75.03 / 95.57 Organización del Computador

Trabajo práctico Nro. 1

Secuencia de operaciones lógicas (I)

Se dispone de un archivo que contiene registros con el siguiente contenido:

- Operando: secuencia de caracteres que simbolizan bits (caracteres 0 o 1) (16 bytes)
- Operación: carácter que simboliza una operación a efectuar entre dos operandos, y puede ser O (Or), X (Xor) y N (And) (1 byte)

Nota: Tenga en cuenta que los registros son secuencias de bytes contiguos en la memoria, sin saltos de línea. Por
ejemplo, el contenido de un archivo con tres registros podría ser:
1111111111111111X0000111100001111N0000000000001111O

Por otro lado se lee desde teclado un valor de Operando inicial con el mismo formato que el del campo de los registros del archivo (caracteres 0 ó 1) (16 bytes). Se pide realizar un programa en assembler Intel
80x86 que vaya informando por pantalla el resultado de aplicar a los operandos la secuencia de operaciones informadas en los registros del archivo de entrada. La aplicación de las operaciones se hará
de la siguiente manera:
Op. Ini Operac. Reg.1 Op. Reg 1 -> Res. Parcial Operac. Reg. 2 Op. Reg 2 -> Res. Parcial …

Ejemplo:
Operando Inicial = 0000000000000101
Operando Registro 1 = 1111111111111111
Operación Registro 1 = X (Xor)

- Resultado Parcial = 1111111111111010

Operando Registro 2 = 0000111100001111
Operación Registro 2 = N (And)

- Resultado Parcial = 0000111100001010

…

Trabajo práctico Nro. 2

Sum Digits

Se pide desarrollar un programa en assembler Intel 80x86 que solicite el ingreso de un número por teclado en base 2, 4, 8, 10 o 16 y permita:

A. Realizar la conversión entre las bases indicadas.

B. Calcular la sumatoria de los dígitos desde 1_{10} hasta el número N_{10} mostrando el resultado por la pantalla. (Nota: esta sumatoria debe realizarse considerando el número en base 10)
Ej.: Si el número ingresado fuese D_{16} ;primero se debe pasar a base 10 y luego realizar la sumatoria de los dígitos desde 1_{10} hasta 14_{10}:
$[1 + 2 + 3 + 4 + 5 + 6 + 7 + 8 + 9 + (1) + (0) + (1) + (1) + (1) + (2) + (1) + (3) + (1) + (4)]_{10} = 60_{10}$

Trabajo práctico Nro. 3

Ordenamiento por método de Selección

Dado un archivo que contiene n números en BPF c/signo de 8 bits (n <= 30) se pide codificar en assembler Intel 80x86 un programa que imprima por pantalla los movimientos que se realizan (por ejemplo “Realizando el intercambio de valores”) y el contenido de dicho archivo ordenado en forma ascendente o descendente de acuerdo a lo que elija el usuario, usando un algoritmo de ordenamiento basado en el método de selección.

```
procedure seleccion(int[] vector)
	int i, j, k, p, buffer, limit = vector.length-1;
	for(k = 0; k < limit; k++)
		p = k;
		for(i = k+1; i <= limit; i++)
			if(vector [i] < vector [p])
				p = i;
			if(p != k)
				buffer = vector [p];
				vector [p] = vector [k];
				vector [k] = buffer;
			end if
		end for
	end for
end procedure
```

El método de ordenamiento por selección funciona seleccionando el menor elemento del vector y llevándolo al principio; a continuación selecciona el siguiente menor y lo pone en la segunda posición del vector y así sucesivamente.

Nota: no es correcto generar el archivo con un editor de textos de forma tal que cada registro sea una tira de 16 caracteres 1 y 0. Se aconseja el uso de un editor hexadecimal.

