# INSTITUTO TECNOLOGICO DE IZTAPALAPA

Ingeniería en Sistemas Computacionale

## TEMA:

Máquina de Acceso Aleatorio

## LUGAR DE REALIZACIÓN:

Instituto Tecnológico de Iztapalapa


**PROFESOR:**

Parra Hernández Abiel Tomas

**PRESENTA:**

Ortega Barrón Diana

Vega de Alba Omar Enrique

**N° DE CONTROL:**

181080479

171080114


Porcentaje de aportación
Ortega Barron Diana - 90%
Vega de Alba Omar Enrique - 90%

CIUDAD DE MÉXICO JUNIO/2021

# ÍNDICE

# ÍNDICE DE FIGURAS

# RESUMEN

También definida como la Maquina de Acceso Aleatorio, la cual maneja un doble uso dentro de la computación dado que a veces, se refiere a computadoras específicas con memorias de acceso aleatorio, es decir, memorias con acceso a cualquier otra dirección. En donde hace referencia a un cierto modelo de cómputo que no es tan abstracto, es decir, la Maquina de Turing, la cual es la más cercana en cuanto a su operación a las computadoras digitales programables estándar. Ya que por un lado, la memoria de una Máquina de Turing está completamente contenida en una cinta, haciéndola una especie de «maquina de acceso serial».

Palabras clave

memorias de acceso aleatorio, modelo de cómputo, abstracto, maquina de acceso serial, RAM, transferencia de información, acumulador, número infinito, Unidad de Control

# INTRODUCCIÓN

El concepto de máquina de acceso aleatorio (RAM) comienza con el modelo más simple de todos, el llamado modelo de máquina contador. Sin embargo, dos adiciones lo alejan de la máquina de mostrador. El primero mejora la máquina con la conveniencia del direccionamiento indirecto; el segundo mueve el modelo hacia la computadora más convencional basada en acumuladores con la adición de uno o más registros auxiliares (dedicados), el más común de los cuales se llama "el acumulador" un registro es una ubicación con una dirección (una designación / localizador único y distinguible equivalente a un número natural) y un contenido: un número natural único.

A pesar de la aparente potencia y versatilidad de las variantes de las máquinas de Turing vistas,todas ellas comparten una característica limitativa común: su dispositivo de almacenamiento es secuencial, es decir, para acceder a la información almacenada en una determinada ubicación, la máquina debe en primer lugar moverse, celda a celda, por todas las localizaciones entre la posición actual y la posición buscada. En cambio, los ordenadores reales tienen memorias de acceso aleatorio, cada elemento de las mismas puede ser accedido en un único paso utilizando su dirección.

# OBJETIVOS

General:

- Poder conocer el funcionamiento de la máquina de acceso aleatorio e implementar un ejemplo comparándolo con la máquina de Turing.

Específicos:

- Poder realizar un ejercicio donde podamos explicar el funcionamiento de la RAM e implementar el ejemplo de la máquina de Turing.
- Comparar con certeza la máquina de acceso aleatorio y la máquina de Turing.
- Conocer las características principales de la RAM.

# JUSTIFICACIÓN

Suelen existir diversos problemas en el área en base al aprendizaje de máquina y cómo se genera el procedimiento simbólico para plantear una búsqueda de algoritmo que realice una salida deseada y con entradas particulares, ya que su proceso involucra este tipo de problemáticas, el cual reducen un cierto espacio de búsqueda en el procedimiento de algoritmos, ya que un elemento particularmente es adecuado para la solución de dicho problema situado.

una pregunta realmente relevante es, ¿Cómo puede una computadora aprender programas y así mismo resolver esos mismos programas? Su necesidad es automatizar un desarrollo de procedimientos algorítmicos que se han concretado por procedimientos en el desarrollo de la computadora digital. Teniendo en cuenta que su programación automática es analizar los programas de computadora y sintetizar
diferentes programas o modificarlos para la perfección de estos mismos.

Diversos investigadores y computólogos han llevado a cabo año tras año trabajos de programación automática para estimar un éxito en la creación de programas estimados a la ejecución de programas que son modificados de acuerdo a un procedimiento aleatorio de una generación y evaluación. Pues la hipótesis de muchos es que no causó mucha relevancia como se esperaba, ya que sus recursos eran sintetizar ciertos programas para incrementar parcialmente en relación a su tamaño. Sin embargo, todos los intentos fallaron a causa de su explosión combinacional.

# Metodología

La metodología no es más que un conjunto de elementos de tipo racional que se emplean para alcanzar objetivos referentes a una investigación, por ello, al término se le conoce como la metodología de investigación o, en su defecto, como la metodología de un proyecto.

Esta técnica de estudio puede verse en diferentes ámbitos de la vida, de hecho, muchísimas carreras aplican la metodología para realizar cualquier tipo de trabajo, un ejemplo básico de esto es la metodología jurídica.

