FINAL 25/7

Te recomiendo que le preguntes a wolo cualquier cosa que no entiendas, el chabon le mete onda. En el 1 x ej el loco nos tiro que era fflush y sleep. Un copado.

EJERCICIO1

Para el siguiente programa que usa las syscalls de UNIX:

| int main (int argc, char\*\* argv){  if(fork( ) > 0){  putchar(‘a’); fflush(NULL);  sleep(4);  }  if(fork( ) > 0){  putchar(‘b’); fflush(NULL);  sleep(2);  }  if(fork( ) > 0){  putchar(‘c’); fflush(NULL);  sleep(1);  }  return 0;  } |
| --- |

1. Indique la cantidad de ocurrencias de cada letra cuando el proceso y todos sus hijos terminan.

| Letra | a | b | c |
| --- | --- | --- | --- |
| Apariciones |  |  |  |

1. Muestre todos los resultados finales posibles como secuencias de letras abc.

EJERCICIO 2

Para el siguiente programa en RISC-V, del cual damos el codigo fuente para comprenderlo mejor, escribir la traza de memoria virtual completa incluyendo instrucciones y datos.

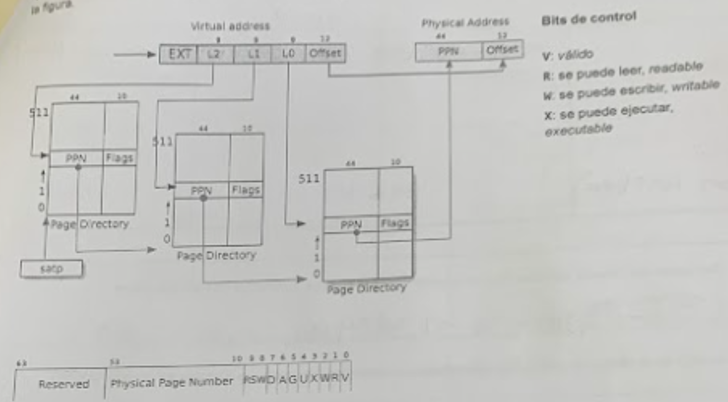
(Subraye esta parte porque en la clase de consulta para este final uno de los errores comunes fue no escribir la direccion de memoria de la instruccion a ejecutar. Me parecio cute que esta vuelta haya hecho hincapié desde el enunciado como resolverlo).

| Código fuente | Desensamblado RISC-V |
| --- | --- |
| unsigned long s[8] =  {2, 3, 5, 7, 11, 13, 17, 19};  unsigned long idx[8] =  {5, 6, 3, 4, 1, 2, 7, 0};  unsigned long d[8];  int main(int c, char \*\*v)  {  unsigned long n = 8;  for (unsigned long i = 0; i < n; i++)  d[idx[i]] = s[i];  } | 000000000000000c <main>:  c: 00000697 lla a4, 0x1000 # idx  10: 000068693 lla a3, 0x2000 # s  14: 8536 mv a0, a3  16: 00000597 lla a1, 0x3000 # d  000000000000001a <.L2>:  1a: 631c ld a5, 0(a4)  1c: 078e slli a5, a5, 0x3  1e: 97ae add a5, a5, a1  20: 6298 ld a2, 0(a3)  22: e390 sd a2, 0(a5)  24: 0721 addi a4, a4, 8  26: 06a1 addi a3, a3, 8  28: fea719e3 bne a4, a0, 1a <.L2>  2c: 8082 ret |
| Datos | |
| 1000:0x00000005  1008:0x00000006  1010:0x00000003  1018:0x00000004  1020:0x00000001  1028:0x00000002  1030:0x00000007  1038:0x00000000 | 2000: 0x00000002  2008:0x00000003  2010:0x00000005  2018:0x00000007  2020:0x0000000b  2028:0x0000000d  2030:0x00000011  2038:0x0000013 |

EJERCICIO 3

Tenemos un esquema de paginacion RISC-V con paginas de 4KiB de 3 niveles con formato (9,9,9,12)->(44,12) como muestra la figura.

