EC - Práctica 2

Javier Gálvez Obispo

9 de diciembre de 2017

1. Sesión de depuración Suma_01_S_cdec

4. ¿Qué modos de direccionamiento usa la instrucción **add (%ebx, %edx,4), %eax**? ¿Cómo se llama cada componente del primer nodo? El último componente se llama escala. ¿Qué sucedería si lo eliminásemos?

Utiliza direccionamiento indexado y direccionamiento directo.
offset(base, index, scale)
M[offset + R[base] + R[index] * scale]
El programa deja de funcionar correctamente.

7. La instrucción **jne** en el programa original se podría cambiar por alguno de entre otros tres saltos condicionales (uno de ellos es sencillamente un mnemotécnico para el mismo código de operación) y el programa seguiría funcionando igual. ¿Cuáles son esos 3 mnemotécnicos? ¿Qué tendría que suceder para que se notaran diferencias en el original

Los otros 3 saltos que se pueden utilizar son: jnz, jg, ja.
La variable resultado tendría otro valor al acabar el programa.

2. Sesión de depuración Suma_01_S_libC

1. ¿Qué error se obtiene si no se añade **-lc** al comando de enlazar ¿Qué tipo de error es? (en tiempo de ensamblado, enlazado, ejecución...)

suma_02_S_libC.o: En la función '_start':
/home/javi/DGIIM/EC/practica2/suma_02_S_libC.s:26: referencia a 'printf' sin definir
/home/javi/DGIIM/EC/practica2/suma_02_S_libC.s:30: referencia a 'exit' sin definir

Es un fallo de enlazado.

2. ¿Qué error se obtiene si no se añade la especifación del enlazador dinámico al comando de enlazar? ¿Y si se indica como enlazador un fichero inexistente? ¿Qué tipo de error es? (en tiempo de ensamblado, enlazado, ejecución...)

$: ld -m elf_i386 suma_02_S_libC.o -o suma_02_S_libC -lc /lib/ld-linux.so.2
$: ./suma_02_S_libC
bash: ./suma_02_S_libC: No existe el archivo o el directorio

Crea el ejecutable pero no deja ejecutarlo, es un fallo de ejecución.

$: ld -m elf_i386 suma_02_S_libC.o -o suma_02_S_libC -lc -dynamic-linker /lib/ld-linux.so.3
$: ./suma_02_S_libC

bash: ./suma_02_S_libC: No existe el archivo o el directorio

Crea el ejecutable pero no deja ejecutarlo, es un fallo de ejecución.

3. Popcount.c

8. Gráfica comparación de optimizaciones de gcc para popcount.c

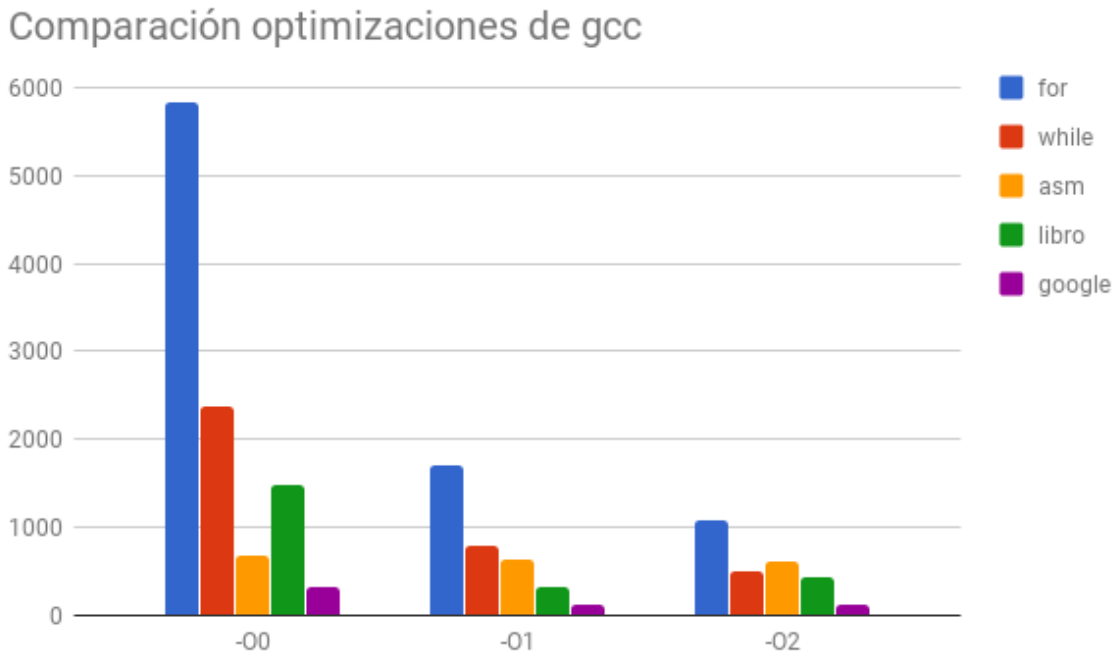

Figura 1: Gráfica rendimientos popcount.