Cuando se habla de la metodología de investigación, se debe mencionar que existe una disputa referente a las adecuaciones que pueden llegar a realizarse en diferentes métodos de estudio que son objeto de investigaciones previas. Las disputas, con el paso del tiempo, se han multiplicado, más aún en la ciencia humana.

La metodología de un proyecto o las técnicas de estudios en general, representa una de las tantas etapas de los trabajos de investigación en la que es imperativo el uso de posiciones teóricas que, próximamente, se llevarán a la práctica de acuerdo a métodos concretos.

## Características de las metodologías

Las características de la metodología de investigación son amplias, sin embargo, estas tienen una base en diferentes procedimientos, mismos que se emplean una vez que se ha finalizado la investigación para poder conseguir los objetivos del trabajo, para ello, es necesario ejecutar la determinación, hacer uso de un diseño especial para el trabajo investigativo, analizar los fenómenos objeto del análisis y aplicar indicios de medición, elegir, delimitar y describir la población y muestra del proyecto, tomar en cuenta la búsqueda de información y las técnicas para llevar a cabo los instrumentos para recolectar todos los datos necesarios para la investigación, describir todos los procedimientos y, finalmente, tomar las técnicas de cada resultado analizado.

Es importante mencionar que metodología y método no es lo mismo. Las metodologías se usan para analizar cada método ejecutado en el transcurso del proyecto, los métodos son esas técnicas o herramientas empleadas para comenzar un análisis o estudio.

La metodología de la investigación, en este sentido, es también la parte de un proyecto de investigación donde se exponen y describen razonadamente los criterios adoptados en la elección de la metodología, sea esta cuantitativa o cualitativa.

### Metodología cuantitativa

La metodología cuantitativa es aquella empleada por las ciencias naturales o fácticas, que se vale de datos cuantificables a los cuales accede por observación y medición.

Para su análisis, procede mediante la utilización de las estadísticas, la identificación de variables y patrones constantes. Su método de razonamiento es deductivo, para lo cual trabaja con base en una muestra representativa del universo estudiado.

### Metodología cualitativa

La metodología cualitativa es aquella empleada para abordar una investigación dentro del campo de las ciencias sociales y humanísticas.

Como tal, se enfoca en todos aquellos aspectos que no pueden ser cuantificados, es decir, sus resultados no son trasladables a las matemáticas, de modo que se trata de un procedimiento más bien interpretativo, subjetivo, en contraposición con la metodología cuantitativa.

Su método de razonamiento es inductivo: va de lo particular a lo universal. En su caso, se accede a los datos para su análisis e interpretación a través de la observación directa, las entrevistas o los documentos.

## Metodología ágil.

¿Cuál es la metodología ágil?

El enfoque ágil para el desarrollo de software busca distribuir de forma permanente sistemas de software en funcionamiento diseñados con iteraciones rápidas.

Sin embargo, la frase "metodología ágil" es engañosa porque implica que el enfoque ágil es la única forma de abordar el desarrollo de software. La metodología ágil no hace referencia a una serie de indicaciones sobre qué hacer exactamente durante el desarrollo de software. Se trata más bien de una forma de pensar en la colaboración y los flujos de trabajo, y define un conjunto de valores que guían nuestras decisiones con respecto a lo que hacemos y a la manera en que lo hacemos.

En concreto, las metodologías ágiles de desarrollo de software buscan proporcionar en poco tiempo piezas pequeñas de sistemas de software en funcionamiento para mejorar la satisfacción del cliente. Estas metodologías utilizan enfoques flexibles y el trabajo en equipo para ofrecer mejoras constantes.

En respuesta a los enfoques en cascada de la gestión de proyectos, en que estos se organizan como series de secuencias lineales, un grupo de desarrolladores de software redactó el Manifiesto para el Desarrollo Ágil de Software.

De acuerdo con lo que establecieron, los equipos de desarrollo ágil de software deberían valorar:

**Las personas y las interacciones** antes que los procesos y las herramientas

**El software en funcionamiento** antes que la documentación exhaustiva

**La colaboración con el cliente** antes que la negociación contractual

**La respuesta ante el cambio** antes que el apego a un plan

### Marcos ágiles

Los marcos ágiles para el desarrollo de software (como Scrum, kanban o programación extrema [XP]) son la base de procesos conocidos de desarrollo de software, como DevOps y la integración continua/implementación continua (CI/CD).

Scrum es probablemente el marco ágil que más se utiliza en la actualidad. Sin embargo, no es la única alternativa dentro del mundo ágil y, para ser sinceros, no todos sus aspectos pueden caracterizarse como ágiles. Este marco de gestión de trabajos se diseñó para equipos pequeños e interdisciplinarios de entre 5 y 9 personas, los cuales dividen su trabajo en acciones que se puedan completar en un período de tiempo uniforme denominado "sprint".

