Universidad de Costa Rica

Escuela de Ingeniería Eléctrica

**IE-0521 Estructuras de Computadoras Digitales II**

Luis Javier Herrera Barrantes, B93840

Erick Carvajal Barboza, PhD

21 de octubre del 2022

---

# Tarea 3: Simulación de Procesador con Ejecución Fuera de Orden

---

## 1. Resultado de contadores con configuración por defecto

### 1.1. Contadores utilizados

#### 1.1.1. % Instrucciones de enteros finalizadas (commit) que no son Multiplicación o Divisón

Se utilizaron los siguientes contadores:

- `simInsts` # Number of instructions simulated (Count)
- `system.cpu.statIssuedInstType_0::IntDiv` # Number of instructions issued per FU type, per thread (Count)
- `system.cpu.statIssuedInstType_0::IntMult` # Number of instructions issued per FU type, per thread (Count)

#### 1.1.2. % Instrucciones de enteros finalizadas que son Multiplicación o Divisón

Para calcular este resultado simplemente se calculó el porcentaje restante del resultado anterior para llegar al 100 %.

### 1.1.3. % Instrucciones finalizadas de punto flotante

Se utilizaron los siguientes contadores:

- `system.cpu.commit.floating` # Number of committed floating point instructions. (Count)
- `simInsts` # Number of instructions simulated (Count)

### 1.1.4. % Instrucciones finalizadas que son branch

Se utilizaron los siguientes contadores:

- `system.cpu.commit.branches` # Number of branches committed (Count)
- `simInsts` # Number of instructions simulated (Count)

### 1.1.5. % Instrucciones finalizadas que son referencias a memoria o caché

Se utilizaron los siguientes contadores:

- `system.cpu.icache.tags.totalRefs` # Total number of references to valid blocks. (Count)
- `system.cpu.dcache.tags.totalRefs` # Total number of references to valid blocks. (Count)
- `system.cpu.commit.memRefs` # Number of memory references committed (Count)
- `simInsts` # Number of instructions simulated (Count)

### 1.1.6. Promedio de micro-operaciones simuladas por instrucción

Se utilizaron los siguientes contadores:

- `simOps` # Number of ops (including micro ops) simulated (Count)
- `simInsts` # Number of instructions simulated (Count)

### 1.1.7. % branches predicho correctamente

Se utilizaron los siguientes contadores:

- `system.cpu.branchPred.condPredicted` # Number of conditional branches predicted (Count)
- `system.cpu.branchPred.condIncorrect` # Number of conditional branches incorrect (Count)

### 1.1.8. Promedio de instrucciones finalizadas por ciclo

Se utilizó el siguiente contador:

- `system.cpu.ipc` # IPC: Instructions Per Cycle ((Count/Cycle))

### 1.1.9. Promedio de micro-operaciones finalizadas por ciclo

Se utilizó el siguiente contador:

- `system.cpu.ipc` # IPC: Instructions Per Cycle ((Count/Cycle))

Para calcular este resultado se utilizó también el **promedio de micro-operaciones simuladas por instrucción**.

### 1.1.10. Hit Rate del caché de instrucciones

Se utilizaron los siguientes contadores:

- `system.cpu.icache.overallHits::cpu.inst` # number of overall hits (Count)
- `system.cpu.icache.overallMisses::cpu.inst` #number of overall misses (Count)

### 1.1.11. Hit Rate del caché de datos

Se utilizaron los siguientes contadores:

- `system.cpu.dcache.overallHits::cpu.inst` # number of overall hits (Count)
- `system.cpu.dcache.overallMisses::cpu.inst` # number of overall misses (Count)

### 1.2. Tabla Resultado

Al realizar todas las debidas operaciones y cálculos se obtuvo la tabla resultado mostrada en la Figura 1.

| | blocked-matmul | BFS | SHA | queens |
|---|---|---|---|---|
| % Instrucciones de **enteros finalizadas (commit)** que **no** son **Multiplicación** o **Divisón** | 99.87% | 98.75% | 99.98% | 99.96% |
| % Instrucciones de **enteros finalizadas** que son **Multiplicación** o **Divisón** | 0.13% | 1.25% | 0.02% | 0.04% |
| % Instrucciones **finalizadas** de **punto flotante** | 76.06% | 72.78% | 18.64% | 0.14% |
| % Instrucciones **finalizadas** que son **branch** | 2.32% | 24.07% | 4.31% | 24.67% |
| % Instrucciones **finalizadas** que son referencias a **memoria** o **caché** | 71.90% | 70.12% | 42.30% | 111.61% |
| Promedio de **micro-operaciones** simuladas **por instrucción** | 1.742 | 1.909 | 1.301 | 2.019 |
| % **branches** predicho **correctamente** | 95.96% | 97.77% | 94.82% | 89.13% |
| Promedio de **instrucciones finalizadas** por **ciclo** | 2.134 | 0.943 | 1.998 | 0.802 |
| Promedio de **micro-operaciones finalizadas** por **ciclo** | 3.718 | 1.800 | 2.599 | 1.619 |
| **Hit Rate** del **caché** de **instrucciones** | 0.9986 | 0.998 | 0.998 | 0.998 |
| **Hit Rate** del **caché** de **datos** | 0.9906 | 0.896 | 0.991 | 0.992 |

