Apunte adicional sobre programación en MIPS R2000

La programación en assembly requiere de acuerdos que deben ser voluntariamente respetados por el programador y que no pueden ser forzados por el compilador debido a la naturaleza del lenguaje. Cualquier proyecto por más pequeño que sea, puede necesitar utilizar subrutinas y hacer pasaje de parámetros, por ello durante los exámenes se evalúa, además de la correctitud en la ejecución, el cumplimiento de los acuerdos en el manejo de registros y la pila para las llamadas tanto recursivas, como las que no lo son.

Adicionalmente cualquier consigna tiene asociado un algoritmo que está especificado para que el alumno no deba pensar en la resolución del problema per se, sino en la codificación solicitada.

Tambien debe observarse que algunos métodos de ingreso como la consola introducen caracteres indeseados en los strings que pueden interferir en la construcción de la solución. El alumno debe enfrentar tales problemas como parte de su entrenamiento.

A continuación se proveen un conjunto de ejercicios que representan diferentes grados de dificultad en diferentes áreas, tanto en cadenas como con números.

Ninguno requiere específicamente alojamiento dinámico, aunque algunos pueden beneficiarse de ello.

Problema 1: El Matching en los routers

Consigna:

Hacer un programa en assembly de MIPS R2000 que solicite un cadena conteniendo una dirección IP de host válida *h1*, otra cadena conteniendo una dirección IP de red válida *r1* y otra cadena conteniendo la máscara de red de la dirección IP introducida *m1* y finalmente reporte si *h1* es una dirección de la red *r1*. Repetir la acción hasta que la primer cadena sea nula.

Definición: una cadena conteniendo una dirección IP o máscara tiene siempre la forma x.x.x.x donde x es un número entre 0 y 255.

Definición: una dirección IP de host *h1* en formato de 32 bits pertenece a una dirección IP de red *r1* en formato de 32 bits si dada una máscara *m1* de 32 bits la operación *h1 && m1 == r1* (1) es cierta.

Estrategia:

Se asume que las cadenas *h1*, *r1* y *m1* ingresadas son válidas. Deberá transformar dichas cadenas a números enteros de 32 bits para poder realizar la operación (1). Para ello creará una subrutina *aton* (ascii to network) cuyo argumento es una cadena pasada por referencia y que devuelve un entero de 32 bits. A partir del resultado obtenido en (1) imprimirá el cartel correspondiente indicando pertenencia o no.

Se sugiere utilizar como estrategia analizar cada componente del string separado por punto y convertirlo a un número usando la expresión básica digito * peso iterativamente.

Veamos un ejemplo:

Para la máscara 255.255.255.248 hay 4 componentes. Analizemos como convertir la primer componente que es 255, para ello debo convertir cada caracter numérico en un número y realizar la siguiente cuenta 2 * 100 + 5 * 10 + 5 o su equivalente (2 * 10 + 5) * 10 + 5 . Luego de realizada dicha operación se obtiene el valor numérico 0xff en hexadecimal que debe estár en los primeros 8 bits del entero de 32 bits a obtener o sea 0xff000000 luego repito la operación con la segunda y tercera componente obteniendo 0xffffff00 y finalmente la última será 0xf8 por lo que la máscara convertida con aton quedará 0xfffffff8.

Ejemplo de ejecución:

```
Introduzca una dirección IP: 200.3.124.5
Introduzca una dirección de red: 200.3.124.0
Introduzca una máscara: 255.255.255.248
La dirección 200.3.124.5 pertenece a la red 200.3.124.0

Introduzca una dirección IP: 200.3.124.16
Introduzca una dirección de red: 200.3.124.0
Introduzca una máscara: 255.255.255.248
La dirección 200.3.124.16 no pertenece a la red 200.3.124.0
```

Problema 2: Obtención de la TFA

Introducción:

En muchas disciplinas es necesario obtener algún tipo de análisis estadístico de la información. Uno de los conceptos más conocidos es la frecuencia absoluta de un suceso.

Definición: llamamos frecuencia absoluta de un suceso al número de veces que aparece dicho suceso

Un suceso sencillo dentro de un texto sería la ocurrencia de una letra. Su frecuencia sería el número de veces que ocurre esa letra en el texto. Para obtener la frecuencia absoluta de una letra específica contamos el número de veces que aparece esa letra en el texto dado. Por ejemplo (1) la letra “a” en el siguiente texto:

“el sol asomaba en el horizonte”

aparece dos veces o sea su frecuencia absoluta es 2. Si obtenemos todas las frecuencias de las distintas letras que conforman el texto anterior obtenemos una tabla como la que sigue a continuación, llamada tabla de frecuencias absolutas o TFA:

```
a b c d e f g h i j k l m n o p q r s t u v w x y z
3 1 0 0 4 0 0 1 1 0 0 3 1 2 4 0 0 1 2 1 0 0 0 0 0 1
```

Consigna:

Hacer un programa en assembly de MIPS R2000 que solicite un cadena arbitraria de texto que solo contiene alguna de las 26 letras del alfabeto en minúsculas además de espacios y devuelva la TFA de dicho texto. Repetir la acción al menos que la cadena sea nula.