Perdon x la calidad ijijiijj



Supongamos que tenemos el registro de paginacion apuntando al marco fisico

satp = 0x00000000FE0.

| 0x00000000FE0  —---------  0x1FF: 0x00000000, ----  .  .  .  0x004: 0x00000000, ----  0x003: 0x000000000, —-  0x002: 0x0000000FEA, XWR-  0x001: 0x0000000FEA, XWRV  0x000: 0x0000000FEA, XWR- | 0x00000000FEA  —-----------  0x1FF: 0x00000000, ----  .  .  .  0x004: 0x00000000, ----  0x003: 0x00000000, ----  0x002: 0x000000AD0BE, XWR-  0x001: 0x000000AD0BE, XWRV  0x000: 0x000000AD0BE, XWR- | 0x000000AD0BE  —----------  0x1FF: 0x00000000, ----  .  .  .  0x004: 0x00000000, ----  0x003: 0x0000001DA1B0, X-R-  0x002: 0x0000000ECADA, -WRV  0x001: 0x0000CAFECAFE, ----  0x000: 0x0000000ABAD, X--V |
| --- | --- | --- |

1. Calcular la cantidad min y max de RAM que pueden ocupar las tablas de pagina segun el esquema (9,9,9,12)->(44,12).

MIN: \_\_\_\_ KiB Max:\_\_\_\_\_GiB

1. Hacer el mapeo inverso de la direccion fisica 0xABAD666 a todas sus direcciones virtuales.
2. Hacer el mapeo inverso de la direccion fisica 0x0000CAFECAFE a todas sus direcciones virtuales.

EJERCICIO 4

Para el siguiente multiprograma de 32 componentes suponer atomicidad linea a linea.

| s = 0; | | | |
| --- | --- | --- | --- |
| while (32>s){  s = s + 1;  } | while (32>s){  s = s + 1;  } | …32 copias total… | while (32>s){  s = s + 1;  } |
| s = ? | | | |

1. Dar todos los valores posibles de s al terminar el multiprograma, ordenados de menor a mayor.
2. Agregar en el codigo de arriba solo semaforos y su inicializacion, para que siempre de el valor MINIMO.

EJERCICIO 5

Una desarrolladora de FS propone una idea nueva: Chained Indrect Blocks (CIB)

Tenemos indices de bloque de 32 bits, bloques de 4KiB, 8 bloques directos y 1 indirecto, peeero, el ultimo indice del bloque indirecto apunta a un nuevo bloque indirecto y asi sustantivamente.

1. Calcular el tamaño maximo de archivo en MiB (lol)
2. Calcular la sobrecarga que se produce (tamañon ocupado por los bloques indirectos) para un archivo de 32 MiB.

EJERCICIO 6

Los siguientes fragmentos de codigo corresponden a un sistema de archivos tipo UNIX.

| #define NDIRECT 12  #define NINDIRECT (BSIZE / sizeof(uint))  #define MAXFILE (NDIRECT + NINDIRECT)  // On-disk inode structure  struct dinode {  short type; // File type  short major; // Major device number (T\_DEVICE only)  short minor; // Minor device number (T\_DEVICE only)  short nlink; // Number of links to inode in file system  uint size; // Size of file (bytes)  uint addrs[NDIRECT+1];// Data block addresses  }; | // Truncate inode (discard contents).  // Caller must hold ip->lock.  void  itrunc(struct inode \*ip)  {  int i, j;  struct buf \*bp;  uint \*a;  for(i = 0; i < NDIRECT; i++){  if(ip->addrs[i]){  bfree(ip->dev, ip->addrs[i]);  ip->addrs[i] = 0;  }  }  if(ip->addrs[NDIRECT]){  bp = bread(ip->dev, ip->addrs[NDIRECT]);  a = (uint\*)bp->data;  for(j = 0; j < NINDIRECT; j++){  if(a[j])  bfree(ip->dev, a[j]);  }  brelse(bp);  bfree(ip->dev, ip->addrs[NDIRECT]);  ip->addrs[NDIRECT] = 0;  }  ip->size = 0;  iupdate(ip);  } |
| --- | --- |

Calcular el tamaño maximo de un archivo en KiB.