# Análisis de Problemas: Informe de Diseño.

Garro M. Max
*Tecnológico de Costa Rica*
*Estudiante de Ingeniería en Computadores*
Email: maxgarro22@estudiantec.cr

Johnson S. Naheem
*Tecnológico de Costa Rica*
*Estudiante de Ingeniería en Computadores*
Email: naheemjohnson@estudiantec.cr

## I. Identificación de las variables.

Las variables se cambiaron para poder observar el comportamiento de cada ejecución. Además que se cambiaron algunos parámetros para poder observar si cambiaban las estadísticas, como es el caso de los branch predictors:

- globalPredictor y choicePredictorSize:Mejora en la precisión de predicción para ramas con comportamientos locales, lo que puede reducir el CPI y mejorar el IPC.
- nHistoryTables: Más tablas de historial permiten capturar una mayor diversidad de patrones de ramas, mejorando la precisión del predictor.
- localPredictorSize: Un tamaño máximo de historial de 150 permite que el predictor capture patrones de ramas más largos.
- maxHist: Un tamaño máximo de historial de 150 permite que el predictor capture patrones de ramas más largos.
- logSizeLoopPred: Un tamaño logarítmico de 4 para el predictor de bucles permite capturar mejor los patrones de bucles.

| **Parámetro de Branch Predictor** | **Impacto en rendimiento** |
|---|---|
| `[TournamentBP]` (globalPredictor=16384, choicePredictorSize=16384) | Mejora en precisión de predicción, reduce CPI, aumenta IPC, menor tiempo de ejecución |
| `[BiModeBP]` (globalPredictor=16384, choicePredictorSize=16384) | Similar al TournamentBP, mejora en precisión de predicción, reduce CPI, aumenta IPC |
| `[LTAGE]` (nHistoryTables=16) | Mejora significativa en precisión de predicción, reduce CPI, aumenta IPC |
| `[LocalBP]` localPredictorSize = Param.Unsigned(4096, "Size of local predictor") | Mejora en precisión de predicción para ramas locales, reduce CPI, aumenta IPC |
| `[TAGE]` maxHist = Param.Unsigned(150, "Maximum history size of TAGE") | Mejora en precisión de predicción para patrones largos, reduce CPI, aumenta IPC |
| `[TAGE_SC_L_8KB]` (logSizeLoopPred=4) | Mejora en precisión de predicción para bucles, reduce CPI, aumenta IPC |

Tabla I: Impacto de los diferentes parámetros de Branch Predictor en el rendimiento

Y por otro lado, variando los parámetros que se encuentran en $runGem5.sh$, se logran los siguiente impactos:

| Parámetro | Hit Rate/Miss Rate | CPI | IPC | BTB | Tiempo de Ejecución |
|---|---|---|---|---|---|
| –l2_size | Mejor Hit Rate/Menos Miss Rate con tamaño mayor | Menor CPI con tamaño mayor | Mayor IPC con tamaño mayor | N/A | Menor tiempo con tamaño mayor |
| –replacement_policy | Dependiendo de la política, puede mejorar Hit Rate | Mejor CPI con política eficiente | Mejor IPC con política eficiente | N/A | Menor tiempo con política eficiente |
| –bp-type | N/A | Mejor CPI con predicción eficiente | Mayor IPC con predicción eficiente | Mejor rendimiento del BTB | Menor tiempo con predicción eficiente |
| –cpu-type | N/A | Menor CPI con O3CPU | Mayor IPC con O3CPU | N/A | Menor tiempo con O3CPU |
| Arquitectura | Dependiente de implementación | Dependiente de implementación | Dependiente de implementación | Dependiente de implementación | Dependiente de implementación |

Tabla II: Impacto de varios parámetros en diferentes métricas de rendimiento

Dependiendo del tipo de parámetro se ve afectado un valor de estudio, como en el caso cuando se varía bp-type, se obtienen todos los valores menos el Hit Rate y Miss Rate, ya que estos últimos no se ven afectados por el cambio de dicho parámetro.

