    // Int* elementos       movz x0, #0x1000    // Int* elementos
    movz x1, #0x0010    // N                    movz x1, #0x0010    // N
    mov x2, x0          // x2 ptr al actual     mov x2, x0          // x2 ptr al actual
    ld x1, x1, #2       // N * 4, sizeof(int)   ld x1, x1, #2       // N * 4, sizeof(int)
    add x0, x0, x1      // x0 es ptr limite     add x0, x0, x1      // x0 es ptr limite
    mov x1, #0          // Volver a escribir    mov x1, #1          // Volver a escribir

L1:                                 L1:
    sturw x1, [x2]                      sturw x1, [x2]
    add x2, x2, #4                      add x2, x2, #4
    cmp x2, x0                          cmp x2, x0
    bne L1                              bne L1
    ret                                 ret
```

Ej 4) Completar el cuadro de create/foo/bar y explicar brevemente porque hace cada cosa

FILE SYSTEM IMPLEMENTATION 13

	data bitmap	inode bitmap	root inode	foo inode	bar inode	root data	foo data	bar data [0]	bar data [1]	bar data [2]
			read							
				read		read				
create (/foo/bar)		read write					read			
								write		
				read write						
			write							
write()	read write					read				
								write		
				write						
write()	read write					read				
									write	
				write						
write()	read write					read				
										write
				write						

Ej 5) Se borro el d-bitmap, pero la inode table de 8 elementos y el bitmap correspondientes estan intactos. En un esquema bien simple con block-size de 512 bytes y 8 bloques directos, recuperar el d-bitmap

INODE TABLE		ADDRESS INDIRECT							
INODE	SIZE	0	1	2	3	4	5	6	7
0	1024	10	7	6	0	0	0	9	0
1	2	0	0	0	0	0	0	0	0
2	2	11	13	17	19	23	24	31	37
3	3583	12	14	15	16	18	20	21	22
4	1024	10	7	6	0	0	0	0	0
5	2	9	5	6	7	0	0	0	0
6	666	12	12	12	12	12	12	12	12
7	1	3	40	41	42	0	0	0	0

i-bitmap:

0	1	1	1	1	0	0