## Metodologías tradicionales

Las metodologías tradicionales, como su nombre nos indica, son las que se han usado toda la vida. Buscan imponer disciplina al proceso de desarrollo software y de esa forma volverlo predecible y por ello eficiente.

De hecho, estas metodologías tienen un enfoque predictivo, donde se sigue un proceso secuencial en una sola dirección y sin marcha atrás. La estimación/captura de requisitos se realiza una única vez (exacto, una vez solo) al principio del proyecto y es precisamente por eso que nuestra estimación tendrá mucha importancia ya que de ella dependen todos los recursos que emplearemos en el proyecto. Si queremos adoptar una metodología tradicional, el desarrollo de un proyecto debe empezar siempre con un riguroso proceso de captura de requisitos, análisis y diseño. Recuerda: los requisitos son acordados de una vez y, para todo el proyecto, no se esperan cambios en ellos.

Un consejo: adopta este tipo de metodología cuando ya tienes mucha experiencia con un determinado tipo de producto y sabes estimarlo perfectamente. Además, es la opción ideal para proyectos donde los requisitos no cambian y las condiciones del entorno son conocidas y estables.

¿Y qué pasa si tienes que desarrollar un proyecto/producto nuevo? ¿Estás 100% seguro de que todo saldrá exactamente según la estimación y lo previsto?

Un consejo: en este caso, considera aplicar una metodología ágil.

Las metodologías ágiles surgen como alternativa a las tradicionales porque nos ayudan a reducir la probabilidad de fracaso por subestimación de costos, tiempos y funcionalidades en entornos cambiantes.

## Metodologías tradicionales vs agiles.

Tal y como hemos visto, la elección dependerá de los factores que están involucrados en el proyecto: requisitos, equipo, tipo de cliente, entorno, etc.

Un consejo: es recomendable adoptar metodologías tradicionales en proyectos con un problema conocido y una solución al mismo bien definida. En este entorno es fácil analizar, diseñar y ejecutar una solución. Es ideal si sabemos que los requisitos que se establecen al principio del proyecto no van a cambiar. En fin, la metodología tradicional es óptima para proyectos cortos donde el riesgo sea más limitado.

En un entorno muy cambiante donde no está claro el problema a solucionar, ni la forma de hacerlo son más eficaces las metodologías ágiles ya que incorporan mecanismos de gestión del cambio que implican un menor esfuerzo.

# Justificación de la Metodología

ICONIX es una metodología tradicional de desarrollo del software que cuenta con propiedades de otras metodologías conocidas como son RUP y XP, junto con algunas similitudes a metodologías ágiles. Se basa principalmente en el uso del lenguaje UML, pero con la ventaja de ser más ligera, además de proveer más documentación y evitando caer en un estado de paralización del proyecto.

Una característica destacada de ICONIX es su uso del análisis de robustez, consistente en la reducción del paso entre análisis y diseño, permitiendo que los casos de uso sean mucho más fáciles de diseñar y testear.

Está dividida en 4 fases, realizadas en orden, y antes de pasar a la siguiente se analiza y se actualiza el trabajo realizado en la anterior fase siempre que sea necesario:

Fase 1 – **Análisis de requisitos**: dedicada principalmente a la creación de un modelo de dominio junto con la identificación de los casos de uso. También se realizarán diferentes prototipos de interfaces gráficas de usuarios.

Fase 2 – **Análisis y diseño preliminar**: una vez realizados los modelos de casos de usos, podemos empezar a detallar cada uno de ellos. Otro paso será realizar un análisis de robustez para evitar posibles errores en la descripción de los casos de usos, y a su vez actualizar el modelo de dominio.

Fase 3 – **Diseño**: en esta fase nos basaremos en el modelo de casos de usos y el modelo de dominio descritos anteriormente para el diseño de diagramas de secuencia y de clase, respectivamente.

Fase 4 – **Implementación**: fase final donde empezaremos a crear el código a partir de los diagramas de la fase anterior como base.

Uno de los principales problemas de ICONIX es su limitación respecto a proyectos grandes, siendo ineficaz en este sentido. También requiere, al ser en parte una metodología ágil, información rápida y puntual acerca del proyecto.

Para poder sacar el máximo rendimiento a ICONIX es necesario que el equipo de desarrollo tenga altos conocimientos en UML debido a su uso intensivo.

ICONIX

Ventajas

Más ligero que el lenguaje UML y proveyendo más documentación, evitando caer en un estado de paralización del proyecto.

Usa un análisis de robustez que reduce la ambigüedad al describir los casos.

Proporciona suficientes requisitos y documentación de diseño, pero sin parar el análisis.

Es refinado y actualizado a lo largo del proyecto, por lo que facilita la comprensión del problema de espacio.

Desventajas

No puede ser usado para proyectos grandes.

Necesita información rápida y puntual de los requisitos, el diseño y las estimaciones.

Se debe conocer los diagramas UML.

La mayoría de su información está en inglés, con lo que se debe dominar este idioma.