## II. Formulación de benchmarks.

### *II-A. 429.mcf (SPEC CPU2006 Benchmark)*

Se utiliza para evaluar para evaluar el rendimiento de las CPU en tareas intensivas. Al ser un benchmark mcf, resuelve un problema de flujo de red usando un algoritmo llamado Simplex. [**3**] Este tiene patrones de acceso a memoria complejos y menos localizados, lo que lo hace ideal para probar la eficiencia de la caché y las políticas de reemplazo. Debido a sus intensivos cálculos, requiere una CPU con buen rendimiento para resolver el problema en un tiempo moderado.

### *II-B. blackscholes (PARSEC Benchmark)*

El blackscholes es parte del conjunto de benchmarks PARSEC (Princeton Application Repository for Shared-Memory Computers), diseñado para evaluar el rendimiento de sistemas paralelos y de memoria compartida.Este implementa el modelo de Black-Scholes, un modelo matemático para la valoración de opciones financieras. Este benchmark está diseñado para aprovechar arquitecturas paralelas, lo que lo hace útil para evaluar el rendimiento de sistemas multiprocesador y multi-core. Aunque no es tan intensivo en memoria como $429.mcf$, el rendimiento de $blackscholes$ también puede verse afectado por la eficiencia de la caché y la jerarquía de memoria. [**2**]

## III. Formulación de un plan de solución ingenieril para la implementación del sistema.

1. Definición del problema: evaluar el rendimiento de diferentes configuraciones de predictores de ramas y tipos de CPU usando GEM5
2. Recolección de requisitos: **Funcionales:** simular diferentes configuraciones de branch predictors y CPUs en GEM5. **No funcionales:** precisión de la simulación, tiempos de ejecución razonables, capacidad de generar estadísticas detalladas.
3. Análisis y diseño: Diseñar los scripts de simulación (editando el runGem5.sh) y definir las métricas a recolectar. Planificar las configuraciones de hardware y software necesarias.
4. Planificación del proyecto: Establecer un cronograma para realizar las simulaciones. Identificar los recursos necesarios (tiempo de computación, almacenamiento).
5. Desarrollo e implementación: Configurar y ejecutar las simulaciones en GEM5. Integrar scripts y herramientas para recolectar y analizar datos de las simulaciones.
6. Despliegue y entrenamiento: Ejecutar las simulaciones en el entorno planificado. Enseñar a los miembros del equipo en la interpretación de los resultados.
7. Mantenimiento y soporte: realizar ajustes en los scripts y las configuraciones según sea necesario, además se debe moderar todas las ejecuciones por si sucede algún error inesperado.
8. Evaluación y mejora continua: Evaluar los resultados de las simulaciones e implementar mejoras basadas en los hallazgos de las evaluaciones.
9. Documentación: Crear gráficas a partir de los datos que se desean medir y también un documento de tipo SCRUM para tener sprints y mejor orden.

## IV. Evaluación de los benchmarks utilizados y resultados

En los gráficos adjuntos 1 y 2, se comparan diferentes políticas de reemplazo (Random, BTRP, LFURP, FIFORP y MRURP) evaluadas en términos de tiempo de ejecución, CPI, IPC, miss y hit rate utilizando los benchmarks PARSEC y SPEC para O3CPU. En PARSEC, todas las políticas muestran resultados muy similares, destacando Random y FIFORP por tener la misma eficiencia en tiempo de ejecución (0.00479), CPI (0.58736), IPC (1.70252) y hit rate (79.66000) y miss rate (0.00061). BTRP y LFURP también presentan un rendimiento casi idéntico con ligeras disminuciones en la tasa de aciertos (79.47000). MRURP mantiene una alta tasa de aciertos (79.91000) y un bajo tiempo de ejecución (0.00479), con pequeñas variaciones en CPI e IPC. En contraste, SPEC muestra mayores diferencias entre las políticas, con Random teniendo una tasa de aciertos significativamente más baja (67.81) y un tiempo de ejecución más alto (0.059004). BTRP y LFURP tienen tasas de aciertos similares (78.99 y 79.03) y tiempos de ejecución menores (0.023392 y 0.023401). FIFORP y MRURP también muestran altas tasas de aciertos y bajos tiempos de ejecución, con FIFORP logrando un hit rate de 77.91 y MRURP de 79.19. Las diferencias en CPI e IPC son más marcadas en SPEC, indicando un rendimiento menos eficiente en comparación con PARSEC. Esto sugiere que, dependiendo del benchmark utilizado, la eficiencia de las políticas de reemplazo puede variar significativamente.

