Sistemas Operativos, Parcial 1, martes 18 octubre 2022

Apellidos:_____ **Nombres:**_____

Ejercicio 1

El siguiente código de máquina y su desensamblado RISC-V **computa la suma prefijo en el mismo arreglo** (*in-place prefix sum*). El arreglo `a` está en el segmento ELF `.bss` y empieza en `0x2FC0` y termina en `0x3008` **exclusive**. Como sus elementos son `unsigned long`, cada uno ocupa 8 bytes y por lo tanto tiene 9 elementos.

```
0000000000000634 <main>:
 634: 0613                   li    a2,0x3008   # <__BSS_END__> &a[9]
 636: b206                   li    a5,0x2FC8   # <a+0x8> &a[1]
 638: 6398                   ld    a4,0(a5)    # a4 = a[i]
 63a: ff87b683               ld    a3,-8(a5)   # a3 = a[i-1]
 63e: 9736                   add   a4,a4,a3
 640: e398                   sd    a4,0(a5)    # a[i] = a4
 642: 07a1                   addi  a5,a5,8     # "i++"
 644: fec79ae3               bne   a5,a2,0x638 # <main+0x10>, "i<9"
 648: 8082                   ret
```

trap F F F F F trap

Escribir la **traza de memoria** completa que genera la ejecución del proceso **incluyendo los *instruction fetch***.

634 636

Inicialización

0613, b206,

Vuelta 1

638 6398, 2FC8, FF87B683, 2FC0, 9736, e398, 2FC8, 07A1, FEC79AE3, 63a 63e 640 642 644

Vuelta 2

638 6398, 2FC9, FF87B683, 2FC8, 9736, e398, 2FC9, 07A1, FEC79AE3, 2FD0 63a 2FC8 63e 640 2FD0 642 644

Vuelta 3

638 6398, 2FCA, FF87B683, 2FC0, 9736, e398, 2FCA, 07A1, FEC79AE3, 2FD8 63a 2FD0 63e 640 2FD8 642 644

Vuelta 4

638 6398, 2FCB, FF87B683, 2FCA, 9736, e398, 2FCB, 07A1, FEC79AE3, 2FE0 63a 2FD8 63e 640 2FE0 642 644

Vuelta 5

638 6398, 2FCC, FF87B683, 2FCB, 9736, e398, 2FCC, 07A1, FEC79AE3, 2FE8 63a 2FE0 63e 640 2FE8 642 644

Vuelta 6

638 6398, 2FCD, FF87B683, 2FCC, 9736, e398, 2FCD, 07A1, FEC79AE3, 2FF0 63a 2FE8 63e 640 2FF0 642 644

Vuelta 7

638 6398, 2FCE, FF87B683, 2FCD, 9736, e398, 2FCE, 07A1, FEC79AE3, 2FF8 63a 2FF0 63e 640 2FF8 642 644

Vuelta 8

638 6398, 2FCF, FF87B683, 2FCE, 9736, e398, 2FCF, 07A1, FEC79AE3, 3000 63a 2FF8 63e 640 3000 642 644

Fin

8082.

648

Ejercicio 2

Supongamos que en `trampoline.S`, la rutina en ensamblador RISC-V que guarda los registros de espacio de usuario se comete un pequeño error por culpa del gato 🙀. La parte que los restituye está perfecta.