# Detalle de elección.

Una vez estudiadas las metodologías seleccionadas, el siguiente paso será escoger cuál emplear en nuestro proyecto. Basándonos en los requisitos necesarios para cada metodología, hemos optado por usar ICONIX como base nuestro desarrollo. La razón de esta elección se basa en la facilidad de comprensión de su funcionamiento respecto a MSF, además de su bajo coste, adecuado para nuestro equipo de desarrollo.

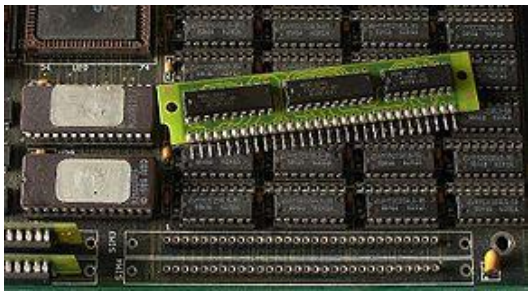
Herramientas útiles para implementar la metodología:

Para conseguir realizar un trabajo eficaz es necesario que nuestro equipo tenga un alto conocimiento del lenguaje UML, ya que es la base en la que se sustenta la estructura de la metodología. Para ello podemos usar “Visual Paradigm” el cual es una herramienta que nos ayudará a crear de manera sencilla y fácil los diagramas UML.

# ANTECEDENTES EN BASE A LA INVESTIGACIÓN

La memoria de acceso aleatorio (Random Access Memory, RAM) se usa como memoria de trabajo de pc's y otros dispositivos para el sistema operativo, los programas y la mayoría del programa. Se llaman «de acceso aleatorio» ya que se puede leer o redactar en una postura de memoria con un periodo de espera igual para cualquier postura, no siendo primordial continuar un orden para entrar (acceso secuencial) a la información de la forma más inmediata viable.

Memoria de acceso aleatorio, imagen 1.0



*Memoria de acceso aleatorio, imagen 1.0*

En la situación que no existan o no se detecten los módulos, la mayoría de tarjetas madres emiten una secuencia de sonidos que indican la falta de memoria primaria, se podría decir, que uno de los primeros tipos de memoria RAM ha sido la memoria de núcleo magnético, hecha entre 1949 y 1952 y utilizada en varios computadores hasta el desarrollo de circuitos incluidos a fines de la década de 1960 a inicios de los 70. Previamente que aquello, las pc's utilizaban relés y líneas de retardo de diversos tipos construidas para llevar a cabo las funcionalidades de memoria primaria con o sin ingreso aleatorio.

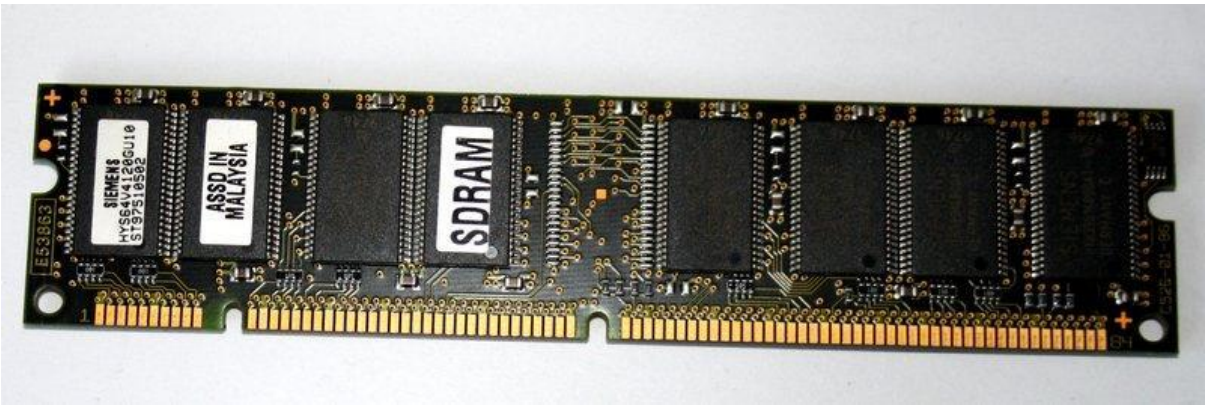
*Antecedentes, imagen 1.1*

En 1969 fueron lanzadas una de las primeras memorias RAM fundamentadas en semiconductores de silicio a causa de Intel con el incluido 3101 de 64 bits de memoria y para el siguiente año se manifestó una memoria DRAM de 1024 bits, alusión 1103 que se constituyó en un hito, debido a que ha sido la primera en ser comercializada exitosamente, lo cual representó el inicio del fin para las memorias de núcleo magnético. Comparativamente con los incluidos de memoria DRAM recientes, la 1103 es primigenia en diversos puntos, sin embargo, poseía un funcionamiento más grande que la memoria de núcleos. Puesto que en 1973 se manifestó una innovación que permitió otra miniaturización y se ha convertido en estándar para las memorias DRAM: la multiplexación en tiempo de las direcciones de memoria. Con la era se logró obvio que la instalación de RAM sobre el impreso primordial, impedía la miniaturización, entonces se idearon los primeros módulos de memoria como el SIPP, aprovechando los resultados positivos de la obra modular.

