

Sistemas Operativos
Carlos Hernán Gómez Gómez
CHMAQUINA V2020

Primera fase del programa

Implemente un Programa **que corra sobre un computador y la Web o un dispositivo inteligente (ej. Smartphone)**, que realice una simulación gráfica de un **chcomputador** ficticio de funcionamiento básico. Se deja como opcional que la implementación sea multihilos(en el caso de elegir esta opción podrá acogerse a una bonificación en nota.

El programa debe simular un procesador muy elemental y una memoria principal a través de un vector de hasta 9999 posiciones, las cuales pueden ser variadas al momento de iniciar el programa, se asume por defecto que el chcomputador empieza con 100 posiciones de memoria para facilitar el proceso de pruebas. El programa debe estar en capacidad de leer un conjunto de programas en un seudo lenguaje de máquina que llamaremos **CHMAQUINA** y los cargara en las posiciones disponibles de la citada memoria, leerá una instrucción por cada línea de entrada.

Las primeras posiciones de la memoria estarán reservadas para el núcleo del sistema operativo (kernel), el tamaño de este deberá poderse ingresar al iniciar la corrida del simulador, su valor por defecto es $10 \cdot z + 9$ posiciones (donde z será el último dígito de la cédula del estudiante).

El programa deberá realizar un chequeo de Sintaxis, produciendo un listado de errores si los hay, de lo contrario procederá a la carga definitiva del programa en memoria y quedará listo para ejecución del mismo bajo las reglas de corrida de múltiples programas como se indica más adelante.

En cualquier momento en la ejecución del programa podrá pedírsele al programa mostrar el mapa de memoria (es decir el Vector de memoria y sus posiciones, las variables, lo mismo que el valor del acumulador).

Reglas del CHMAQUINA

Se asumirá que el sistema operativo ocupa las primeras posiciones de la memoria, su contenido para este proyecto no es importante y su tamaño se podrá variar solo al iniciar el ambiente de trabajo.

El programa utilizará un acumulador para registrar los valores de los cálculos y recibirá como nombre reservado **“acumulador”**.

Las posiciones de memoria que almacenen datos tendrán un nombre asociado que iniciara con una letra y máximo tendrá 255 caracteres. Cuando no se especifique lo contrario, la inicialización de variables se asume en cero si es numérico y blanco si es alfanumérico. Estas variables deberán ser creadas antes de ser usada y tendrá un nombre asociado.

Las instrucciones constarán de 2 partes; el código de la operación y el(los) operando(s) dependiendo el tipo de instrucción.

Sistemas Operativos
Carlos Hernán Gómez Gómez
CHMAQUINA V2020

El código de operación corresponde al nemónico del código de operación y éste puede ser:

Sistemas Operativos
Carlos Hernán Gómez Gómez
CHMAQUINA V2020

Operación	Descripción
cargue	Cárguese/copie en el acumulador el valor almacenado en la variable indicada por el operando.
almacene	Guarde/copie el valor que hay en el acumulador a la variable indicada por el operando.
vaya*	Salte a la instrucción que corresponde a la etiqueta indicada por el operando y siga la ejecución a partir de allí.
vayasi*	Salte: <ul style="list-style-type: none"> ➤ Si el valor del acumulador es mayor a de cero a la instrucción que corresponde a la etiqueta indicada por el primer operando. ➤ Si el valor del acumulador es menor a cero a la instrucción que corresponde a la etiqueta indicada por el segundo operando ➤ o Si el acumulador es cero a la siguiente instrucción adyacente a la instrucción vayasi y siga la ejecución a partir de allí.
nueva	Crea una nueva variable cuyo nombre es el especificado en el primer operando, en el segundo operando definirá el tipo de variable (C Cadena/alfanumérico, I Entero, R Real/decimal, L lógico o booleano (1 Verdadero o 0 Falso), un tercer operando establecerá un valor de inicialización; a cada variable se le asignará automáticamente una posición en la memoria. Las variables deberán estar definidas antes de ser utilizadas. Las variables no inicializadas tendrán por defecto el valor cero para reales y enteros, espacio para cadenas, 0 para lógicos. El separador de decimales es el punto.
etiqueta	La etiqueta es un nombre que opcionalmente se le puede asignar a una instrucción en el programa para evitar trabajar con las posiciones en memoria de las instrucciones y poder utilizar un nombre simbólico independiente de su ubicación. Crea una nueva etiqueta cuyo nombre es el especificado en el primer operando y a la cual le asignará automáticamente la posición indicada en el segundo operando (esta será la posición relativa de la instrucción a la que se le asigna este nombre con respecto a la primera instrucción del programa). Las instrucciones que definen etiquetas podrán definirse en cualquier posición del programa, pero en todo caso antes de la instrucción retorne.
lea	Lee por teclado el valor a ser asignado a la variable indicado por el operando
sume	Incrementa el valor del acumulador en el valor indicado por la variable señalada por el operando.
reste	Decrementa el acumulador en el valor indicado por la variable que señala el operando.
multiplique	Multiplique el valor del acumulador por el valor indicado por la variable señalada por el operando.
divida	Divida el valor del acumulador por el valor indicado por la variable señalada por el operando. El divisor deberá ser una cantidad diferente de cero.
potencia	Eleve el acumulador a la potencia señalada por el operando (los exponentes pueden ser valores enteros, positivos o negativos)
modulo	Obtenga el modulo al dividir el valor del acumulador por el valor indicado por la variable señalada por el operando.

Sistemas Operativos
Carlos Hernán Gómez Gómez
CHMAQUINA V2020

concatene	Genere una cadena que una la cadena dada por el operando a la cadena que hay en el acumulador (Operando alfanumérico)
elimine	Genere una subcadena que elimine cualquier aparición del conjunto de caracteres dados por el operando de la cadena que se encuentra en el acumulador (operando alfanumérico)
Extraiga	Genere una subcadena que extraiga los primeros caracteres (dados por el valor numérico operando) de la cadena que se encuentra en el acumulador (operando numérico)
Y	Produce una operación lógica Y (AND) entre el primer operando y el segundo operando que son variables lógicas y la almacena en el tercer operando.
O	Produce una operación lógica O (OR) entre el primer operando y el segundo operando que son variables lógicas y la almacena en el tercer operando.
NO	Produce una operación de negación lógica para el primer operando que es una variable lógica y el resultado se almacena en el segundo operando.
muestre	Presente por pantalla el valor que hay en la variable indicada por el operando, si el operando es acumulador muestre el valor del acumulador.
imprima	Lo mismo que el anterior pero presentándolo en la impresora.
XXX	Esta será una función complementaria a las anteriores creada por el estudiante, se deja a voluntad su nombre y forma de funcionamiento.
retorne	El programa termina; debe ser la última instrucción del programa y tiene opcionalmente un operando numérico entero

La ejecución de los programas normalmente se hace de forma secuencial de instrucciones, la primera después la segunda, la tercera....etc, las instrucciones de transferencia de control (vaya y vayasi) son la forma de cambiar este orden de ejecución, obligando que el programa no siga en el orden secuencial predeterminado, sino que continúe en la instrucción señalada por una etiqueta (es decir una instrucción que tiene asignado un nombre como referencia). Vaya y vayasi cumple esta función, la primera de forma incondicional y la segunda condicionada al valor del acumulador como se especifica en su definición.

La inicialización de posiciones de memoria se hará como instrucciones en las cuales se crean las variables y se les asigna valor, como se explicó en la operación Nueva.

El código puede tener comentarios por líneas, los cuales se denotarán por dos backslash (//) en las dos primeras posiciones de la instrucción, de igual manera se podrán insertar líneas en blanco entre instrucciones del programa, cuyo propósito es de legibilidad del programa.

Se podrán realizar operaciones entre valores enteros y reales, los resultados intermedios se manejarán como reales y el resultado final obedecerá al tipo de variable que almacena el resultado.