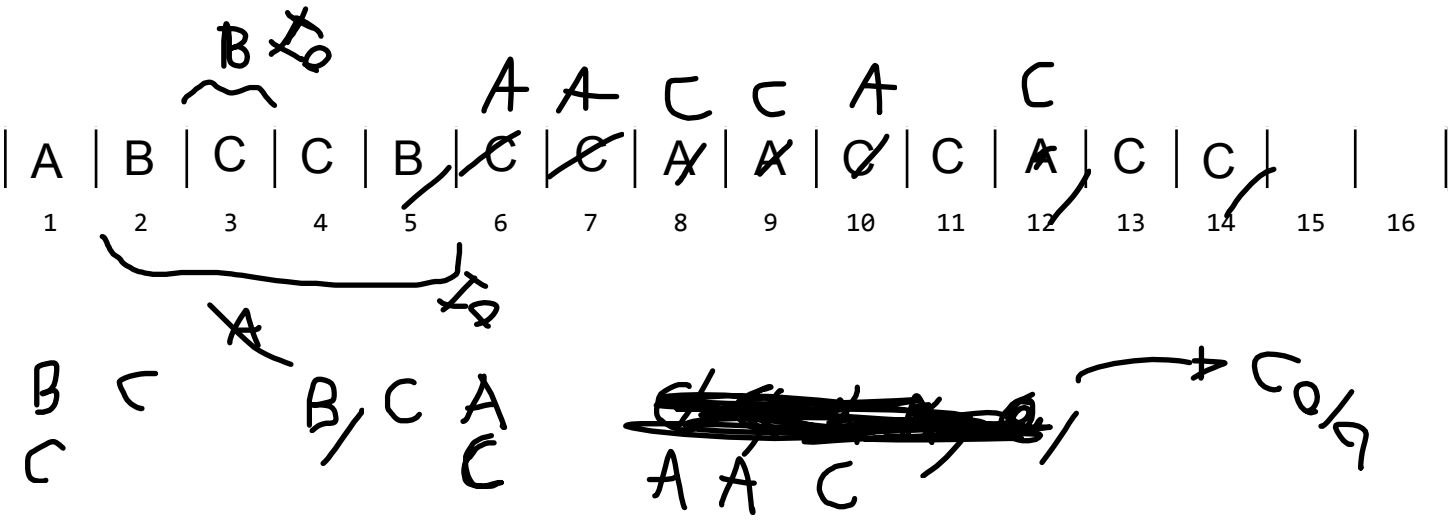
```
sd a1, 120(a0)                  ld a1, 120(a0)
sd a2, 128(a0)                  ld a2, 128(a0)
sd a3, 136(a0)                  ld a3, 136(a0)
sd a4, 144(a0)                  ld a4, 144(a0)
sd a2, 152(a0) #ERROR!          ld a5, 152(a0)
sd a6, 160(a0)                  ld a6, 160(a0)
sd a7, 168(a0)                  ld a7, 168(a0)
```

Indicar en el código de máquina del Ejercicio 1, **ENTRE** qué líneas se puede producir un TRAP sin cambiar el funcionamiento del programa (poner "**TRAP**") y entre qué líneas ese TRAP resulta fatal para la ejecución de nuestro código (poner "**F**"). Recalcamos, poner TRAP ó F entre cada una de las líneas de código indicando si se puede producir un TRAP de manera inocua o no.

Ejercicio 3

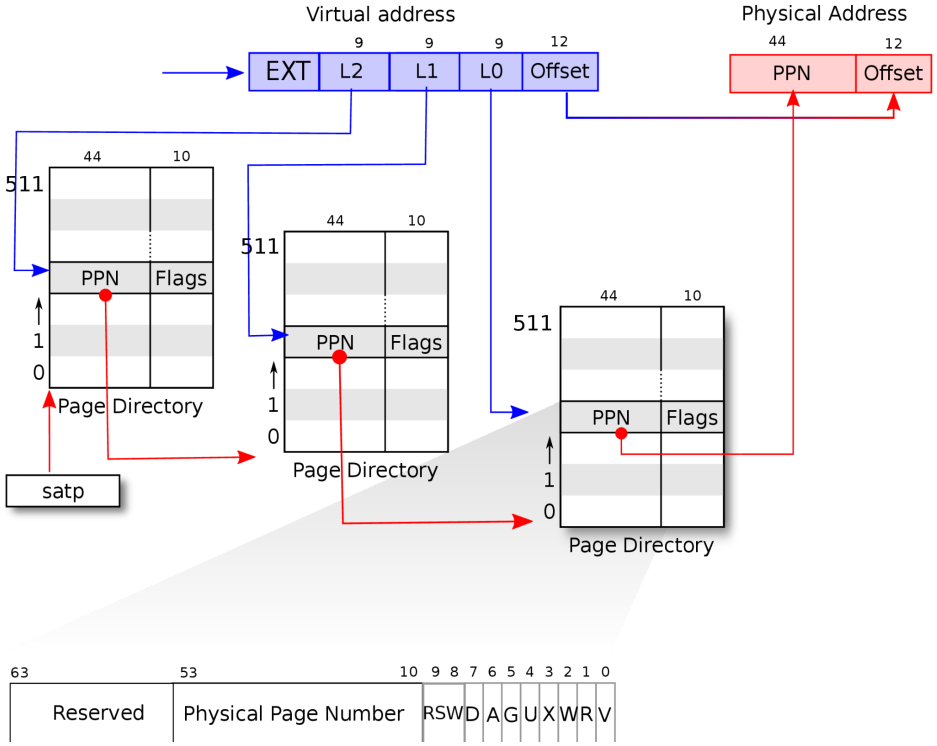
Planificar con Round Robin Q=2 para los siguientes procesos que tienen mezcla entre cómputo CPU y espera IO. Ante situaciones de simultaneidad, ordenar alfabéticamente, por ejemplo ¿Cuál de los tres procesos inicia en tiempo 0?: el "A".

Proceso	Inicio	CPU	IO	CPU
A	0	1	4	3
B	0	1	1	1
C	0	8		



Ejercicio 4

Tenemos un esquema de paginación RISC-V con páginas de 4 KiB de 3 niveles con formato 9,9,9,12 -> 44,12 como muestra la figura.



