**Respuesta a los ejercicios**

**Kayla Aguila Super**

* **Solución al Problema 5.4**

1. En el código optimizado, el registro %xmm0 se utiliza simplemente como un valor temporal, tanto establecido como utilizado en cada iteración del bucle. En el código más optimizado, se utiliza más a la manera de la variable "acc" en "combine4", acumulando el producto de los elementos del vector. La diferencia con "combine4", sin embargo, es que la ubicación "dest" se actualiza en cada iteración por la segunda instrucción "vmovsd". Podemos ver que esta versión optimizada funciona de manera similar al siguiente código en C:

1 /\* Asegurarse de que "dest" se actualice en cada iteración \*/

2 void combine3w(vec\_ptr v, data\_t \*dest)

3 {

4 long i;

5 long length = vec\_length(v);

6 data\_t \*data = get\_vec\_start(v);

7 data\_t acc = IDENT;

8

9 /\* Inicializar si la longitud es <= 0 \*/

10 \*dest = acc;

11

12 for (i = 0; i < length; i++) {

13 acc = acc OP data[i];

14 \*dest = acc;

15 }

16 }

1. Las dos versiones de "combine3" tendrán una funcionalidad idéntica, incluso con alias de memoria.
2. Esta transformación se puede hacer sin cambiar el comportamiento del programa, porque, con la excepción de la primera iteración, el valor leído desde "dest" al comienzo de cada iteración será el mismo valor escrito en este registro al final de la iteración anterior. Por lo tanto, la instrucción de combinación puede simplemente usar el valor ya en %xmm0 al comienzo del bucle.

* **Solución al Problema 5.5**

1. La función realiza 2n multiplicaciones y n adiciones.
2. Podemos ver que el cálculo que limita el rendimiento aquí es el cálculo repetido de la expresión "xpwr = x \* xpwr". Esto requiere una multiplicación en punto flotante (5 ciclos de reloj), y el cálculo para una iteración no puede comenzar hasta que el de la iteración anterior haya finalizado. La actualización del resultado solo requiere una suma en punto flotante (3 ciclos de reloj) entre iteraciones sucesivas.

* **Solución al Problema 7.1**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Símbolo | ¿Entrada en .symtab? | Tipo de símbolo | Módulo donde se define | Sección |
| buf | sí | externo | m.o | .data |
| bufp0 | sí | global | swap.o | .data |
| bufp1 | sí | global | swap.o | COMMON |
| swap | sí | global | swap.o | .text |
| temp | no | - | - | - |

* **Solución al Problema 7.2**

1. El enlazador elige el símbolo fuerte definido en el módulo 1 sobre el símbolo débil definido en el módulo 2 (regla 2):

(a) REF(main.1)→DEF(main.1)

(b) REF(main.2)→DEF(main.1)

1. Esto es un error, porque cada módulo define un símbolo fuerte "main" (regla 1).
2. El enlazador elige el símbolo fuerte definido en el módulo 2 sobre el símbolo débil definido en el módulo 1 (regla 2):

(a) REF(x.1)→DEF(x.2)

(b) REF(x.2)→DEF(x.2)

* **Solución al Problema 7.3**

linux> gcc p.o libx.a

linux> gcc p.o libx.a liby.a

linux> gcc p.o libx.a liby.a libx.a

En la primera línea, el programa "p.o" se enlaza con la biblioteca estática "libx.a" y se genera un archivo ejecutable. En la segunda línea, el programa "p.o" se enlaza con las bibliotecas estáticas "libx.a" y "liby.a", y se genera un archivo ejecutable. En la tercera línea, el programa "p.o" se enlaza con las bibliotecas estáticas "libx.a", "liby.a" y "libx.a", en ese orden, y se genera un archivo ejecutable.