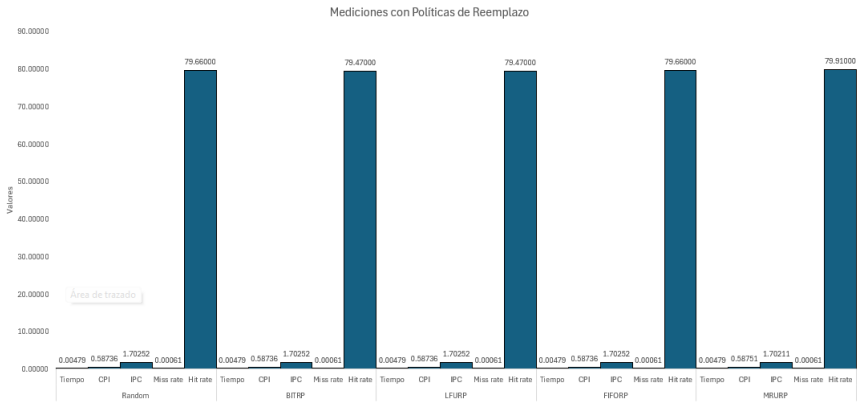


Figura 1: Mediciones con Politicas de Reemplazo utilizando PARSEC para 03CPU.

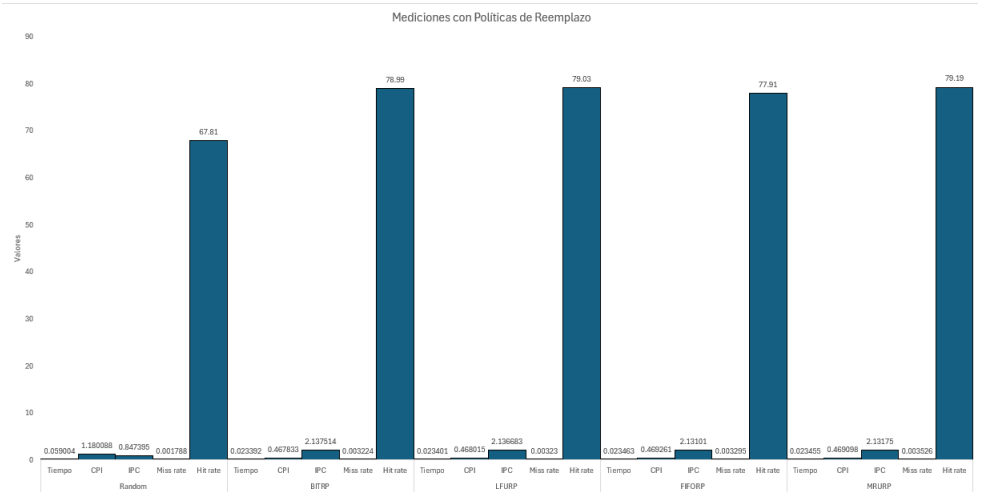


Figura 2: Mediciones con Politicas de Reemplazo utilizando SPEC para 03CPU.

En los gráficos adjuntos 3 y 4, se comparan diferentes mediciones con memoria (1MB, 2MB, 4MB, 8MB y 16MB) evaluadas en términos de tiempo de ejecución, CPI E IPC, utilizando los benchmarks PARSEC y SPEC para O3CPU. Se observa que todas las configuraciones de memoria muestran resultados idénticos en las métricas evaluadas. El tiempo de ejecución para todas las configuraciones es de 0.004785, el CPI es constante en 0.587364 y el IPC es de 1.702521. Esto indica que, en este caso, el tamaño de la memoria caché no afecta el rendimiento medido por estas métricas. Por otro lado para las mediciones realizadas utilizando SPEC, el tiempo de ejecución fue de 0.023392, un CPI de 0.467833 y un IPC de 2.137514, y observando las demas configuraciones de memoria en promedio se mantienen los valores de igual manera. Al realizar un constraste

entre PARSEC Y SPEC, se puede observar que en general PARSEC tiene un mejor tiempo de ejecución, sin embargo, la metrica SPEC tiene mejores valores de CPI e IPC, ya que el CPI es de menor valor y el IPC es de mayor valor, por lo que se produce una mejor eficiencia con las instrucciones.