4. Parity.c

1.Diseñar la fórmula sugerida en el cuarto párrafo de la sección 4.2. ¿Cómo se ha razonado ese cálculo?

El código se puede ver en el archivo parity.c.
Siguiendo los consejos que nos dá el guión.

2. ¿Por qué necesitamos declarar la lista de enteros como **unsigned** ¿Qué problema habría si se declara como int? ¿Notaríamos en nuestro programa la diferencia? En caso negativo... ¿qué tendría que suceder para notar la diferencia?

Si se declara int en vez de unsigned el bit más significativo sería el del signo y nuestro programa dejaría de funcionar correctamente.

3. En la 3ª versión (aplica máscara al final) se dejó pendiente comparar la 2ª para ver si la mejora es tan importante ¿lo es? ¿Para todos los niveles de optimización?

Como se puede ver en la gráfica (Figura 2) la tercera versión (libro, color amarillo) es más rápida, para todas las optimizaciones, que la segunda versión (while, color rojo).

4. La 4ª versión (paso a ASM de la 3ª) probablemente sea la versión .ªextraña"de este ejemplo, porque no sea mejor que la anterior (incluso usando restricciones a registros) y/o porque tarde lo mismo independientemente del nivel de optimización. Intentar buscar explicación a ambas características, comparando los códigos ASM generados

Como podemos ver en la gráfica (Figura 2) la cuarta versión si que es más rápida que la tercera para -O0 y -O1 aunque para -O2 tardan lo mismo y si que tarda menos con las optimizaciones de gcc.

8. En la 6º versión se usa un **clobber** EDX. Probar a quitarlo y recompilar con los tres niveles de optimización. ¿Pasa algo? ¿Para qué niveles? ¿Por qué? La explicación debe basarse en el código ASM generado.

Si lo quitamos para -O1 y -O2 deja de funcionar correctamente esa versión.
Utiliza el registro %ecx en vez de %edx que el que nosotros utilizamos en nuestro código.

8. Gráfica comparación de optimizaciones de gcc para parity.c

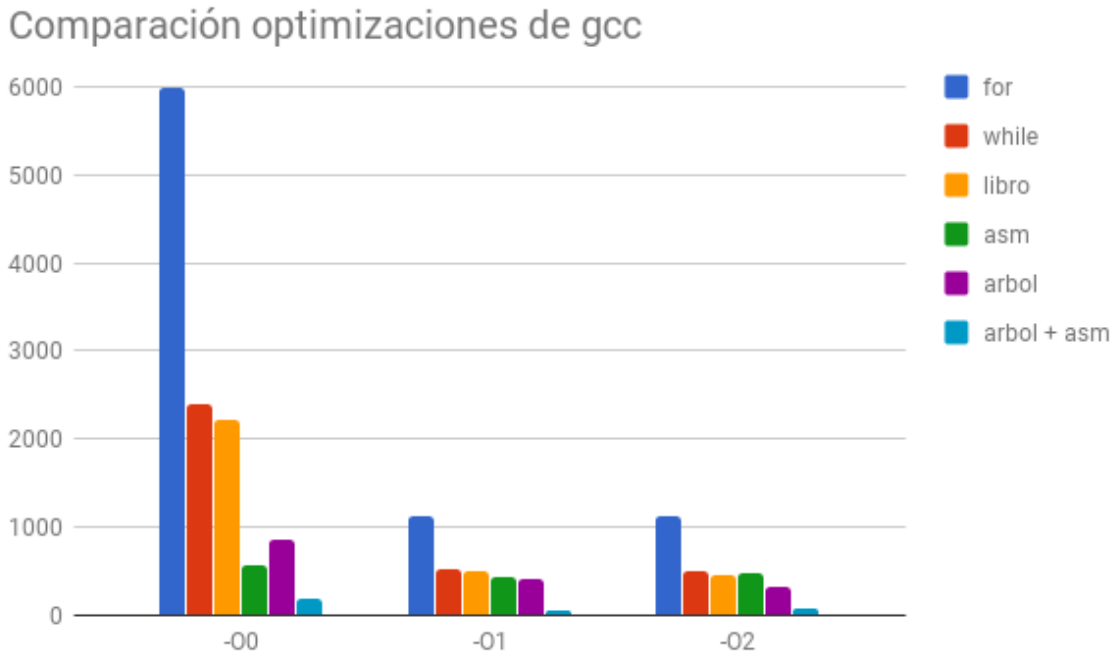

Figura 2: Gráfica rendimientos parity.