Estrategia:

Realizar un subrutina *tfa* cuyos argumentos son una cadena pasada por referencia y un vector de enteros de tamaño 26 que constituye la TFA y que no tiene parámetros de devolución. El programa principal solicitará una frase e imprimirá la TFA, así sucesivamente hasta que se ingrese una frase vacía.

Construir la TFA empezando por la primer letra y así sucesivamente es trivial pero ineficiente, porque si una letra no está en el texto debemos recorrerlo completo. Una alternativa es recorrer la frase e incrementar la cuenta del valor existente en la TFA.

Veamos un ejemplo:

Si la primer letra es una “e”, como en el ejemplo (1) podemos buscar la posición de “e” en la tabla e incrementar su valor previo (como es la primera vez es cero) en uno y luego seguir con la próxima letra. Los espacios simplemente se ignoran, porque estarían fuera del rango de la tabla.

Ejemplo de ejecución:

```
Introduzca una frase: el sol asomaba en el horizonte
a b c d e f g h i j k l m n o p q r s t u v w x y z
3 1 0 0 4 0 0 1 1 0 0 3 1 2 4 0 0 1 2 1 0 0 0 0 0 1
```

Problema 3: El cifrado de CESAR

Introducción:

El cifrado por sustitución es una vieja técnica ya utilizada en los tiempos de Roma, para ocultar información al enemigo. El proceso consiste en tomar una frase y transformarla en otra que no sea legible. Para ello las letras que componen dicha frase son sustituidas por otras según una tabla simple letra *x* → por letra *y*. La condición fundamental es que dicha tabla sea una función biyectiva para que la frase cifrada se pueda descifrar.

Consigna:

Hacer un programa en assembly de MIPS R2000 que solicite una cadena de 26 letras diferentes, a la que llamaremos clave y una frase en letras mayúsculas y luego cifre dicha frase por sustitución utilizando dicha clave. El programa permitirá comprobar por inspección visual que la frase cifrada es correcta, para ello deberá descifradar la frase cifrada e imprimirla nuevamente. Se repetirá la acción al menos que se entre una frase cuya cadena sea nula, sin solicitar nuevamente una clave.

Estrategía:

Realizar un subrutina *cifrar* que tiene como argumentos tres cadenas por referencia *s1*, *s2* y *clave1*. Dicha subrutina deberá leer cada carater de *s1* y utilizando su posición relativa en la tabla ASCII obtener el caracter correspondiente de *clave1* e insertarlo en *s2*. Además realizar una subrutina decode que tiene como argumentos dos cadenas *clave1*, *clave2* y transforma la primer clave en otra clave. Con esta nueva clave se puede usar *cifrar* para descifrar el mensaje cifrado con la primer clave.

Sea la cadena *clave1* de la forma “TUZCAMYDFXIHQBKJOLNERSPVGW”, para comprender el concepto de sustitución tomemos un ejemplo sencillo como MAMA donde M está en la posición 12 de la tabla ASCII si consideramos que A está en la posición 0 (recordar que las mayúsculas están en forma consecutiva desde la letra A cuyo valor es 65). Luego la posición 12 en la *clave1* la ocupa la letra “Q” y la 0 la “T” entonces MAMA se codifica QTQT. Este es el funcionamiento de *cifrar*, cuyo proceso es trivial y eficiente. Como se observa a continuación superponiendo la cadena de letras consecutivas y la clave forman la tabla mencionada al principio:

```
ABCDEFGHIJKLMNOPQRSTUVWXYZ
TUZCAMYDFXIHQBKJOLNERSPVGW
```

Descifrar el mensaje usando la clave de cifrado no es trivial pero es posible y eficiente debido a la biyectividad de la tabla. A partir de la misma se puede obtener la siguiente clave que consiste en la función inversa:

```
ENDHTIYLKPORFSQWMUVABXZJGC
```

Debido a la propiedad de simetría del cifrado por sustitución si ciframos nuevamente QTQT usando *cifrar* con la clave anterior obtenemos MAMA.

¿Como obtenemos la última clave a partir de la primera? Esta transformación es simple por ejemplo para T que está en la tabla ASCII en la posición relativa 19, debería representar en la clave la letra A (recordar que estamos descifrando QTQT) o sea que debo contruir una clave donde A ocupe la posición 19 y así con las 26 letras restantes. O sea uso cada letra de la clave como índice donde debo colocar la letra que le corresponde en el alfabeto, como se muestra en el ejemplo:

```
TUZCAMYDFXIHQBKJOLNERSPVGW
ABCDEFGHIJKLMNOPQRSTUVWXYZ
+------------------+        (paso a la posición 19)
...................A......
```

Luego en el programa principal debe llamarse a *cifrar* obtener la cadena cifrada e imprimirla, llamarse a *decode* obtener la nueva clave, para llamar nuevamente a *cifrar* con dicha clave y la frase cifrada y obtener la cadena descifrada e imprimirla.