# MARCO TEÓRICO

## TIPOS DE MEMORIA

1. SRAM (Static Random Access Memory), RAM estática, memoria estática de acceso aleatorio. Volatil y No Volatil




SRAM, figura 2.0

- NV RAM; La memoria SRAM es más cara, pero más rápida y con un menor consumo (especialmente en reposo) que la memoria DRAM. Debido a su compleja estructura interna, es menos densa que DRAM, y por lo tanto no es utilizada cuando es necesaria una alta capacidad de datos, como por ejemplo en la memoria principal de los computadores personales. Por otra parte, las SRAM utilizadas con frecuencia baja, tienen un consumo bastante menor, del orden de micro-vatios. No volátil.
- M RAM; A diferencia de la RAM convencional los datos no se almacenan como carga eléctrica o flujos de corriente, sino por medio de elementos de almacenamiento magnético. Los elementos están formados por dos discos ferromagnéticos, cada uno de los cuales puede mantener un campo magnético, separados por una fina capa de aislante.

  La lectura se realiza midiendo la resistencia eléctrica de la celda. En general, una celda se selecciona con base en la alimentación de un transistor asociado que conduce la corriente desde una línea de alimentación a través de la celda a tierra. La escritura puede realizarse de varias maneras.

2. DRAM (Dynamic Random Access Memory), RAM dinámica, memoria dinámica de acceso aleatorio.

DRAM, figura 2.1

- DRAM Asincrónica (Asynchronous Dynamic Random Access Memory), memoria de acceso aleatorio dinámica asincrónica.
- FPM RAM (Fast Page Mode RAM)
- EDO RAM (Extended Data Output RAM)

3. SDRAM (Synchronous Dynamic Random-Access Memory, memoria de acceso aleatorio dinámica sincrón


SDRAM,figura 2.2

- SDRAM (Synchronous Dynamic Random-Access Memory, memoria de acceso aleatorio dinámica sincrónica). Una de las características más destacable dentro de las RDRAM es que su ancho de palabra es de tan solo 16 bits comparado con los 64 a los que trabajan las SDRAM, y también trabaja a una velocidad mucho mayor, llegando hasta los 400 MHz. Al trabajar en flancos positivos y negativos, se puede decir que puede alcanzar unos 800 MHz virtuales o equivalentes; este conjunto le da un amplio ancho de banda.

  Por eso, a pesar de diseñarse como alternativa a la SDR SDRAM, se convirtió en competidora de la DDR SDRAM. Debido a que La RDRAM es un tipo de memoria síncrona.

- RDRAM (Rambus Dynamic Random-Access Memory). Es una implementación de alto desempeño de las DRAM, el sucesor de las memorias Rambus RDRAM y un competidor oficial de las tecnologías DDR2 SDRAM y GDDR4.

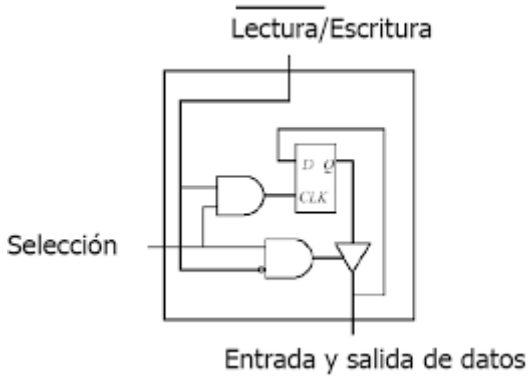
- SDR SDRAM (de las siglas en Inglés Single Data Rate Synchronous Dynamic Random-Access Memory). Las memorias SDR SDRAM son memorias síncronas, con tiempos de acceso de entre 25 y 10 ns y que se presentan en módulos DIMM de 168 contactos.

# RELACIÓN EN EL SISTEMA

Cada paso del algoritmo consiste de (hasta) tres fases:

1. Una fase de LECTURA, durante la cual el procesador lee un dato desde una posición arbitraria de la memoria en uno de sus registros internos
2. Una fase de CÓMPUTO, durante la cual el procesador realiza una operación básica sobre los contenidos de uno o dos de sus registros.