Figura 1: Tabla resultado obtenida al correr la simulaciones para el caso *default*

## 2. Variaciones por configuración

### 2.1. Cambios del ROB

En la Figura 2 se puede apreciar los resultados obtenidos al realizar la simulaciones para los cuatro programas propuestos a la hora de variar el tamaño del ROB.

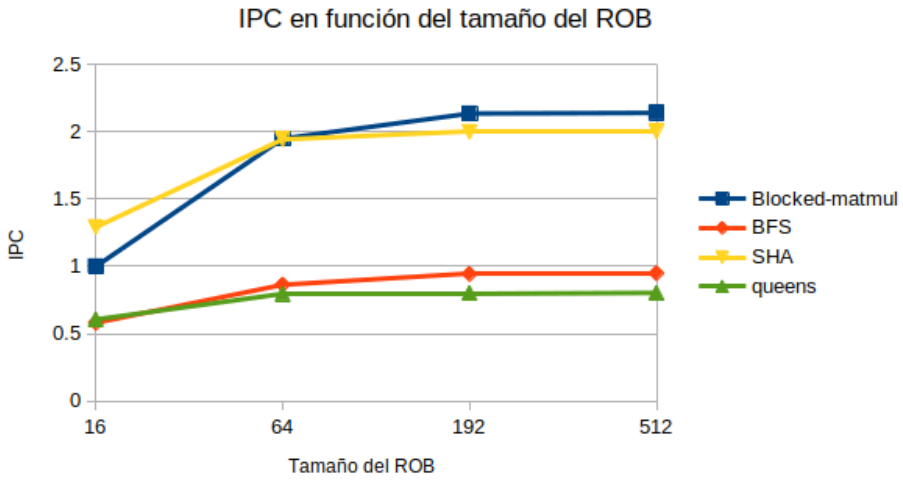


Figura 2: Gráfica resultado ante variaciones en el tamaño del ROB para varias simulaciones

Se destaca entonces que los cuatro programas mostraron comportamientos bastante similares. Es decir, el IPC para todos tendió a aumentar conforme se fue aumentando el tamaño del ROB, lo cual es de esperar. Los programas de *Blocked-matmul* y *BFS* tuvieron un IPC notablemente superior a *queens* y *BFS*. Estas diferencias se pueden deber principalmente al tipo de programa que se está simulando en `gem5`.

## 2.2. Cambios del ancho de pipeline

En la Figura 3 se aprecian los resultados de IPC obtenidos para los cuatro programas propuestos al variar el ancho del pipeline.

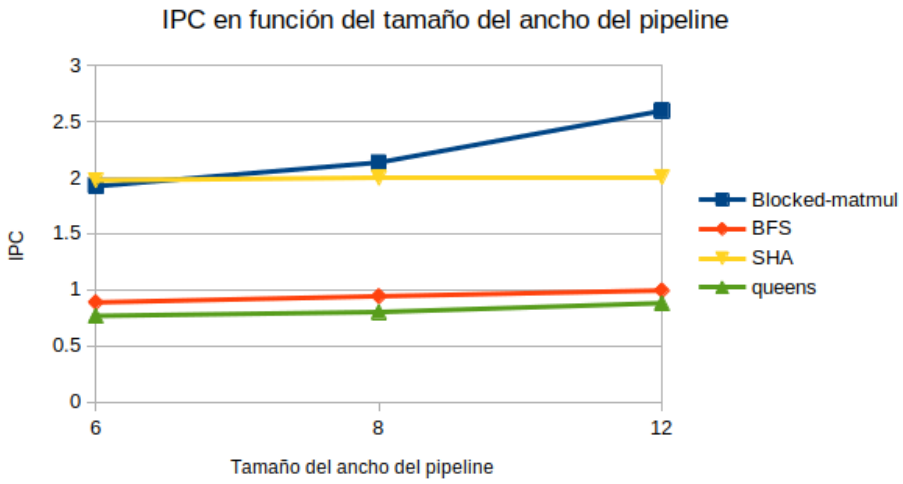


Figura 3: Gráfica resultado ante variaciones en el tamaño del ancho del pipeline para varias simulaciones

