O'Hallaron: CS:APP, 2ª Ed.

Signatura: ESIIT/C.1 BRY com

Capítulo 3: Representación de Programas a nivel de máquina

Problemas Prácticos:

3.30-3.34, pp.257-58, 262, 265-66

3.30. El siguiente fragmento de código ocurre con frecuencia en la versión compilada de rutinas de librería:

```
1        call next
2   next:
3        popl %eax
```

a. ¿Qué valor resulta asignado al registro `%eax`?
b. Explicar por qué no hay una instrucción `ret` correspondiente a esta `call`.
c. ¿Qué propósito o utilidad tiene este fragmento de código?

3.31. La siguiente secuencia de código ocurre justo cerca del inicio del código ensamblador generado por GCC para un procedimiento C:

```
1        subl   $12, %esp
2        movl   %ebx, (%esp)
3        movl   %esi, 4(%esp)
4        movl   %edi, 8(%esp)
5        movl   8(%ebp), %ebx
6        movl   12(%ebp), %edi
7        movl   (%ebx), %esi
8        movl   (%edi), %eax
9        movl   16(%ebp), %edx
10       movl   (%edx), %ecx
```

Vemos que sólo tres registros (`%ebx`, `%esi`, y `%edi`) se salvan en la pila (líneas 2-4). El programa modifica éstos y otros tres registros (`%eax`, `%ecx`, y `%edx`). Al final del procedimiento, los valores de los registros `%edi`, `%esi`, y `%ebx` son restaurados (no se muestra), mientras que los otros tres se dejan en su estado modificado.
Explicar esta aparente inconsistencia en el salvado y restauración del estado de los registros.

3.32. Una función C `fun` tiene el siguiente código de cuerpo:

```
*p = d;
return x-c;
```

El código IA32 implementando este cuerpo es como sigue:

```
1        movsbl  12(%ebp), %edx
```

```
2          movl    16(%ebp), %eax
3          movl    %edx, (%eax)
4          movswl  8(%ebp), %eax
5          movl    20(%ebp), %edx
6          subl    %eax, %edx
7          movl    %edx, %eax
```

Escribir un prototipo para la función `fun`, mostrando los tipos y orden de los argumentos `p`, `d`, `x`, y `c`.

3.33. Dada la función C

```
1    int proc(void)
4    {
5        int x, y;
6        scanf("%x %x", &y, &x);
9        return x-y;
10   }
```

GCC genera el siguiente código ensamblador:

```
1     proc:
2        pushl  %ebp
3        movl   %esp, %ebp
4        subl   $40, %esp
5        leal   -4(%ebp), %eax
6        movl   %eax, 8(%esp)
7        leal   -8(%ebp), %eax
8        movl   %eax, 4(%esp)
9        movl   $.LC0, (%esp)
10       call   scanf
      Dibujar marco de pila en este punto
11       movl   -4(%ebp), %eax
12       subl   -8(%ebp), %eax
13       leave
14       ret
```

Asumir que el procedimiento `proc` empieza a ejecutarse con los siguientes valores de registros:

Registro	Valor
%esp	0x800040
%ebp	0x800060

Suponer que `proc` llama a `scanf` (línea 10), y que `scanf` lee los valores `0x46` y `0x53` de la entrada estándar. Asumir que el string "`%x %x`" está almacenado en la posición de memoria `0x300070`.

A. ¿A qué valor queda ajustado `%ebp` en la línea 3?
B. ¿A qué valor queda ajustado `%esp` en la línea 4?
C. ¿En qué direcciones están almacenadas las variables locales `x` e `y`?

D. Dibujar un diagrama del marco de pila para `proc` justo después de que retorne `scanf`. Incluir tanta información como se pueda sobre las direcciones y contenidos de los elementos del marco de pila.
E. Indicar las regiones del marco de pila que no son utilizadas por `proc`.

3.34. Para una función C que tiene la estructura general

```
int rfun(unsigned x) {
    if ( ________ )
       return ________;
   unsigned nx = ________;
    int rv = rfun(nx);
    return ________;
}
```

GCC genera el siguiente código ensamblador (omitiendo ajuste y destrucción de marco de pila):

```
1        movl   8(%ebp), %ebx
2        movl    $0, %eax
3        testl   %ebx, %ebx
4        je  .L3
5        movl  %ebx, %eax
6        shrl    %eax
7        movl   %eax, (%esp)
8        call   rfun
9        movl  %ebx, %edx
10       andl  $1, %edx
11       leal   (%edx,%eax), %eax
12     .L3:
```

A. ¿Qué valor almacena `rfun` en el registro salva-invocado `%ebx`?
B. Rellenar las expresiones que faltan en el código C mostrado arriba.
C. Describir en castellano qué función calcula este código.