Ejercicios CF 28nov18

En el reporte, por cada ejercicio, muestre como resultado el despliegue de la ventana Console (cmd.exe) y explique su contenido. Entre su despliegue y lo que usted responda, deberá visualizarse la correcta justificación. Incluya en su respuesta todos los valores que se necesiten.

En cada ejercicio también despliegue la parte importante del programa en lenguaje ensamblador.

--------------------------------------------------------------------------------------- --

Los datos y resultados deben ser precedidos por un texto adecuado.

Lea siempre, todo el ejercicio completo antes de empezar a resolverlo.

1. Haga un programa ensamblador, que evalúe una expresión aritmética en notación *postfix*, donde las variables y constantes son todos números reales REAL8.

Por ejemplo si tenemos la siguiente expresión aritmética en notación *infix* **RA \* 6 + SC**, su correspondiente expresión en notación *postfix* es **RA 6 \* SC +**.

Por ejemplo si tenemos la siguiente expresión aritmética en notación *infix* **RA + 6 \* SC**, su correspondiente expresión en notación *postfix* es **RA 6 SC \* +**.

Para convertir expresiones de infix a postfix siempre se toma en cuenta el orden jerárquico de los operadores.

Para ver un ejemplo de ejecución de una expresión aritmética en notación *postfix*, usando el FPU x86 revise la sección 12.2.1 FPU Register Stack, de la séptima edición del libro de Irvine.

Convierta bajo un proceso manual la siguiente expresión *infix* (RA + SB)/TC \* UD – RA + VE a notación *postfix*.

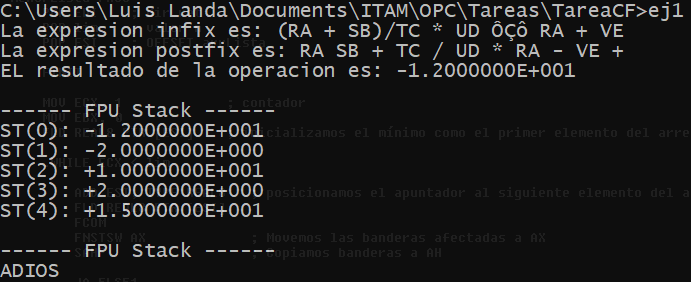
Una vez teniendo escrita la expresión en notación *postfix*, programe y evalúe en ensamblador dicha expresión *postfix*.

Suponga los siguientes valores para las variables REAL8: RA con 2.0, SB con 3.0, TC con -2.0, UD con 10.0 y VE con 15.0.

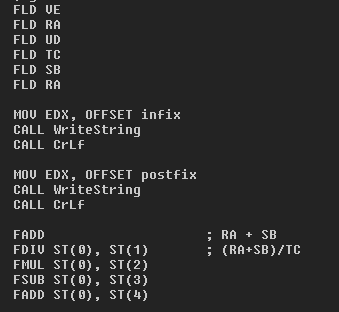
Imprima en un mismo renglón un texto adecuado, antecediendo al resultado numérico real.

Al final imprimirá ADIOS.

El FPU Stack deberá quedar vacio, debiendo desplegar su contenido antes de ADIOS.



Podemos observar que el resultado de la operación se ubica en ST(0) y que al terminar la ejecución, el *stack* queda limpio.



Como la conversión infija a postfija se realizó a mano, lo importante de este ejercicio es la forma en que se manejó el *stack*, en particular el orden con el que se ingresaron los valores. Las operaciones se realizaron como si se estuviera evaluando la operación postfija como se evidencia en la foto superior.

1. En primer lugar, deberá leer un valor entero que llamaremos “factor”, que es un entero entre 0 y 9. Después deberá leer una lista de *n* enteros (1 a 10), con los textos de pedido de datos reales REAL8 (punto flotante) correspondientes, dejando los valores leidos en *arrLista*.

Guarde tanto “n” como “factor” en variables, así como el arreglo *arrLista*, dentro del segmento .DATA. Considere los enteros como DWORD. Primero el procedimiento “LeerLista” se encargará de leer cada uno de los valores reales, con textos e índices adecuados.

Después en un procedimiento llamado “FacLista” deberá multiplicar cada uno de los *n* enteros por el “factor”, dejando los resultados en la misma *arrLista*.

Luego un procedimiento “MenorLista” calculará y regresará, tanto el menor de los valores reales como su índice en el arreglo de reales; El valor menor lo regresará por el FPU Stack y el índice por el stack del programa.