Bits de control

V: *válido*

R: se puede leer, *readable*

W: se puede escribir, *writable*

X: se puede ejecutar, *executable*

Supongamos que tenemos el registro de paginación apuntando al marco físico `satp=0x00000000FE0`.

`0x00000000FE0`
`------------`
`0x1FF: 0x00000000000, ----`
⋮
`0x004: 0x00000000000, ----`
`0x003: 0x00000000000, ----`
`0x002: 0x00000000FEA, XWRV`
`0x001: 0x00000000FEA, XWRV`
`0x000: 0x00000000FEA, XWRV`
	`0x00000000FEA`
	`------------`
	`0x1FF: 0x00000000000, ----`
	⋮
	`0x004: 0x00000000000, ----`
	`0x003: 0x00000000000, ----`
	`0x002: 0x000000AD0BE, XWRV`
	`0x001: 0x000000AD0BE, XWRV`
	`0x000: 0x000000AD0BE, XWRV`
		`0x000000AD0BE`
		`------------`
		`0x1FF: 0x00000000000, ----`
		⋮
		`0x004: 0x00000000000, ----`
		`0x003: 0x00000D1AB10, XWR-`
		`0x002: 0x00000DECADA, -WRV`
		`0x001: 0x000CAFECAFE, ----`
		`0x000: 0x0000000ABAD, X--V`

a) Traducir de **virtual a física** las direcciones:

`0x0000` 0x 000 0000 0000 0000 0000 0000 0000 0000 0000 0000 PD1 = 0 → (0000000FEA) PD2 = 0 → (00000AD0BE) PD3 = 0 → (000000ABAD) OFFSET = 0 → 000000ABAD000

`0x1000` 0x1000 = 0x 000 0000 0000 0000 0000 0000 0001 0000 0000 0000 PD1 = 0 → (0000000FEA) PD2 = 0 → (00000AD0BE) PD3 = 1 → (00CAFECAFE) INVALIDA

`0x2000` 0x2000 = 0x 000 0000 0000 0000 0000 0000 0010 0000 0000 0000 PD1 = 0 → (0000000FEA) PD2 = 0 → (00000AD0BE) PD3 = 2 → (0000DECADA) OFFSET = 0 - > 0000DECADA000

`0x3000` 0x3000 = 0x 000 0000 0000 0000 0000 0000 0011 0000 0000 0000 PD1 = 0 → (0000000FEA) PD2 = 0 → (00000AD0BE) PD3 = 3 → (0000D1AB10) INVALIDA

---

b) Traducir de la dirección física `0xDECADA980` a **todas** las virtuales que la apuntan.

0000DECADA980

0x80402980
0x80202980
0x80002980

0x40402980
0x40202980
0x40002980

0x402980
0x202980
0x2080

Ejercicio 5

A la luz del esquema de paginación del Ejercicio 4, indicar de manera esquemática que es lo que pasaría con la traza de memoria respecto a la ejecución de un proceso corriendo el código de máquina del Ejercicio 1.

Inicialización

ABAD634 , ABAD636 ,

Vuelta 1

ABAD638 , DECADA4040 , ABAD63a , DECADA4030 , ABAD63e , ABAD640 , DECADA4040 , ABAD642 , ABAD644 ,

Vuelta 2

ABAD638 , DECADA4048 , ABAD63a , DECADA4040 , ABAD63e , ABAD640 , DECADA4048 , ABAD642 , ABAD644 ,

Vuelta 3

ABAD638 , DECADA4050 , ABAD63a , DECADA4048 , ABAD63e , ABAD640 , DECADA4050 , ABAD642 , ABAD644 ,

Vuelta 4

ABAD638 , DECADA4058 , ABAD63a , DECADA4050 , ABAD63e , ABAD640 , DECADA4058 , ABAD642 , ABAD644 ,

Vuelta 5

ABAD638 , DECADA4060 , ABAD63a , DECADA4058 , ABAD63e , ABAD640 , DECADA4060 , ABAD642 , ABAD644 ,

Vuelta 6

ABAD638 , DECADA4068 , ABAD63a , DECADA4060 , ABAD63e , ABAD640 , DECADA4068 , ABAD642 , ABAD644 ,

Vuelta 7

ABAD638 , DECADA4070 , ABAD63a , DECADA4068 , ABAD63e , ABAD640 , DECADA4070 , ABAD642 , ABAD644 ,

Vuelta 8

ABAD638 , DECADA4078 , ABAD63a , DECADA4070 , ABAD63e , ABAD640 , DECADA4078 , ABAD642 , ABAD644 ,

Fin

ABAD648 .