Daliana Cabrera Grau

Respuesta al problema 7.1 del libro:

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Symbol | .symtab entry? | Symbol type | Module where defined | Section |
| buf | Sí | extern | m.o | .data |
| bufp0 | Sí | global | swap.o | .data |
| bufp1 | Sí | global | swap.o | COMMON |
| swap | Sí | global | swap.o | .text |
| temp | No | - | - | - |

El símbolo buf es un símbolo externo, declarado e inicializado en el módulo m.o, por tanto, se localiza en la sección .data.

El símbolo bufp0 es un símbolo global declarado e inicializado en el módulo swap.o, por tanto, se localiza en la sección .data.

El símbolo bufp1 es un símbolo global declarado, pero no inicializado, en el módulo swap.o, por tanto, se localiza en la sección COMMON.

El símbolo swap es un símbolo global, específicamente una función, con declaración y cuerpo en el módulo swap.o, por tanto, se localiza en la sección .text.

El símbolo temp es un símbolo local, por tanto, no tiene una entrada en la tabla de símbolos.

Respuesta al problema 7.2 del libro:

A. El enlazador escoge el símbolo fuerte definido en el módulo 1, sobre el símbolo débil definido en el módulo 2.

(a) REF(main.1) → DEF( main.1)

(b) REF(main.2) → DEF( main.1)

B. Aquí se produce un error porque en los dos módulos main está definido como un símbolo fuerte.

C. El enlazador escoge el símbolo fuerte definido en el módulo 2, sobre el símbolo débil definido en el módulo 1.

(a) REF(x.1) → DEF(x . 2)

(b) REF(x.2) → DEF( x . 2)

Respuesta al problema 7.3 del libro:

1. En este escenario, solo se tiene una dependencia directa, donde "p.o" depende de "libx.a". La solución más simple sería ejecutar:

gcc p.o libx.a

1. En este escenario, "p.o" depende de "libx.a" y "libx.a" depende de "liby.a". Se necesita proporcionar ambas bibliotecas estáticas al enlazador. El siguiente comando resolverá todas las dependencias:

gcc p.o libx.a liby.a

1. En este escenario, se tiene una dependencia circular entre "libx.a" y "liby.a". Esto puede crear problemas de resolución de dependencias. Para resolverlo, se necesita proporcionar ambas bibliotecas en el comando de enlace. La solución se puede lograr ejecutando el siguiente comando:

gcc p.o libx.a liby.a libx.a

Respuesta al problema 5.4:

A. En el código menos optimizado, el registro %xmm0 se utiliza como un valor temporal que se establece y se usa en cada iteración del bucle. En el código más optimizado, se utiliza de manera similar a una variable llamada "acc" en la función "combine4", acumulando el producto de los elementos del vector. Sin embargo, en cada iteración, la ubicación "dest" se actualiza mediante la segunda instrucción "vmovsd".

La versión optimizada funciona de manera similar al siguiente código en C:

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

long i;

long length = vec\_length(v);

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

data\_t acc = IDENT;

\*dest = acc;

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

acc = acc OP data[i];

\*dest = acc;

}

}

B. Las dos versiones de "combine3" tienen la misma funcionalidad, incluso si hay solapamiento de memoria.

C. Esta optimización se puede hacer sin cambiar el comportamiento del programa, ya que, excepto en la primera iteración, el valor leído de "dest" al comienzo de cada iteración será el mismo valor que se escribió en el registro anterior al final de la iteración anterior. Por lo tanto, la instrucción de combinación puede utilizar simplemente el valor que ya se encuentra en %xmm0 al comenzar el bucle.

Respuestas al problema 5.5:

A. La función realiza n sumas y 2n multiplicaciones.

B. El cálculo que limita el rendimiento en este caso es la repetición del cálculo de la expresión xpwr = x \* xpwr, lo cual implica una multiplicación de punto flotante (5 ciclos de reloj) y el cálculo de una iteración no puede comenzar hasta que el cálculo de la iteración anterior haya finalizado. La actualización del resultado solo requiere una suma de punto flotante (3 ciclos de reloj) entre iteraciones sucesivas.