Para este caso se tuvo que los cuatro programas simulados mostraron comportamientos parecidos. El comportamiento predominante en la simulación fue la tendencia a subir el IPC conforme se fuera aumentando el ancho del pipeline para el simulador. Sin embargo, para el caso de *SHA*, *queens* y *BFS* el aumento de su IPC fue prácticamente imperceptible, dando el efecto de una linea recta a la hora de realizar la gráfica. Por otro lado, para el único caso del programa de *Blocked-matmul* el aumento del IPC si fue bastante notable conforme aumentó el ancho del pipeline. El cambio tan drástico que diferencia el comportamiento de los programas simulados se puede deber principalmente a las configuraciones por defecto del compilador `gem5` y al tipo de programas que se esté ejecutando.

## 2.3. Cambios del predictor de saltos

En la Figura 4 se evidencian los resultados obtenidos al correr los cuatro programas en el compilador `gem5`.

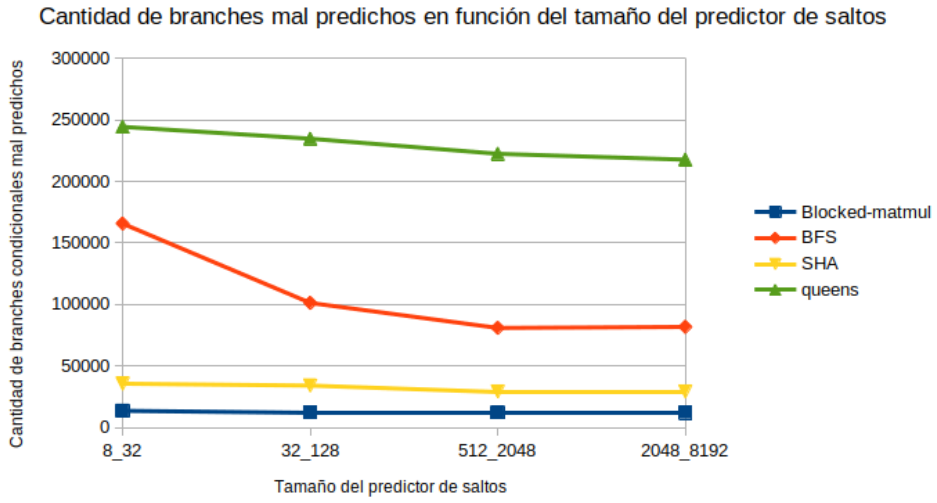


Figura 4: Gráfica resultado ante variaciones en el tamaño del predictor de saltos para varias simulaciones

Para este caso se propuso la observación de la variación de la cantidad de branches mal predicho al cambiar el tamaño del predictor de saltos del simulador. Esto para diversas configuraciones de tamaño del predictor de saltos. Como era de esperar, la tendencia para los cuatro programas simulados fue la de disminuir la cantidad de branches erróneamente predichos conforme se fue aumentando el tamaño del predictor. En la gráfica de resultados no hay ningún comportamiento fuera de lo normal que se deba destacar, simplemente se aprecia una diferencia en la cantidad de branches mal predichos entre las aplicaciones simuladas lo cual es bastante común para la totalidad de pruebas realizadas en la tarea. Para la aplicación de *BFS* se observa

una disminución de los branches mal predichos mucho más pronunciada que las otras tres. Como ya se mencionó esto se debe al tipo de aplicación que se está simulando.

## 2.4. Cambios del caché de instrucciones

Para esta sección se simuló la cantidad de misses en el caché de instrucción a la hora de variar el tamaño de dicho caché. En la Figura 5 se observan los resultados.