Trabajo práctico Nro. 4

Manejo de Matrices (I)

Se pide desarrollar un programa en assembler Intel 80x86 que cargue desde teclado una cantidad X de matrices (con X <= 5) de NxM elementos (con N y M <= 8) y luego permita realizar las siguientes operaciones entre ellas:

- Suma de 2 o más matrices.
- Producto de 2 matrices.
- Hallar la traspuesta de 1 matriz.
- Modificar o consultar el valor de un elemento (M_{ij}) de una matriz.

Nota: Los elementos que componen a cada matriz (M_{ij}) son números enteros entre -99 y 99

Trabajo práctico Nro. 5

Interpretación de BPFlotante IEEE 754

Se pide desarrollar un programa en assembler Intel 80x86 que permita:

- Ingresar configuraciones binaria o hexadecimal de números almacenados en formato IEEE 754 de precisión simple e imprima su notación científica normalizada en base 2 (Ej. + 1,110101 x 10^{101}).

- Ingresar notaciones científicas normalizadas en base 2 y visualicemos su configuración binaria o hexadecimal del dicho número almacenado en formato IEEE 754 de precisión simple.

Trabajo práctico Nro. 6

Bolsas de Correo (I)

Se tienen n objetos de pesos P1, P2, ..., Pn (con n <= 20) que deben ser enviados por correo a una misma dirección. La forma más simple sería ponerlos todos en un mismo paquete; sin embargo, el correo no acepta que los paquetes tengan más de 11 Kg. y la suma de los pesos podría ser mayor que eso. Afortunadamente, cada uno de los objetos no pesa más de 11 Kg.

Se trata entonces de pensar un algoritmo que de un método para armar los paquetes, tratando de optimizar su cantidad. Debe escribir un programa en assembler Intel 80x86 que:

- Permita la entrada de un entero positivo n.
- La entrada de los n pesos, verificando que $0<P_i<=11$ donde i <=n.
- Los P_i pueden ser valores enteros.
- Exhiba en pantalla la forma en que los objetos deben ser dispuestos en los paquetes.

A su vez existen tres destinos posibles: Mar del Plata, Bariloche y Posadas. El correo por normas internas de funcionamiento no puede poner en el mismo paquete objetos que vayan a distinto destino.

Desarrollar un algoritmo que proporcione una forma de acomodar los paquetes de forma que no haya objetos de distinto destino en un mismo paquete y cumpliendo las restricciones de peso. Se sugiere tener una salida como la siguiente:

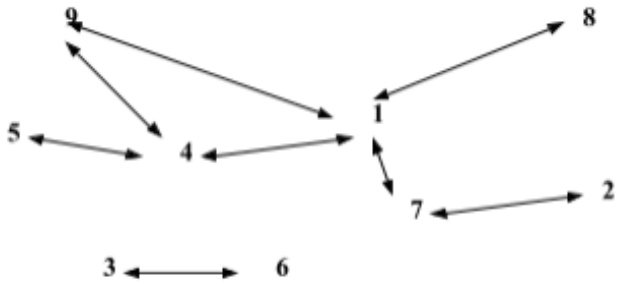
- $Destino_i$ – $Objeto_1$ (P_1) + $Objeto_2$ (P_2)+ ….. + $Objeto_n$ (P_n)
- $Destino_i$ – $Objeto_1$ (P_1) + $Objeto_2$ (P_2)+ ….. + $Objeto_n$ (P_n)
- $Destino_i$ – $Objeto_1$ (P_1) + $Objeto_2$ (P_2)+ ….. + $Objeto_n$ (P_n)

Trabajo práctico Nro. 7

Líneas ferroviarias Light

Se tiene una lista de estaciones de ferrocarril. A efectos de simplificar, los nombres de las estaciones han sido reemplazados por 1, 2, ..., n-1, n donde n <= 20 es el número total de estaciones. Se tiene, además, una lista de los tramos de vía que unen, directamente entre sí, estaciones contiguas. Los tramos de vía pueden utilizarse en ambos sentidos.