3. Una fase de ESCRITURA, durante la cual el procesador escribe el contenido de un registro en una posición de memoria arbitraria




Arquitectura, figura 2.3

# LA MÁQUINA PARALELA DE ACCESO ALEATORIO

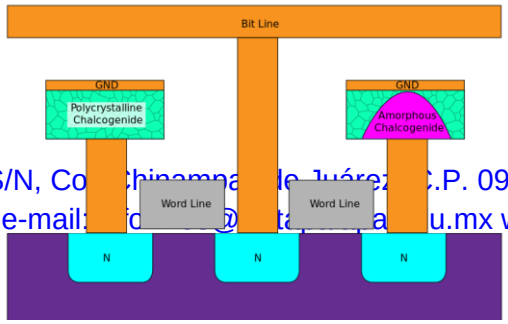
- Un número de procesadores idénticos *P [1], P [2],* ... P[n] (del mismo tipo usado en la RAM). En principio, *N* es ilimitado. Para nuestros propósitos, sin embargo, *N* es un número arbitrariamente grande, pero finito.
- Una memoria común (también del tipo de la usada en la RAM) con *M* posiciones. Nuevamente *M* es en principio ilimitado. Sin embargo, para nuestros propósitos es un número arbitrariamente grande pero finito, tal que M >= N. Los N procesadores comparten esta memoria común.
- Una unidad de acceso a memoria (MAU) que permite a los procesadores obtener acceso a la memoria.

Una aplicación interesante y útil de la PRAM ocurre cuando todos los procesadores ejecutan el mismo algoritmo de forma síncrona. Este modo de operación resulta sencillo para el diseño y análisis de algoritmos eficientes para una multitud de problemas. A continuación, lo consideraremos con mayor detalle.

Cada paso de un algoritmo para la PRAM consiste de (hasta) tres fases:

1. Una fase de LECTURA, durante la cual (hasta) "N" procesadores leen simultáneamente desde(hasta) N posiciones de memoria. Cada procesador lee a lo sumo una posición de memoria y almacena el valor obtenido en un registro local.
2. Una fase de CÁLCULO, durante la cual (hasta) "N" procesadores realizan operaciones aritméticas o lógicas básicas sobre sus datos locales.

3. Una fase de ESCRITURA, durante la cual (hasta) N procesadores escriben simultáneamente en(hasta) N posiciones de memoria. Cada procesador escribe el valor contenido en un registro local en a lo sumo una posición de memoria



PRAM, figura 3.1

En la descripción precedente, la frase “hasta N procesadores” significa que, a través de control algorítmico, puede evitarse que algunos procesadores (si así se desea) ejecuten un paso dado. De hecho, dado que los procesadores están indexados, el algoritmo puede determinar explícitamente cuáles de ellos deben estar activos.

# ACCESO A MEMORIA

Los procesadores tienen varias formas distintas de obtener acceso a la memoria, hecho posible gracias al repertorio de instrucciones de la PRAM. Tales instrucciones para lectura y escritura son las siguientes:

**Lectura Exclusiva (ER, Exclusive Read).** En este modo de acceso a memoria, los procesadores ganan acceso a las posiciones de memoria con el propósito de leer en un esquema “uno a uno”, como se muestra en la Figura 2.3(a). Así, cuando esta instrucción se ejecuta, p procesadores pueden leer simultáneamente el contenido de p posiciones distintas de memoria, donde p <= N, tal que cada uno de los p procesadores involucrados lean exactamente una posición de memoria, y cada una de las posiciones de memoria involucradas sea leída por exactamente un procesador.

**Lectura Concurrente (CR, Concurrent Read).** En este modo de acceso a la memoria, dos o más procesadores pueden leer de la misma posición de memoria al mismo tiempo, como se muestra en la Figura 2.3(b). Así, cuando esta instrucción se ejecuta, p procesadores pueden leer simultáneamente el contenido de p' posiciones de memoria distintas, donde p <= N y p' <= p, tal que cada uno de los procesadores involucrados lea exactamente de una posición de memoria, mientras que cada una de las' posiciones de memoria involucradas pueda ser leída por más de un procesador. Obviamente, CR incluye a ER como un caso especial.

# INSTRUCCIÓN CW

CW específica almacenar en una posición de memoria dada, cuando varios procesadores intenten escribir en ella simultáneamente. Por consiguiente, están disponibles diversas variantes de CW:

- CW CON PRIORIDAD: se le asignan ciertas prioridades a los procesadores. De todos los procesadores intentando escribir en una posición de memoria dada, sólo se le permitirá al que tenga mayor prioridad. El algoritmo determina cómo se asignan dichas prioridades (por ejemplo, la prioridad de cada procesador puede ser igual a su índice), o variar de acuerdo a cierta regla durante la ejecución del algoritmo.
- CW COMÚN: se permite a los procesadores que desean escribir en una posición de memoria que lo hagan, sólo si intentan escribir el mismo valor. En tal caso, un procesador elegido arbitrariamente lo hará. La instrucción se especifica adicionalmente como sigue:
- FALLO COMÚN. Si los valores que van a escribir los procesadores en una posición de memoria no son iguales, se mantiene sin cambios el contenido de dicha posición.
- COLISIÓN COMÚN. Esta instrucción requiere que se almacene una etiqueta de“fallo” en la posición de memoria en caso de que la instrucción no sea exitosa, debido a que dos o más procesadores intentan escribir valores diferentes entre sí.
- PRUEBA DE FALLOS COMÚN. Aquí no se tolera el fallo. El algoritmo debe diseñarse de forma que siempre que más de un procesador quiera escribir en la misma posición de memoria, deba tratar de escribir el mismo valor (de otra forma, la ejecución del algoritmo se detiene).
- CW ARBITRARIA. De todos los procesadores tratando de escribir simultáneamente en una misma posición, cualquiera puede tener éxito sin afectar a la corrección del algoritmo. Sin embargo, dicho algoritmo debe especificar exactamente cómo elegir el procesador que tendrá éxito.
- CW ALEATORIA. Aquí el procesador que tendrá éxito al escribir se elige mediante un proceso aleatorio. Esta instrucción puede usar algoritmos que incluyen un elemento de aleatoriedad en su ejecución.
- CW COMBINATORIA. En este modo de escritura concurrente, todos los valores que un conjunto de procesadores desean escribir simultáneamente en una posición de memoria se combinan en un único valor, el cual entonces se almacena en dicha posición. Esta instrucción retoma formas, dependiendo de la función que requiera usar el algoritmo para combinar los valores (escritos por varios procesadores) antes de almacenar el resultado (en la posición de memoria).

Están disponibles las siguientes variantes:

1. Funciones aritméticas: los valores a escribirse o bien se suman (usando SUMA) o se multiplican (usando PRODUCTO).
2. Funciones lógicas: un conjunto de valores booleanos pueden combinarse usando AND, OR o XOR. También están disponibles las negaciones de estas funciones, es decir, NAND,NOR y NXOR, respectivamente.

3. Funciones de selección: se elige el valor más grande o el más chico usando MAX o MIN, respectivamente. Si varios procesadores almacenan el mayor o el menor, se elige uno especificando posteriormente si debe usarse la función PRIORIDAD, ARBITRARIO o ALEATORIO

# RESULTADOS

Una secuencia de valores S = { S[1], S[2], ..., S[n] }, donde n>=2, así como un dato x, se almacenan en la memoria compartida de la PRAM. Se sabe que x <> S[i] para toda i, 1<=i<=n. Si S[i]<x para toda i, 1<=i<=n, se calcula el mayor de los valores Si].De otro modo, si S[i]>x para toda i, 1<=i<=n, se calcula el menor de los valores. Finalmente, si algunos S[i] son mayores que otros son menores, se calcula el valor promedio de los menores y el valor promedio de los mayores.

El problema puede resolverse con una PRAM de n procesadores P[1], P[2], ..., P[n].Usando ER, P[i] lee S[i] para toda i, 1<=i<=n. Luego usando **CR**, todos los procesadores lee nx. Ahora P[i] compara S[i] con x para toda i, 1<=i<=n. **Usando CW NXOR**, P[i] escribe 1 en una posición **A** de memoria si S[i]<x; de otro modo, escribe 0, 1<=i<=n. Todos los procesadores ahora leen A usando CR. Si vale 1, implica que todos los S[i] son menores que x o que todos los S[i] son mayores que x. En tal caso, todos los procesadores usan ya sea **CW MAX** (si S[i]<x) o **CW MIN** (si S[i]>x) para escribir el mayor o el menor valor de la secuencia S en una posición B de la memoria. Por otra parte, si A contiene un 0, significa que algunos S[i] son menores que x mientras que otros son mayores. Aquellos procesadores para los cuales S[i]<x (S[i]>x) usan **CW SUM** para escribir su S[i] en la posición C[1] (C[2]) de la memoria y un 1 en la posición D[1] (D[2]). Por tanto los promedios deseados los encontrarán los procesadores P [1] y P[2], que calculan G[1]=C[1]/D[1] y G[2]=C[2]/D[2], respectivamente.

El algoritmo puede expresarse como sigue: todos los pasos los realizan los procesadores P [1], P [2], ..., P[n] excepto los pasos 5 y 6, que sólo llevan a cabo los procesadores P[1] y P[2]. Cada paso se implementa usando un número de pasos elementales.

```
Paso 1:
     LEER:          P[i] usa ER para leer S[i] en r0[i]
     CALCULAR:      No se usa
     ESCRIBIR:      No se usa
Paso 2:
     LEER:          P[i] usa CR para leer x en r1[i]
     CALCULAR  :    if r0 < r1 # S[i] < x
                    then r2 := 1
                    else r2 := 0
                    end if
     ESCRIBIR:      P[i] usa CW NXOR para escribir r2[i] en A
Paso 3:
     LEER:          P[i] usa CR para leer A en r2[i]
     CALCULAR:      if r2 = 1 # A = 1
                    then
                         if r0 < r1 # S[i] < x
                         then do (3.1)
                         else do (3.2)
                         end if
                    else
                         if r0 < r1 # S[i] < x
                         then do (3.3) and (4.1)
                         else do (3.4) and (4.2)
                         end if
                    end if
     ESCRIBIR:      (3.1) P[i] usa CW MAX para escribir r0[i] en B
                    (3.2) P[i] usa CW MIN para escribir r0[i] en B
```