El programa no debe permitir la sobrecarga del acumulador (Overflow/desborde) por lo cual sacará un mensaje de error que le permita al usuario tomar la decisión que corresponda. Inicialmente la protección de memoria se hará por registro base y registro límite, esto es, cada programa empieza en una posición de memoria (registro base) y termina en otra posición de memoria denominada (registro límite) con base en las cuales se evitará la violación de las

Sistemas Operativos
Carlos Hernán Gómez Gómez
CHMAQUINA V2020

normas básicas de ejecución, también debe tenerse claro que los programas tendrán área de código y área de datos.

El programa podrá ejecutarse en modalidad normal (corrida continua) o paso a paso (instrucción por instrucción), en todo caso se podrá visualizar la instrucción que se esté ejecutando en cada momento y el respectivo valor del acumulador.

Los chprogramas serán almacenados previamente en archivos con extensión ch en cualquier carpeta de algún medio de almacenamiento, de allí podrán ser cargados al señalarlos de la lista.

Deberá proveerse al ambiente de desarrollo con un editor de programas chmaquina, en el cual se puedan crear chprogramas y/o modificarlos, este editor hará el chequeo de sintaxis básico indicándole al usuario las instrucciones con errores y podrá sobre el mismo permitir la corrida del programa si no presenta errores.

Para prueba del programa pueden plantearse diversos **CHMAQUINA** que irán aumentando en complejidad; también podrá aumentarse el grupo de instrucciones y la capacidad de memoria de Nuestro Computador ficticio.

Opcionalmente el programa deberá correr en versión multihilos con todos los controles de concurrencia y administración de ejecución que ello implique.

Se podrán cargar y correr varios chprogramas hasta agotar la memoria disponible, para la corrida de los chprogramas en la primera fase se utilizará una cola circular, la cual será visitada con base al orden de llegada (cola simple-primero en entrar primero en ser atendido-FCFS).

El sistema debe indicar por medio de alguna convención si está trabajando en modo usuario (ejecución del programa) o modo kernel (el sistema tiene el control y administración del ambiente), mostrando la acción de cambio de contexto (el paso de un modo al otro). Se podrán ver los distintos estados en los cuales estén los procesos, a nivel de proceso y a nivel de cola.

Debe presentarse el manual técnico y de usuario, código fuente, código ejecutable, sustentación completa, en cada una de las fases.

Semanalmente debe registrar la bitácora de avance en el blog de seguimiento del estudiante, donde se informará en forma concisa de las tareas realizadas y a realizar en el proyecto. En el asunto del mensaje se colocara: "Bitácora de avance chmaquina AA/MM/DD inicial a AA/MM/DD final". La bitácora deberá incluir el código fuente elaborado del proyecto hasta ese momento.

En la primera bitácora se presentará un cronograma tentativo de desarrollo del proyecto en el cual se incluirán actividades (como estudio preliminar, planeación, diseño, desarrollo, control de calidad, documentación). Deberá entregarse un documento con la tabla de contenido tentativa de la documentación del proyecto. Cada semana debe mostrarse el avance en el cronograma y en la documentación.

Sistemas Operativos
Carlos Hernán Gómez Gómez
CHMAQUINA V2020

La no presentación de bitácora generará un castigo en la nota del 10% por cada semana de no presentación. Estas condiciones rigen para todas las fases.

Toda la documentación y avances del proyecto se irán registrando en el blog creado por el estudiante.

Hasta el punto anterior se considera la primera fase del programa.

En los primeros 15 días del proyecto deberá presentarse una versión del prototipo funcional de la interface gráfica del sistema que se constituirá como la fase 0.

Segunda Fase

Para abordar la segunda fase se debe garantizar que la primera fase esté funcionando adecuadamente, además deberán realizarse o preverse los ajustes necesarios para culminar con esta fase.

En esta fase se podrán variar los algoritmos de planificación de procesos y administración de memoria.

