Nombre:_____ Grupo: _____

Nombre: _____

Hoja de respuesta al Estudio Previo

1. Hacer “inlining”de una función significa:

Una función inline es una optimización de dicha función que nos permite reducir el tiempo de ejecución, para ello el inlining remplaza la función haciendo una copia del código de esta, además podría recompilar otras funciones que se llamen dentro de la función inline para optimizar el código.

Para especificar el inline de una función:

```
inline void nombreFuncion();
```

Aunque nosotros queramos que el compilador haga el inline, este considera si aplicarlo o no.

2. La opción específica de compilación de gcc que permite al compilador hacer “inlining”de todas las funciones simples es (especifica si se activa o no al activar la opción -O2). ¿Para qué sirve la opción -finline-limit?:

Si usamos el comando man gcc y nos desplazamos al apartado de las optimizaciones podemos ver que al aplicar la opción -O2, el único inline que aplica es el siguiente:

```
-finline-small-functions:
```

Aplica el inlining si el cuerpo de la función es más pequeño que el código de llamada a la función. El compilador es el que decide que funciones son lo suficientemente sencillas para que merezcan aplicar el inline.

Por lo que la respuesta es que el compilador aplica el inline a las funciones que este considera lo suficientemente simples.

La opción -finline-limit permite controlar el tamaño máximo que pueden tener las funciones que queremos aplicar el inline. Para usar esta opción hay que especificar una serie de parámetros, como por ejemplo el tamaño máximo deseado.

3. Explica una forma práctica de saber si en un programa ensamblador existe la función “Pedrito”. Explica cómo averiguar si, además de existir, esa función es invocada o no.

Para ver si existe una función “Pedrito” en ensamblador:

a) Ejecutamos el siguiente comando para obtener el código en ensamblador (si es que aún no lo tenemos):

```
gcc -S nombre_programa.c
```

b) Abrimos el archivo y miramos si existe una línea de código que contenga:

```
.globl      Pedrito
```

Para ver si la función es invocada mirar si existe alguna línea de código que contenga:

```
call Pedrito
```

4. El primer código ensamblador tiene:

Instr. estáticas: 5 Instr. dinámicas: 5 000 000

Si la ejecución tarda 10 ms y 14000000 de ciclos:

MIPS: $\frac{N}{t_{ej}*10^6} = \frac{5*10^6}{10ms*10^6} = 500$ IPC: $\frac{N.}{ciclos} = \frac{5*10^6}{14*10^6} = 0{,}357$

CPI: $\frac{ciclos.}{N} = \frac{14*10^6}{5*10^6} = 2{,}8$ frecuencia: $\frac{CPI*N}{t_{ej}} = \frac{2{,}8*5*10^6}{10ms} = 1{,}4\ GHz$

5. El segundo código (compilado con -O) tiene:

Instr. estáticas: 4 Intsr. dinámicas: 4 000 000

Si la ejecución tarda 5 ms y 7000000 de ciclos:

MIPS: $\frac{N}{t_{ej}*10^6} = \frac{4*10^6}{5ms*10^6} = 800$ CPI: $\frac{ciclos.}{N} = \frac{7*10^6}{4*10^6} = 1{,}75$

frecuencia: $\frac{CPI*N}{t_{ej}} = \frac{1{,}75*4*10^6}{5ms} = 1{,}4\ GHz$ Speedup: $\frac{T_1}{T_2} = \frac{10\ ms}{5\ ms} = 2$

Las igualdades y diferencias observadas respecto al apartado anterior se deben a:

El MIPS y la CPI son iguales debido a que usamos una opción de compilación diferente, lo que hace que el número de ciclos y el tiempo de ejecución varíe, en cambio la frecuencia se mantiene, ya que el número de ciclos es proporcional al tiempo de ejecución.

6. El programa total puede obtener un Speedup de:

Si el código es instantáneo: $\frac{200\ ms}{0} = \infty$ Si se compila con -O: $\frac{200\ ms}{4\ ms} = 50$

7. Una forma práctica para medir el rendimiento (MIPS e IPC) del programa en C que acabamos de ver es:

Obtenemos el número de instrucciones dinámicas del programa, posteriormente implementamos las funciones GetTime() y getticks() al código inicial para averiguar el número de ciclos y el tiempo que tarda en ejecutarse el código para finalmente medir el rendimiento usando las siguientes fórmulas:

$$MIPS = \frac{\#instrs}{10^6 * t_{ej}} \quad \text{y} \quad CPI = \frac{ciclos}{\#instrs}$$

8. Dadas 5 ejecuciones de 10ms, 8ms, 13ms, 4ms y 2ms. Su media:

Geométrica: $\sqrt[5]{10ms * 8ms * 13ms * 4ms * 2ms} = 6{,}081\ ms$

Aritmética: $\frac{(10+8+13+4+2)}{5}\ ms = 7{,}4\ ms$

Descartando los valores extremos su media es:

Geométrica: $\sqrt[3]{10ms * 8ms * 4ms} = 6{,}84\ ms$

Aritmética: $\frac{(10+8+4)}{3}\ ms = 7{,}33\ ms$

Se observa que:

Obtenemos que la media geométrica y la media aritmética se parecen más después de descartar los valores extremos de su media.