Por ejemplo, si n = 9 y los tramos son (8,1), (3,6), (4,9), (1,7), (2,7), (1,4), (5,4) y (1,9) esto podría representarse gráficamente como:



Escribir un programa en assembler Intel 80x86 que lea n y la lista de tramos. Luego, agrupar las estaciones en líneas, donde una línea se define como un conjunto de estaciones alcanzables entre sí, y listarlas. En el ejemplo dado, se produciría un resultado semejante al siguiente:

Línea 1: 1 2 4 5 7 8 9
Línea 2: 3 6

Trabajo práctico Nro. 8

Bolsas de Correo (II)

Se tienen n objetos de pesos P1, P2, ..., Pn (con n <= 20) que deben ser enviados por correo a una misma dirección. La forma más simple sería ponerlos todos en un mismo paquete; sin embargo, el correo no acepta que los paquetes tengan más de 15 Kg. y la suma de los pesos podría ser mayor que eso. Afortunadamente, cada uno de los objetos no pesa más de 15 Kg.

Se trata entonces de pensar un algoritmo que de un método para armar los paquetes, tratando de optimizar su cantidad. Debe escribir un programa en assembler Intel 80x86 que:

- Permita la entrada de un entero positivo n.
- La entrada de los n pesos, verificando que $0<P_i<=15$ donde i <=n.
- Los P_i pueden ser valores enteros.
- Exhiba en pantalla la forma en que los objetos deben ser dispuestos en los paquetes.

A su vez existen tres destinos posibles: Posadas, Salta y Tierra del Fuego. El correo por normas internas de funcionamiento no puede poner en el mismo paquete objetos que vayan a distinto destino.

Desarrollar un algoritmo que proporcione una forma de acomodar los paquetes de forma que no haya objetos de distinto destino en un mismo paquete y cumpliendo las restricciones de peso. Se sugiere tener una salida como la siguiente:

- $Destino_i$ – $Objeto_1$ (P_1) + $Objeto_2$ (P_2)+ ….. + $Objeto_n$ (P_n)
- $Destino_i$ – $Objeto_1$ (P_1) + $Objeto_2$ (P_2)+ ….. + $Objeto_n$ (P_n)
- $Destino_i$ – $Objeto_1$ (P_1) + $Objeto_2$ (P_2)+ ….. + $Objeto_n$ (P_n)

Trabajo práctico Nro. 9

Secuencia de operaciones lógicas (II)

Se dispone de un archivo que contiene registros con el siguiente contenido:

- Operando: secuencia de caracteres que simbolizan bits (caracteres 0 o 1) (16 bytes)
- Operación: carácter que simboliza una operación a efectuar entre dos operandos, y puede ser O (Or), X (Xor) y N (And) (1 byte)

Nota: Tenga en cuenta que los registros son secuencias de bytes contiguos en la memoria, sin saltos de línea. Por ejemplo, el contenido de un archivo con tres registros podría ser:
1111111111111111X0000111100001111N0000000000001111O

Por otro lado se lee desde teclado un valor de Operando inicial con el mismo formato que el del campo de los registros del archivo (caracteres 0 ó 1) (16 bytes). Se pide realizar un programa en assembler Intel 80x86 que vaya informando por pantalla el resultado de aplicar a los operandos la secuencia de operaciones informadas en los registros del archivo de entrada. La aplicación de las operaciones se hará de la siguiente manera:

Op. Ini Operac. Reg.1 Op. Reg 1 -> Res. Parcial Operac. Reg. 2 Op. Reg 2 -> Res. Parcial …

Ejemplo:
Operando Inicial = 0000000000000101
Operando Registro 1 = 1111111111111111
Operación Registro 1 = X (Xor)

- Resultado Parcial = 1111111111111010

Operando Registro 2 = 0000111100001111
Operación Registro 2 = N (And)

- Resultado Parcial = 0000111100001010

…

Trabajo práctico Nro. 10

Manejo de conjuntos (I)

Desarrollar un programa en assembler Intel 80x86 que permita definir n conjuntos (con n <= 6) cuyos elementos tienen una longitud de 1 a 2 caracteres alfanuméricos (A..Z , 0..9).