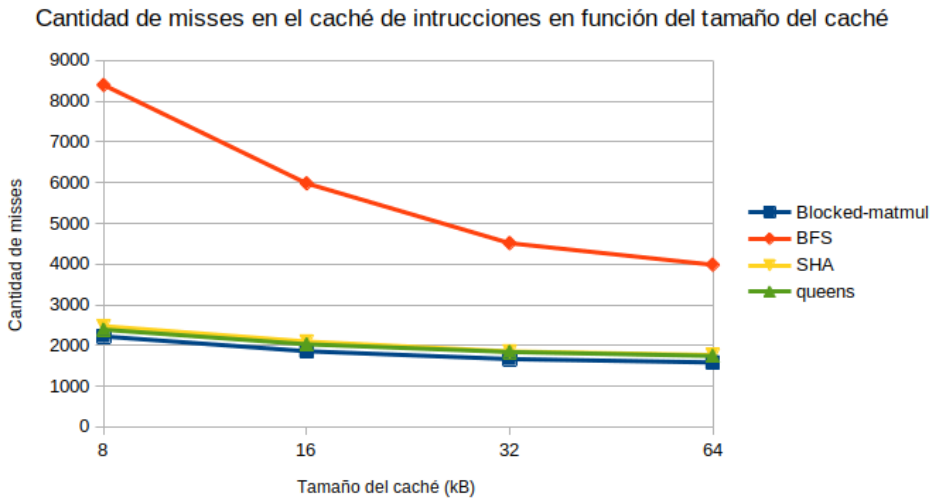


Figura 5: Gráfica resultado ante variaciones en el tamaño del caché de instrucciones para varias simulaciones

Al variar el tamaño del caché de instrucciones se puede notar que tres (*queens*, *Blocked-matmul* y *SHA*) de las cuatro aplicaciones muestran comportamientos extremadamente similares. Tendiendo a disminuir la cantidad de misses al aumentar el tamaño del caché de instrucciones. Por otro lado, para el caso de la aplicación *BFS*, esta muestra el mismo comportamiento descrito pero mucho más pronunciado. Es decir, la cantidad de misses llega a diferer considerablemente más al aumentar el tamaño del caché que para el resto de aplicaciones. Al igual que las secciones anteriores este comportamiento se puede deber al tipo de aplicación que se está simulando ya que al analizar las diferentes pruebas realizadas en la totalidad de la tarea no se logra observar un comportamiento similar y que resalte entre las mismas aplicaciones.

## 2.5. Cambios del caché de datos

Por último, en la Figura 6 se observa el comportamiento de la cantidad de misses en el caché de datos para las cuatro aplicaciones simuladas al variar el tamaño del caché.

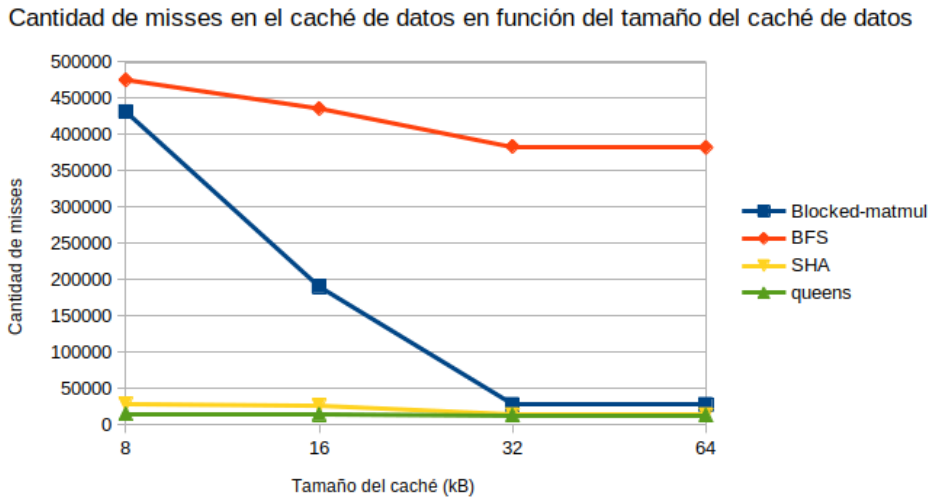


Figura 6: Gráfica resultado ante variaciones en el tamaño del caché de datos para varias simulaciones

En la presente simulación se observan comportamientos bastante distintos entre las aplicaciones simuladas. Sin embargo, se logra notar la tendencia general para las cuatro aplicaciones. Siendo esta la disminución de la cantidad de misses al aumentar el tamaño del caché para posteriormente, cuando el tamaño del caché muestra su valor máximo, notar un leve aumento en la cantidad de misses. Este aumentó en la cantidad de misses se da debido al hecho de que tener un caché muy grande aumenta los datos guardados en caché que no se van a utilizar lo cual es bastante ineficiente y tiene efectos negativos en el rendimiento. Además, se destaca que las aplicaciones de *queens* y *SHA* muestran comportamientos prácticamente idénticos mientras que *Blocked-matmul* y *BFS* muestran comportamientos diferentes entre sí y diferentes entre las demás, siendo *Blocked-matmul* el que muestran el cambio en la cantidad de misses más drástico de las cuatro. Esto debido a la diferencias entre los tipos que de programas que se están corriendo en `gem5`.