Al final un procedimiento “Imprime” se encargará de imprimir los elementos reales de *arrLista*, con textos e índices adecuados.

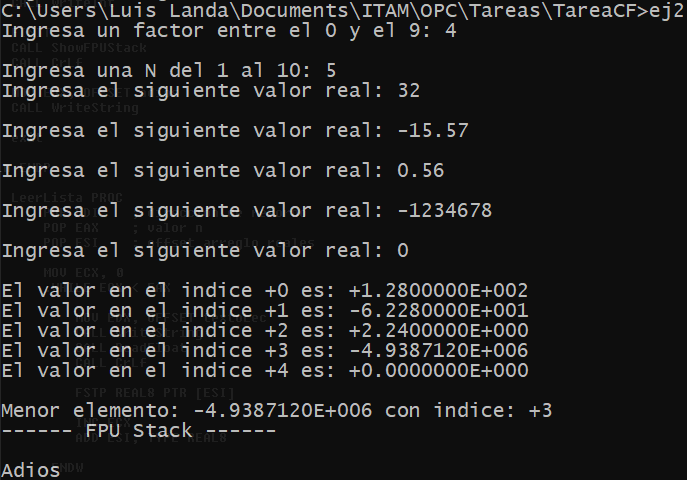
El pasaje de parámetros deberá hacerlo al estilo de los lenguajes de alto nivel, empleando el stack del programa, considerando como argumentos tanto a la “n”, a “factor” como a la dirección del arreglo (OFFSET arrList) donde están los reales.

Tiene disponibles los registros del FPU Stack.

|  |  |
| --- | --- |
| **main PROC**  ; pide el dato ***factor***  ; pide el dato ***n***  CALL LeerLista (n, arrLista)  . . .  CALL FacLista (n, factor, arrLista)  . . .CALL MenorLista (n, arrLista)  . . .CALL Imprime (n, arrLista)  ; imprime “Adios”  . . .  exit  **main ENDP**  **LeerLista PROC(m, arr)**  ; Lee los **m** valores reales y los deja en **arr**  ret  **LeerLista ENDP** | **FacLista PROC(m, f , arr)**  ; Multiplicar los **m** valores reales por **f** y  ; dejarlos en **arr**  ret  **FacLista ENDP**  **MenorLista PROC(m, arr)**  ; Calcule y regrese el menor de los valores  ; reales y su índice.  ; El menor, lo regresara por el FPU Stack.  ; El índice, por el stack del programa.  ret  **MenorLista ENDP**  **Imprime PROC(m, arr)**  ; Imprime los **m** valores reales  ret  **Imprime ENDP**  END main |

Las macro directivas estructuradas que usamos anteriomente, sólo sirven para comparar enteros. Para el caso del procedimiento MenorLista necesitará comparar reales, por lo tanto revise el capítulo 12 de su libro para aprender a implementar las estructuras algorítmicas.

El FPU Stack deberá quedar vacio, debiendo desplegar su contenido antes de ADIOS.



Observamos que se despliegan los textos adecuados, pidiendo al usuario el factor y la cantidad de elementos a leer. Se leen dichos elementos en orden y posteriormente se muestra el arreglo multiplicado por el factor al igual que el menor elemento y su posición en el arreglo. Por último se muestra que el *stack* queda vacío y se despide el programa.



El procedimiento de mayor dificultad fue el de encontrar el mínimo elemento puesto que en los otros procedimientos solo se debía recorrer el arreglo. Vemos que fue necesario utilizar un salto condicional parecido a un *if-else* para ejecutar las comparaciones y asignaciones relevantes. Para comparar los elementos del arreglo, se cargaban en el *stack* y se usaba FCOM sin parámetros para comparar ST(0) y ST(1). El objetivo es que el mínimo siempre se encuentre en ST(0) por lo que si en la comparación anterior el mínimo era ST(1) se hacía un pop. El índice del mínimo se almacenaba en EBX.

**OBSERVACIONES:**

* La respuesta a esta tarea deberá subirla a Comunidad, a la sección de TRABAJOS Y EXAMENES, a más tardar este miércoles 04 de diciembre, antes de las 23:30 hs.
* La primera página de este reporte deberá contener la portada que hemos manejado.
* El nombre del archivo zip donde usted pondrá el archivo respuesta, deberá llamarse con el nombre del grupo de trabajo seguido del nombre de este ejercicio, p.e. **“SparkEjerciciosBI.zip”,** donde “Spark” sería el nombre de su grupo de trabajo.
* También deberá incluir los programas fuente (archivos .asm).