El programa deberá responder las siguientes consultas:

- Pertenencia de un elemento a un conjunto.
- Intersección entre conjuntos.
- Diferencia entre conjuntos.
- Producto cartesiano entre conjunto.

Tener en cuenta el siguiente ejemplo:

Ca = {0, 1, 3}
Cb = {0, 5, 3}

Diferencia: Ca-Cb = {1}

Intersección: Ca ∩ Cb = {0, 3}

Prod. Cartesiano: Ca X Cb = {(0,0), (0,5), (0,3), (1,0), (1,5), (1,3), (3,0), (3,5), (3,3)}

Nota: El tamaño máximo de cada conjunto será de 20 elementos. Tener en cuenta que el orden de los caracteres en un elemento lo diferencia de otro conformado por los mismos caracteres, siendo así que el elemento AZ es distinto que el elemento ZA y que la unión de dos conjuntos puede resultar en un conjunto con más de 20 elementos o en un conjunto vacío.

Trabajo práctico Nro. 11

Protección de información - Cifrado Playfair

La protección de información consiste en convertir un mensaje original en otro de forma tal que éste sólo pueda ser recuperado por un grupo de personas a partir del mensaje codificado.
El sistema para llevar a cabo la protección deberá basarse en el álgebra lineal, con las siguientes pautas:
- Alfabeto a utilizar 25 caracteres (A .. Z, omitiendo la J).
- Las letras son distribuidas en una matriz de 5x5.
- El mensaje a codificar deberá ser dividido en bloques de a dos caracteres (validando que ninguno de los bloques contenga dos ocurrencias de la misma letra y la 'J').

La conversión se llevará a cabo por bloques, es decir tomando dos caracteres del mensaje por vez.

- Si los caracteres se encuentran en distinta fila y columna de la matriz, considerar un rectángulo formado con los caracteres como vértices y tomar de la misma fila la esquina opuesta.
- Si los caracteres se encuentran en la misma fila, de cada caracter el situado a la derecha.
- Si los caracteres se encuentran en la misma columna, tomar el caracter situado debajo.

Se pide desarrollar un programa en assembler Intel 80x86 que permita proteger información de la manera antes descripta.
El mismo va a recibir como entrada:

- El mensaje a codificar o codificado.
- La matriz de 5x5.

El mismo va a dejar como salida:

- El mensaje codificado u original.

Trabajo práctico Nro. 12

Manejo de Conjuntos (II)

Desarrollar un programa en assembler Intel 80x86 que permita definir n conjuntos (con n <= 6) cuyos elementos tienen una longitud de 1 a 2 caracteres alfanuméricos (A..Z , 0..9).

El programa deberá responder las siguientes consultas:

- Pertenencia de un elemento a un conjunto.
- Igualdad de dos conjuntos.
- Inclusión de un conjunto en otro
- Unión entre conjuntos.

Por ejemplo:

Ca = {0, 1, 3, B, F}
Cb = {0, F}

Cb está incluido en Ca

Nota: El tamaño máximo de cada conjunto será de 20 elementos. Tener en cuenta que el orden de los caracteres en un elemento lo diferencia de otro conformado por los mismos caracteres, siendo así que el elemento AZ es distinto que el elemento ZA y que la unión de dos conjuntos puede resultar en un conjunto con más de 20 elementos o en un conjunto vacío.

Trabajo práctico Nro. 13

Protección de información - Cifrado César

En criptografía, el cifrado César, también conocido como cifrado por desplazamiento, código de César o desplazamiento de César, es una de las técnicas de codificación más simples y más usadas. Es un tipo de cifrado por sustitución en el que una letra en el texto original es reemplazada por otra letra que se encuentra un número fijo de posiciones más adelante según una tabla de 26 posiciones que contiene las letras del alfabeto. Por ejemplo, si usamos el alfabeto ordenado y un desplazamiento de 3, la A sería sustituida por la D (situada 3 lugares a la derecha de la A), la B sería reemplazada por la E, etc. Este método debe su nombre a Julio César, que lo usaba para comunicarse con sus generales.

