Sistemas Operativos (75.08): Lab Kernel

Lavandeira, Lucas (#98042)
lucaslavandeira@gmail.com

Rozanec, Matias (#97404)
rozanecm@gmail.com

22.jun.2018

Índice

Parte I
Resolución

1. Creación de stacks en el kernel

1.1. Ej: kern2-stack

Explicar: ¿qué significa "estar alineado"?
Una variable o array de variables está alineada en memoria a X bits cuando su posición inicial es un múltiplo de X. Por ejemplo estar alineado a 4 bytes significa que su dirección de memoria es un múltiplo de 4 (0, 4, 8, ...). Esto por lo general se hace para tomar ventaja de como el hardware maneja posiciones de memoria (la lectura de posiciones alineadas al tamaño de la palabra de la arquitectura suele ser más eficiente).

Alinear una variable por lo general significa agregar suficientes bytes con valor nulo hasta que quede la posición incial de la misma en donde la queramos.

Mostrar la sintaxis de C/GCC para alinear a 32 bits el arreglo kstack anterior.
GCC provee el **atributo** `aligned` para especificar a cuantos bytes (no bits) se quiere alinear alguna variable. Para alinear `kstack` a 32 bits, se declararía de la siguiente manera:

```
unsigned char __attribute__ ((aligned (4))) kstack[8192];
```

¿A qué valor se está inicializando kstack? ¿Varía entre la versión C y la versión ASM? (Leer la documentación de as sobre la directiva .space.)
Según la directiva `.space` de x86, `.space 8192` toma los próximos 8192 bytes y los inicializa a cero. Declarar el arreglo en C de la manera que lo hicimos no inicializa los contenidos, y deja basura en el stack. Esto es aceptable puesto que el stack se lee solo a través del stack pointer que siempre apunta a la dirección con el último valor escrito.

Explicar la diferencia entre las directivas .align y .p2align de as, y mostrar cómo alinear el stack del kernel a 4 KiB usando cada una de ellas.

2. Parte 2

3. Parte 3