Ejemplo de ejecución:

```
Introduzca la clave: TUZCAMYDFXIHQBKJOLNERSPVGW
Introduzca la frase a cifrar: HOLA MUNDO
La frase cifrada es: DKHT QRBCK
La frase descifrada es: HOLA MUNDO
```

Problema 4: El algoritmo de Euclides

Introducción:

En teoría de números es muy común buscar los divisores de un número dado y Euclides divisó un método para obtener el divisor común que además tiene la propiedad de ser máximo o sea que es el mayor divisor de ambos números llamado máximo común divisor MCD. Dicho algoritmo es muy sencillo y facil de implementar recursivamente.

Consigna:

Hacer un programa en assembly de MIPS R2000 que solicite dos números e informe cuál es el MCD, utilizando el algoritmo de Euclides y repita la acción a menos que uno de los números sea 1 o 0 con lo cuál abandonará el programa.

Estrategia:

A continuación se provee un pseudocódigo del algoritmo de la subrutina *mcd* a implementar:

```
int mcd(int a, int b) {
    if a == b return a else return (a < b) ? mcd(b-a, a) : mcd(b, a-b);
}
```

Ejemplo de ejecución:

```
Introduzca un número: 10
Introduzca otro número: 6
El máximo común divisor es: 2
```

Problema 5: Las Torres de Hanoi

Introducción:

El juego de Las Torres de Hanoi consiste en mover un conjunto de n discos desde una varilla vertical a otra, empezando todos los discos en una varilla ordenados de más grande a más pequeño y habiendo 2 varillas más inicialmente vacías. Todo esto siguiendo las siguientes reglas: sólo puedes mover los discos de uno en uno, los discos tienen que estar ordenados en todo momento estando los más grandes siempre debajo y sólo se puede mover los discos de arriba de cada varilla.

Este juego desencadenó todo una serie de estudios famosos en el campo de la algoritmia, existiendo un algoritmo recursivo bien conocido para su solución con 3 varillas, siendo un problema aún abierto para ciertas generalizaciones como m varillas.

Consigna:

Hacer un programa en assembly de MIPS R2000 que solicite un entero que representa el número de discos e informe todos los movimientos necesarios para solucionar dicho problema. Repita la acción a menos que el número sea menor que dos con lo cuál abandonará el programa.

Estrategia:

Construir una subrutina *hanoi* que imprima las movidas para resolver el juego mediante el siguiente pseudocódigo provisto:

```
void hanoi(int n, char orig, char dest, char spare) {
   if (n==1) imp disco n de orig a dest;
   else {
       hanoi(n-1, orig, spare, dest);
       imp disco n de orig a dest;
       hanoi(n-1, spare, dest, orig);
   }
}
```

Ejemplo de ejecución:

```
Introduzca el número de discos: 2
Movimientos:
disco 1 de A a B
disco 2 de A a C
disco 1 de B a C
```

Problema 6: Permutaciones recursivas

Introducción:

El siguiente es un ejercicio orientado a las especificaciones cuya mayor complejidad recide en el uso de indices complejos en cadenas. El algoritmo provisto no tiene ninguna ventaja excepto la elegancia de su brevedad ya que obtiene las permutaciones de orden n alrededor de un pivote al que incrementa recursivamente, aunque introduce algunas complejidades innecesarias por el caracter nulo que es arrastrado dentro de la cadena. Se sugiere implementarlo tal como está sin introducir modificaciones.

Consigna:

Hacer un programa en assembly de MIPS R2000 que imprima todas las permutaciones posibles de letras diferentes contenidas en una cadena solicitada cuya longitud es desconocida. Se repetirá la acción al menos que se entre una cadena que sea nula.

Estrategia:

Construir una subrutina *strlen* que obtenga el número de caracteres de una cadena. La cadena debe estar limpia (no contener saltos de carro, etc.). Luego construya una subrutina *perm* que permite obtener todas las permutaciones para una cadena de caracteres, cuyos argumentos son un puntero a la cadena y un entero para la posición a fijar, inicialmente a 0. A continuación se provee un pseudocódigo del algoritmo de la subrutina *perm*:

```
void perm(char * cad, int l) {
   char c;
   int i, j;
   int n = strlen(cad);

   for(i = 0; i < n-l; i++) {
      if(n-l > 2) perm(cad, l+1);
      else imprimir cad;
      c = cad[l];
      cad[l] = cad[l+i+1];
      cad[l+i+1] = c;
      if(l+i == n-1) {
         for(j = l; j < n; j++) cad[j] = cad[j+1];
         cad[n] = 0;
      }
   }
}
```

Ejemplo de ejecución:

```
Introduzca una cadena de letras diferentes: abc
Permutaciones:
abc,acb,bac,bca,cab,cba
```