Ej.: Utilizando un desplazamiento de 6 a la derecha y la siguiente tabla
(ABCDEFGHIJKLMNOPQRSTUVWXYZ)

Texto original: ABCDEFGHIJKLMNOPQRSTUVWXYZ
Texto codificado: GHIJKLMNOPQR STUVWXYZABCDEF

Escribir un programa en assembler Intel 80x86 que codifique o decodifique un texto de entrada generando otro texto codificado según la tabla informada y el desplazamiento previamente ingresado (será un número entre 0 y 9).

Nota: El mensaje estará formado por letras mayúsculas y/o minúsculas y además espacios. Los espacios no deberán ser codificados.

Trabajo práctico Nro. 14

Ordenamiento por método de Burbujeo

Dado un archivo que contiene n números en BPF c/signo de 8 bits (n <= 30) se pide codificar en assembler Intel 80x86 un programa que imprima por pantalla que movimiento se realiza (por ejemplo "Dentro del ciclo For – iteración 1") y el contenido de dicho archivo ordenado en forma ascendente o descendente de acuerdo a lo que elija el usuario, usando un algoritmo de ordenamiento basado en el método burbujeo.

```
procedure burbujeo (int[] vector)
    n = length(vector)
    repeat
        swapped = false
        for i = 1 to n-1 inclusive do
            if vector[i] > vector[i+1] then
                swap (vector[i], vector[i+1] )
                swapped = true
            end if
        end for
    until not swapped
end procedure
```

El método de ordenamiento por burbujeo funciona revisando cada elemento de la lista que va a ser ordenada con el siguiente, intercambiándolos de posición si están en el orden equivocado. Es necesario revisar varias veces toda la lista hasta que no se necesiten más intercambios, lo cual significa que la lista está ordenada.

Nota: no es correcto generar el archivo con un editor de textos de forma tal que cada registro sea una tira de 16 caracteres 1 y 0. Se aconseja el uso de un editor hexadecimal.

Trabajo práctico Nro. 15

Manejo de Matrices (II)

Se pide desarrollar un programa en assembler Intel 80x86 que cargue desde teclado una cantidad X de matrices (con X <= 5) de NxM elementos (con N y M <= 8) y luego permita realizar las siguientes operaciones entre ellas:

- Resta de 2 o más matrices.
- Igualdad entre 2 matrices.
- Multiplicar una matriz por un valor escalar leído desde teclado.
- Modificar o consultar el valor de un elemento (M_{ij}) de una matriz.

Nota: Los elementos que componen a cada matriz (M_{ij}) son números enteros entre -99 y 99

Trabajo práctico Nro. 16

Ordenamiento por método de Inserción

Dado un archivo que contiene n números en BPF c/signo de 8 bits (n <= 30) se pide codificar en assembler Intel 80x86 un programa que imprima por pantalla que movimiento se realiza (por ejemplo “Iniciando el ciclo de i menor a la longitud del vector”) y contenido de dicho archivo ordenado en forma ascendente o descendente de acuerdo a lo que elija el usuario, usando un algoritmo de ordenamiento basado en el método de inserción.

```
procedure insercion (int[] vector)
       i ← 1
       while i < length(vector)
           j ← i
           while j > 0 and vector[j-1] > vector[j]
               swap vector[j] and vector[j-1]
               j ← j - 1
           end while
           i ← i + 1
       end while
end procedure
```

El método de ordenamiento por inserción es una manera muy natural de ordenar para un ser humano. Inicialmente se tiene un solo elemento, que obviamente es un conjunto ordenado. Después, cuando hay k elementos ordenados de menor a mayor, se toma el elemento k+1 y se compara con todos los elementos ya ordenados, deteniéndose cuando se encuentra un elemento menor (todos los elementos mayores han sido desplazados una posición a la derecha) o cuando ya no se encuentran elementos (todos los elementos fueron desplazados y este es el más pequeño). En este punto se inserta el elemento k+1 debiendo desplazarse los demás elementos

Nota: no es correcto generar el archivo con un editor de textos de forma tal que cada registro sea una tira de 16 caracteres 1 y 0. Se aconseja el uso de un editor hexadecimal.