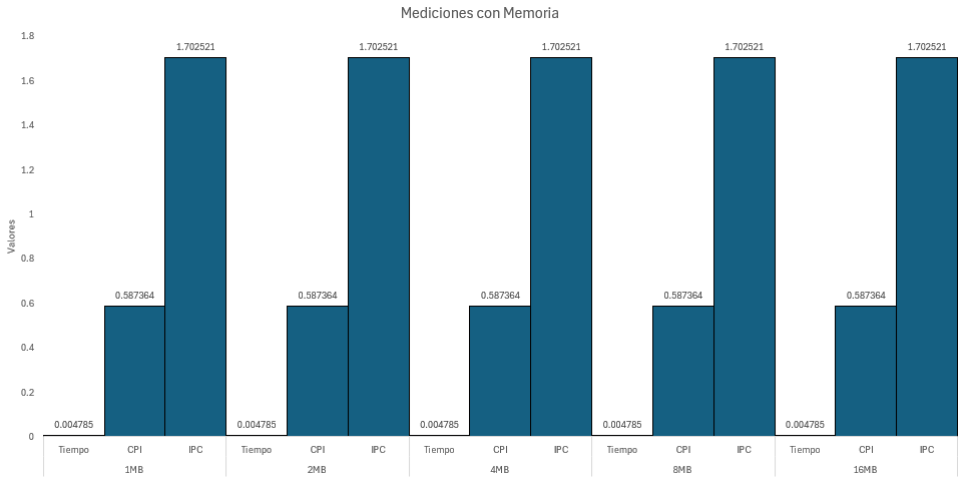


Figura 3: Mediciones con Memoria utilizando PARSEC para 03CPU.

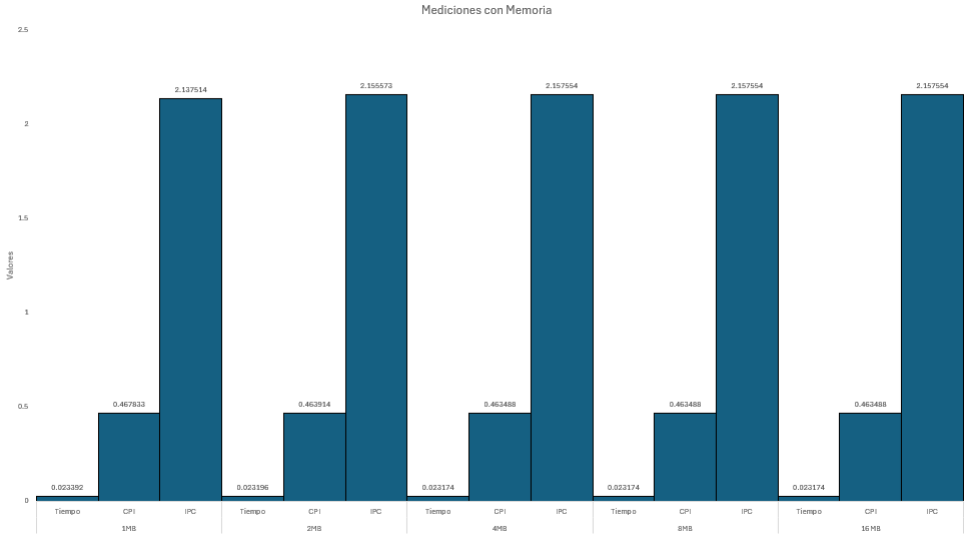


Figura 4: Mediciones con Memoria utilizando SPEC para 03CPU.