resultados, figura 6.1

```
    CALCULAR:      No se usa
    ESCRIBIR:      No se usa
Paso 6:
    LEER:          P[i] usa ER para leer D[i] en r4[i]
    CALCULAR:      r5 = r3 / r4  # g[i] := C[i]/D[i]
    ESCRIBIR:      P[i] usa EW para escribir r5[i] en G[i]
FIN
```

resultados, figura 6.2

Una formulación más suscita del algoritmo se da a continuación bajo el nombre de LÍMITES PRAM. En él, muchos detalles expuestos en la descripción previa (en particular una especificación separada década fase de los pasos, y una identificación de cada forma de acceso a memoria empleada) se han omitido. Tales detalles pueden inferirse fácilmente del contexto.

```
Algoritmo LÍMITES PRAM
```

```
Paso 1:
     for i := 1 to n do in parallel
          (1.1)     if S[i] < x
                    then A :=(NXOR) 1
                    else A :=(NXOR) 0
                    end if
          (1.2)     if A = 1
                    then
                         if S[i] < x
                         then B :=(MAX) S[i]
                         else B :=(MIN) S[i]
                         end if
                    else
                         if S[i] < x
                         then
                              (i)  C[1] :=(SUM) S[i]
                              (ii) D[1] :=(SUM) 1
                         else
                              (i)  C[2] :=(SUM) S[i]
                              (ii) D[2] :=(SUM) 1
                         end if
                    end if
     end for
Paso 2:
     for i := 1, 2 do in parallel
          G[i] := C[i]/D[i]
     end for
END.
```

resultados, figura6.3

En el algoritmo LÍMITES PRAM, la sentencia for i := 1 to n do in paralle significa que todos los procesadores P[i], 1<=i<=n, realizan el Paso 1 simultáneamente: cada procesador P[i] ejecuta primero el Paso (1.1) y luego el Paso (1.2). Similarmente, la sentencia

for i := 1,2 do in parallel

significa que el Paso 2 lo ejecutan sólo P[1] y P[2]. Usaremos esta forma de descripción algorítmica de alto nivel de aquí en adelante. Las instrucciones de acceso a la memoria de la PRAM obedecen las siguientes dos propiedades:

1. Cada una de ellas puede simularse en una RAM en un tiempo proporcional al número de procesadores involucrados.
2. Todas ellas pueden ejecutarse en la **PRAM** en la misma cantidad de tiempo y usando los mismos recursos. Así, la instrucción **ER** por ejemplo, no requiere menos recursos ni menos tiempo que la instrucción **CW**.

Específicamente, la misma **MAU** puede ejecutar todas las instrucciones de **LECTURA** y **ESCRITURA** en la misma cantidad de tiempo.

# Conclusión

Podemos concluir que las memorias RAM tal como las conocemos, por sus siglas en inglés, random access memory, ósea memoria de acceso aleatorio, se compone de circuitos integrados, capaces de almacenar datos de vital importancia para nuestros computadores por retener datos capaces de almacenar datos de vital importancia para nuestros computadores por retener datos de manera temporal para luego acceder y escribir en ellas a velocidades y lo más importante sus capacidades que ahora podemos tener decenas de programas abiertos sin perder ningún dato, pero no siempre ha sido de la manera que es en la actualidad. Las RAM tienen una gran importancia dentro del mundo de las computadoras, debido a que ofrecen unas velocidades altas con poca, ya que evitan que el procesador tenga fallas. Antes que pasara todo esto tuvo que pasar mediante fallas y pruebas, de generación en generación para dar éxito a su perfecta funcionalidad.

# Fuentes de información

[1]Marco teorico- maquina de acceso aleatorio http://catarina.udlap.mx /u_dl_a/tales/documentos/lii/rivera_a_f/capitulo2.pdf

[2]Acceso aleatorio https://es.wikipedia.org /wiki/Acceso_aleatorio#Referencia

[3]Maquinas de turing http://cgosorio.es/Docencia/ALeF/UD23/RAMs.pdf

[4]Maquina de turing https://es.xcv.wiki/wiki/Random-access_machine.

[5] Comparación de las metodologías https://tech.tribalyte.eu/blog-metodologias-tradicional-vs-agil

[6] Metodologías https://www.significados.com/metodologia/

[7] Metodologías https://conceptodefinicion.de/metodologia/

# Cuentas GitHub

- https://github.com/joplin-drix - OrtegaBarron Diana.
- https://github.com/joplin-drix/ReportFinal - repositorio
- https://github.com/iluzioniztha- Vega de Alba Omar Enrique