- ◆ Para la planificación de procesos y administración de memoria asuma :
 - Que el conteo de las instrucciones de entrada y salida de las ráfagas de I/O, el conteo de las demás instrucciones excluyendo las declarativas de las ráfagas de CPU)
 - El tiempo de llegada de un programa es igual al tiempo de llegada del programa anterior más el número de instrucciones de tal programa sobre 4, se asume que el primer programa llega en $t=0$.
 - Se debe permitir la carga de programas en cualquier momento, en el proceso de carga de programas se supone que la CPU está suspendida en modo kernel (no hay actividad de los contadores del sistema)
 - Las prioridades se definirán como un número entero entre 0 y 100, siendo la más alta la de mayor valor.
 - El cambio de algoritmo de planificación y/o administración de memoria solo se podrá realizar cuando no se estén ejecutando programas.
- ◆ Se debe permitir elegir el método de planificación de procesos al iniciar el chmaquina, de acuerdo con él se pedirán los parámetros adicionales necesarios a cada proceso, por defecto se asumirá el método round-robin, a cada proceso se le asignarán n unidades de tiempo (slice), este parámetro se podrá modificar al cargar el ambiente de trabajo, tendrá un valor por defecto de 5 (para efectos de este programa se asumirá que cada instrucción de entrada y salida consume de forma aleatoria de 1 a 9 unidades de tiempo, las demás instrucciones consumen una unidad de tiempo),
- ◆ Los otros métodos a tener en cuenta son: RR con prioridades, SJF expropiativo y no expropiativo, FCFS y por prioridad (expropiativo y no expropiativo).
- ◆ Para evitar la inanición en los algoritmos que corresponda se debe usar compensación por envejecimiento (esta implementación se deja a criterio del desarrollador), el slice por defecto se especificó más arriba.
- ◆ El sistema permitirá seleccionar el método de administración de memoria: múltiples particiones fijas (permitiendo elegir el tamaño de las particiones al seleccionar esta opción), particiones variables con condensación, almacenamiento virtual con paginación de un solo nivel (tome por defecto el área de intercambio igual al doble de la memoria principal, pero permita que el usuario pueda cambiar su tamaño al momento de seleccionar esta opción), asuma un tamaño de página de 5 unidades de

Sistemas Operativos
Carlos Hernán Gómez Gómez
CHMAQUINA V2020

memoria que podrán cambiarse al momento de elegir este método, asuma un algoritmo de reemplazo de páginas de reloj/segunda oportunidad, asuma que el conjunto de trabajo es la página donde está el contador de programa (PC) y la siguiente página.

- ◆ El sistema debe permitir ver la tabla de procesos (con la información relevante), las distintas colas, el mapa de memoria principal y de intercambio (si aplica)

Debe presentarse el manual técnico y de usuario, código fuente, código ejecutable, sustentación completa.

Semanalmente debe enviarse una bitácora de avance al profesor, donde se informará en forma concisa de las tareas realizadas y proyectadas.

A continuación relaciono algunos chprogramas de ejemplo para su análisis y prueba de escritorio, por correo se les enviarán estos y otros programas:

Programa Factorial: Calcula el factorial de 5

```
// Programa para calcular el factorial de 5
nueva unidad | 1
nueva m | 5
nueva respuesta | 1
nueva intermedia | 0
cargue m
almacene respuesta
reste unidad
// Se inicia el ciclo de cálculo del factorial
almacene intermedia
cargue respuesta
multiplique intermedia
almacene respuesta
cargue intermedia
reste unidad
vayasi ciclo fin
etiqueta ciclo 9
etiqueta fin 21
muestre respuesta
imprima respuesta
retorne 0
```

Programa Multipliquevar: Calcula el producto de dos valores entrados por teclado.

```
// Calcula el producto de dos valores entrados por teclado.
nueva unidad | 1
nueva m | 0
nueva n | 0
lea m
lea n
nueva respuesta | 0
```

Sistemas Operativos
Carlos Hernán Gómez Gómez
CHMAQUINA V2020

```
nueva resultado C Resultado=  
nueva intermedia I 0  
cargue n  
// inicia ciclo  
reste unidad  
almacene intermedia  
cargue respuesta  
sume m  
almacene respuesta  
cargue intermedia  
vayasi ciclo fin  
etiqueta ciclo 11  
etiqueta fin 25  
muestre resultado  
muestre respuesta  
imprima resultado  
imprima respuesta  
retorne
```