En los gráficos 5 y 6 , se comparan varios predictores de rama (TAGE_SC_L_8KB, TAGE, LTAGE, LocalBP y BiModeBP) en términos de tiempo de ejecución, CPI, IPC, y rendimiento del BTB. Los predictores TAGE y TAGE_SC_L_8KB muestran un rendimiento superior con valores de CPI bajos y altos IPC, indicando una mayor eficiencia en la predicción de ramas. En contraste, el predictor LTAGE, aunque tiene el menor tiempo de ejecución, presenta un CPI más alto y un IPC más bajo, lo que sugiere que su eficiencia general es inferior. LocalBP, con el tiempo de ejecución más alto y un CPI significativamente elevado, resulta ser el menos eficiente. Por último, BiModeBP muestra un rendimiento moderado con resultados en medio del espectro, destacando en algunos casos pero no superando a los predictores TAGE. Por otro lado, para las mediciones realizadas con SPEC, nuevamente se comparan los mismo predictores de rama, se puede observar que para las distintas métricas en comparación con PARSEC, las simulaciones realizadas con SPEC presentan un mayor tiempo de ejecución, además presentan un CPI mucho menor en comparación con las simulaciones de Parsec, lo cual implica que tienen una mayor eficiencia en promedio, y por ultimo el IPC en las simulaciones SPEC es mucho mayor lo cual implica que de nuevo es mas eficiente que el PARSEC.

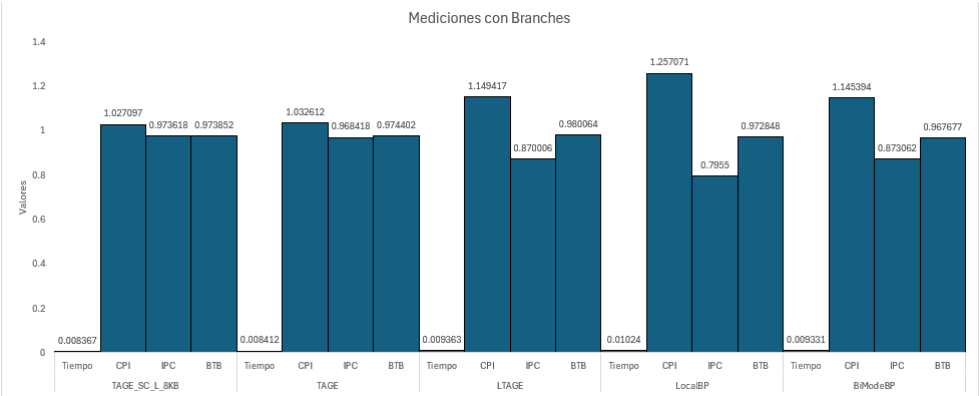


Figura 5: Mediciones con Branches utilizando PARSEC para 03CPU.

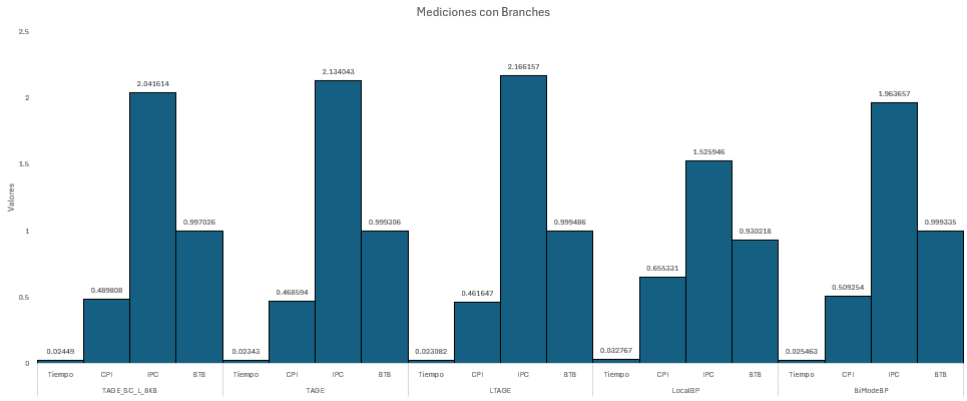


Figura 6: Mediciones con Branches utilizando SPEC para 03CPU.

## REFERENCIAS


[1] Chavarría, L. (s.f.) *Simulador GEM5*. Recuperado de https://tecdigital.tec.ac.cr/dotlrn/classes/IDC/CE4301/S-1-2024.CA.CE4301.1/file-storage/view/Apuntes%2FArqui1_Semana12_Asincr%C3%B3nica_v6.pdf
2

[2] Princeton University. *The PARSEC Benchmark Suite: Characterization and Architectural Implications*. Recuperado de https://www.cs.princeton.edu/research/techreps/TR-811-08
3

[3] Spec. *429.mcf SPEC CPU2006 Benchmark Description*. Recuperado de https://www.spec.org/cpu2006/Docs/429.mcf.html