Trabajo práctico Nro. 17

Bolsas de Correo (III)

Se tienen n objetos de pesos P1, P2, ..., Pn (con n <= 20) que deben ser enviados por correo a una misma dirección. La forma más simple sería ponerlos todos en un mismo paquete; sin embargo, el correo no acepta que los paquetes tengan más de 17 Kg. y la suma de los pesos podría ser mayor que eso. Afortunadamente, cada uno de los objetos no pesa más de 17 Kg.

Se trata entonces de pensar un algoritmo que de un método para armar los paquetes, tratando de optimizar su cantidad. Debe escribir un programa en assembler Intel 80x86 que:

- Permita la entrada de un entero positivo n.
- La entrada de los n pesos, verificando que $0<P_i<=17$ donde $i<=n$.
- Los P_i pueden ser valores enteros.
- Exhiba en pantalla la forma en que los objetos deben ser dispuestos en los paquetes.

A su vez existen tres destinos posibles: Mar del Plata, Mendoza y Salta. El correo por normas internas de funcionamiento no puede poner en el mismo paquete objetos que vayan a distinto destino.

Desarrollar un algoritmo que proporcione una forma de acomodar los paquetes de forma que no haya objetos de distinto destino en un mismo paquete y cumpliendo las restricciones de peso. Se sugiere tener una salida como la siguiente:

- $Destino_i$ – $Objeto_1$ (P_1) + $Objeto_2$ (P_2)+ ….. + $Objeto_n$ (P_n)
- $Destino_i$ – $Objeto_1$ (P_1) + $Objeto_2$ (P_2)+ ….. + $Objeto_n$ (P_n)
- $Destino_i$ – $Objeto_1$ (P_1) + $Objeto_2$ (P_2)+ ….. + $Objeto_n$ (P_n)

Trabajo práctico Nro. 18

Conversor de números

Desarrollar en assembler Intel 80x86 un programa que permita:

- Ingresar la configuración hexadecimal números BCD de 4 bytes e indique el número representado (en base 10)
- Ingresar la configuración binaria de un BFP C/Signo de 32 bits e indique el número representado en base 10;
- Ingresar números en base 10 y mostrar su representación en los formatos anteriores en configuración en base 2, 4 y 8.

Trabajo práctico Nro. 19

Espiral Cuadrada

Consideremos una "espiral cuadrada" que parte del origen de coordenadas y toca consecutivamente los puntos (1,0), (1,1), (0,1), (-1,1), (-1,0), (-1,-1), (0,-1), (1,-1), (2,-1), etc.. De esta manera todos los puntos de coordenadas enteras pertenecen a la espiral. Partiendo del punto inicial (0,0), dichos puntos son numerados en forma consecutiva con los enteros no negativos 0, 1, 2, 3, 4, ... etc..

Desarrollar un programa en assembler Intel 80x86 que solucione lo siguiente:

a) Determinar las coordenadas de un punto de la espiral considerada, numerado por "n", donde "n" es un entero no negativo dado.
b) Determinar el entero no negativo "n", correspondiente a un punto de coordenadas enteras (X, Y). Para n <= 100.
c) Dados 2 puntos, indicar la distancia entre ambos (cantidad de puntos que hay entre uno y otro Ej: entre (1,1) y (-1,1) hay distancia 3). Los puntos pueden ser indicados tanto por pares ordenados como por el número “n” que le corresponde.

Trabajo práctico Nro. 20

Conversor de Fechas

Desarrollar un programa en assembler Intel 80x86 que permita convertir fechas válidas con el formato DD/MM/AAAA a formato romano y/o juliano (DDD/AA); también debe permitir su inversa.

Ejemplo:

- Formato Fecha (DD/MM/AAAA): 14/10/2020
- Formato Romano: XIV / X / MMXX
- Formato Juliano (DDD/AA): 288/20