TEC | Tecnológico de Costa Rica

Instituto Tecnológico de Costa Rica **Área Académica de Ingeniería en Computadores** **Programa de Licenciatura en Ingeniería en Computadores** **Curso:** CE-4301 Arquitectura de Computadores I **Profesor:** Ronald García Fernández **Semestre:** I, 2019	**Examen Final Parte Practica** **Fecha:** 10 de junio, 2019 **Puntos totales:** 100 **Puntos obtenidos:**_____

Nombre: _____ Carné: _____

Instrucciones Generales

- Trabaje individualmente o en grupos no más de 2 personas.
- Justifique sus respuestas adecuadamente.
- El código entregado debe poseer comentarios de no ser así no se revisará, además debe indicar como compilár y usar su código (README, Makefile, etc).
- Para la resolución de los problemas y generación de resultados se pide que emplee GCC como compilador e investigue el uso de herramientas como ***valgrind, readelf, objdump, goldbolt***.
- Debe indicar como usó las herramientas para obtener los resultados.

Parte I Manejo del stack y llamada a funciones en C para un ISA objetivo [55 puntos]

En el lenguaje de programación **C** existen diferentes calificadores para las variables que afectan diferentes aspectos del código generado, para el caso de esta evaluación se quiere analizar el código generado, el manejo del *stack* y el desempeño de llamadas a funciones de 2 tipos: funciones con variables tipo ***static*** y funciones simples, como se muestran en las figuras 1, 2.

```
1  int method_with_static_variables (int x, int y, int z){
2
3      static int counter;
4      counter++;
5      return (x + y + z + counter);
6  }
7
```

Figura 1. Código de función con variable tipo ***static***.

```
1  int method_no_static_variables (int x, int y, int z){
2
3      int counter;
4      counter++;
5      return (x + y + z + counter);
6  }
```

Figura 2. Código de función simple (sin variables tipo ***static***).

Respecto a las 2 funciones anteriores se le pide para una arquitectura específica lo siguiente:

1- Explicar en qué consiste el *call stack* y cual es procedimiento de llamado a funciones su ISA. (5 puntos)

2- Explicar en qué consiste una variable tipo **`static`** en **C**, qué características tiene este tipo respecto al *call stack*. (5 puntos)

3- Explicar en qué consisten los archivos ***ELF*** indicando características principales de sus secciones. (5 puntos)

4- Generar el código de ASM de las 2 funciones y explicar basado en los puntos 1 y 2 la estructura del mismo (no use optimizaciones de GCC) (15 puntos)

5- Generar reporte de los archivos ***ELF*** lo más resumido posible de cada una de las 2 funciones y discutir lo obtenido basándose en lo investigado en el punto 3. (10 puntos)

6- Crear un código de pruebas para evaluar el desempeño de los 2 tipos de funciones respecto a tiempo de ejecución y consumo de memoria (Sugerencia investigue sobre *callgrind*). Justifique cualquier cambio que necesite realizar, así como el objetivo de la prueba y su procedimiento. (15 puntos)

Parte II Uso de C++ contenedores STL su efecto en el desempeño de la memoria cache [45 puntos]

La biblioteca estándar de ***C++ (STL)*** ofrece diferentes tipos de contenedores que permiten diferentes funcionalidades, para efectos de esta evaluación se requiere que analice el efecto del uso de los contenedores **`std::vector`** y **`std::list`** en el desempeño de la memoria caché, para esto se le solicita:

1- Explicar las diferencias entre **`std::vector`** y **`std::list`** indicando como es su manejo de memoria (*allocation, search, etc*). (5 puntos)

2- Explicar como recorrer eficientemente a través de estos contenedores. (Sugerencia use **`iterators`**) (5 puntos)

3- Crear un código de prueba <u>para cada uno de los contenedores</u> donde implemente un vector y una lista de tipo **`double`** con valores aleatorios entre `[-1, 1]`, garantizando que sean del mismo tamaño (suficientemente grande), que realice la siguiente operación:

```
y = (x[i+1] + x[i] + x[i-1])/3
```

Evalúe el desempeño de la memoria caché para cada uno de los códigos de prueba (10 puntos)

4- Analice los resultados obtenidos empleando los conceptos teóricos sobre jerarquías memoria y las características del caché su sistema, la herramienta empleada y de lo investigado en el punto 1 y 2. (25 puntos)

<u>Formato del entregable</u>

Se debe entregar un archivo tipo ***zip*** nombrado de la siguiente forma:

<NOMBRE1>_<CARNET1>_<NOMBRE2>_<CARNET2>_examen_practico.zip

Con los siguientes contenidos:

1- Documento donde conteste las preguntas de las partes I y II.
2- Códigos fuente para las pruebas de la sección I.
3- Códigos fuente de la sección II.
4- README con indicaciones para compilar y correr su código (Puede agregar un Makefile)

<u>Fecha límite de entrega 12 junio 11:00 pm</u>