Universidad de Costa Rica
Escuela de Ingeniería Eléctrica
IE0521 Estructuras de computadoras digitales II

Tarea I: Predictores de saltos

Objetivos

Analizar el redimiendo de distintos esquemas de predicción de saltos por medio de simulaciones en un lenguaje de alto nivel.

Descripción del Proyecto

Cada estudiante deberá realizar un simulador de distintos predictores de saltos, en C\C++, python o Go, que permita obtener métricas de rendimiento utilizando como entrada un *trace* donde se indica la dirección del salto así como el resultado de este.

Parámetros ejecución y resultados de la simulación

Se trabajará sobre un trace con dieciséis millones de saltos condicionales, estos son el resultado de la ejecución de gcc (colección de compiladores de GNU), que es parte de la suite *SPECint2000*. El archivo puede ser descargado del siguiente enlace (TraceLink). Se sugiere que se descargue el archivo comprimido y se procese sin descomprimirlo.

El formato del archivo es el siguiente:

```
3086629576 T
3086629604 T
3086629599 N
3086629604 T
```

Cada línea cuenta con dos datos: el primero, es la dirección de la instrucción del salto y el segundo, es un carácter *T* o *N* que indica si el salto se tomó o no.

El diseño deberá ser parametrizable, es decir, por medio de argumentos se indicará las características del cache a simular.

Los parámetros de entrada serán los siguientes:

1. Tamaño de la tabla BTH. (-s)
2. Tipo de predicción (-bp)
3. Tamaño del registro de predicción global (-gh)
4. Tamaño de los registros de historia privada (-ph)

El programa deberá poder ejecutarse de la siguiente forma:

gunzip -c branch-trace-gcc.trace.gz | branch -s < # > -bp < # > -gh < # > -ph < # >

El número de entradas de la tabla BHT esta dado por 2^s, donde "s."es el tamaño indicado al ejecutar el programa.

Al ejecutar el programa, la siguiente información deberá imprimirse en consola:

Prediction parameters:	
Branch prediction type:	Bimodal
BHT size (entries):	16
Global history register size:	5
Private history register size:	3
Simulation results:	
Number of branch:	0000
Number of correct prediction of taken branches:	0.00
Number of incorrect prediction of taken branches:	0.00
Correct prediction of not taken branches:	0000
Incorrect prediction of not taken branches:	0.00
Percentage of correct predictions	0.00

Descripción de los predictores

El estudiante deberá implementar cuatro tipos de predictores. Estos se describen a continuación.

Predictor Bimodal (-bp 0)

Es el predictor dinámico más simple. Es básicamente un arreglo de contadores de 2bits, donde cada contador puede tomar uno de los siguientes valores :

- *Strongly taken*
- *Weakly taken*
- *Weakly not taken*
- *Strongly not taken*

El tamaño del arreglo se deriva de la opción "-s", el número de entradas del arreglo es 2^s. Para realizar una predicción se selecciona un contador de la tabla usando los últimos n bits del contador del programa (en este caso el valor de n esta dado por s). Todos los contadores se deberán inicializar en el estado *strongly not taken*.

Predictor con historia privada (-bp 1)

Este predictor, es muy similar al anterior, pero en lugar de tener un solo registro con la historia global, tiene toda una tabla con la historia de cada salto.

Tanto el tamaño del BHT como el PHT están dadas por el parámetro "-s". La cantidad de bits de historia que se almacenan en cada entrada de la tabla esta dada por la opción "-ph". Para indexar el arreglo de contadores se utilizan los últimos n bits resultado de la operación XOR entre el contador del programa y el registro privado de historia.

Todos los contadores se deberán inicializar en el estado *strongly not taken*.

Predictor con historia global (-bp 2)

Este predictor usa un arreglo de tamaño 2^s, donde se almacenan de contadores de 2bits que indican la predicción, y un registro con la historia de los últimos saltos, la cantidad de bits de este registro esta dada por la opción "-gh".

Para realizar la predicción, se utilizan los últimos n bits resultado de la operación XOR entre el contador del programa y el registro de historia.

Todos los contadores se deberán inicializar en el estado *strongly not taken*.

Predictor con torneo (-bp 3)

Para este metapredictor se usa el predictor con historia privada y el predictor con historia global. Los dos predictores son accedidos en paralelo y cada uno genera un predicción de forma independiente. Para elegir entre ambas predicciones de usa un meta predictor que indica cual de las dos predicciones usar. El metapredictor es una estructura de tamaño 2^s con contadores de 2bits que indicando el predictor a usar. Todos los contadores se deberán inicializar en el estado *strongly not taken*. Y el meta predictor deberá ser inicializado en el estado *strongly prefer pshared*

Verificación del diseño

Para la verificación del diseño se brindan resultados anotados de los primeros 200 saltos. Estos se encuentran en el siguiente enlace (Resultados)

Evaluación y entrega

Esta primera parte del proyecto tiene un valor de un 10 % y se realizará en forma individual. Para parte de la revisión de la tarea de usara el pipeline de integración continua de gitlab. Este ejecutará el código y comparará los resultados del programa del estudiante contra resultados previamente calculados. Para esto se provee el siguiente repositorio de git (GitLink).

En el repositorio se pueden encontrar los siguientes archivos:

- *compare_result.sh*: script de shell, que ejecutara el código y comparara los resultados.
- *.gitlab-ci.yml*: archivo de configuración de las etapas de integración continua. Esta configura para un proyecto en C++, pero el estudiante podrá cambiar la etapa de construcción para el lenguaje que haya decidido usar.
- expected_result_*.txt: archivos con los resultados esperados. Inicialmente vacíos, una semana antes de la entrega se habilitaran con resultados reales.

Para la calificación se utilizará la rúbrica del Cuadro 1.

La entrega se realizará por medio de un repositorio de git, que deberá contar con la descripción del programa, así como las instrucciones o dependencias para ejecutarlo. En mediación virtual, se habilitará un espacio donde el estudiante deberá proveer la dirección a su repositorio, antes del día **Jueves 10 de setiembre** a media noche.

Cuadro 1: Desglose de la evaluación

Item	Puntaje Máximo
Resultados Predictor Bimodal	20
Resultados Predictor Pshare	20
Resultados Predictor Gshare	20
Resultados Predictor Torneo	15
Procesamiento de entradas	5
Eficiencia (Ejecución menor a dos minutos)	10
Orden y formato del código	10

Otras consideraciones

- Se asume que el estudiante tiene conocimientos de programación, por lo que cualquier refrescamiento del lenguaje y herramientas es responsabilidad del estudiante.
- Se utilizará como sistema operativo base Ubuntu 18.04 en adelante, por lo que el código y sus instrucciones de construcción deben poder ser ejecutadas en este sistema operativo.
- El código debe ser legible y estar comentado apropiadamente. Elija un formato para los comentarios y un formato para el nombre de las variables. Sea consistente con los formatos elegidos a lo largo de su programa.
- Cada función del programa debe contener un encabezado indicando una descripción general, los parámetros de entrada y los de salida.
- Cada día de retraso en la entrega reducirá la nota máxima en 5 %.