1 Sea la siguiente subrutina expresada en lenguaje de alto nivel:

```
/* El parámetro n se pasa por valor y vector por dirección */
funcion (int n, int vector[])
{
        int aux[n];     /* Vector de enteros de una palabra */
        int i = 0;      /* Entero de una palabra inicializado a 0 */

        /* Copia el vector que se pasa como parámetro en
           el vector de la variable local aux */
        while (i < n)
        {
                aux [i] = vector[i];
                i = i + 1;
        }
```

Programe en ensamblador del 88110 el fragmento de la subrutina funcion expuesto anteriormente.

2 Una tabla contiene elementos que indican el coste de una llamada telefónica. Cada entrada de la tabla contiene dos enteros: el primero es el número de teléfono que ha realizado la llamada; el segundo entero contiene el coste de dicha llamada. La tabla no está ordenada y finaliza cuando el campo teléfono contiene el valor 0.

Se desea generar una segunda tabla, del mismo formato que la anterior, que contenga los números de teléfono con el coste acumulado de todas las llamadas de la primera tabla. Para ello se supondrá que existe una subrutina de biblioteca que realiza la búsqueda de un elemento en la tabla:

```
dir_elem = BUSCAR(Tabla, Elem)
```

- `Tabla`: Es la Tabla en la que se desea realizar la búsqueda. Se pasa por dirección.
- `Elem`: Es el número de teléfono (entero) que se desea buscar en la tabla. Se pasa por valor.

  Ambos parámetros se pasan en la pila.

Esta subrutina devuelve en el registro r29 se devuelve la dirección de la entrada de la tabla donde se ha encontrado el nº de teléfono buscado o el valor 0xffffffff si no se ha encontrado.

Se pide realizar las siguientes subrutinas en ensamblador del MC88110:

a) `INSERTAR(Tabla, Elem)` genera una entrada del elemento `Elem` en la tabla `Tabla`. El total de la nueva entrada es 0. Los parámetros se pasan en la pila y se devuelve en r29 la dirección de comienzo de la nueva entrada.

b) `ACTUALIZAR(Resumen, Llamadas)` genera la tabla `Resumen` a partir de la tabla `Llamadas`. Ambas tablas se pasan por dirección. Esta rutina podrá hacer uso de las otras dos rutinas mencionadas en el enunciado.

Se supondrá que están definidas todas las macros que se han explicado en la parte teórica de la asignatura, que la subrutina llamante deja disponibles todos los registros excepto r1, r30 (SP) y r31 (FP); que la pila crece hacia direcciones de memoria decrecientes y el puntero de pila apunta a la última posición ocupada (de la misma forma que se ha utilizado en el proyecto). A modo de ejemplo a continuación se muestra la tabla inicial y la tabla resultado.

TABLA INICIAL
(`Llamadas`)

Teléfono	Coste
609609609	12
601601601	10
609609609	8
609609609	10
0	??

TABLA RESULTADO
(`Resumen`)

Teléfono	Total
609609609	30
601601601	10
0	??

3 Programe en ensamblador del 88110 el fragmento de la subrutina SUBRUTINA que se detalla a continuación, teniendo en cuenta que el índice del primer elemento de los vectores es el 0, los parámetros se pasan en la pila y las variables locales se almacenan en el marco de pila. TRANS es una subrutina que aplica una transformación al elemento que se pasa como parámetro por dirección en la pila.

```
SUBRUTINA (int v[], int n)
{
   int aux[n];         /* Variable local: Vector de n enteros auxiliar */
   int i = n;          /* Variable local: Entero inicializado a n */

   while (i > 0)
      {
         aux[n-i] = v[i-1]; /* En aux se genera el vector "invertido" */
         TRANS(&aux[n-i]);  /* Se aplica una transformación al elemento del
                               vector invertido que se pasa por dirección */
         i = i - 1;         /* es decir, el primer elemento de v pasa a ser */
                            /* el último de aux */
      }

   . . . .
}
```

4 Programe en ensamblador del 88110 los fragmentos de la subrutina `SUBRUTINA` que se detallan a continuación:

```
SUBRUTINA (int p1, int p2, int p3)
{
   int  vl_a = 0;    /* vl_a = 0 */
   char vl_b = 0;    /* vl_b = null 8 bits */
   char vl_c = 45;   /* vl_c = 'A'  8 bits */

   /* Fragmento 1 */
         ....
   /* Fragmento 2 */

   funcion (0, p2, vl_a);

   /* Fragmento 3 */
         ....
   /* Fin */
}
```

a) Marco de pila: Creación del marco de pila incluyendo la inicialización de las variables locales.

b) Fragmento 2: Llamada a la subrutina `funcion` incluyendo el paso de los tres parámetros por valor y en la pila. Suponga que en el Fragmento 1 se hace uso de los registros r5, r6, r7, r8, r9 y r10; que r9 y r10 contienen información relevante para el Fragmento 3; y que la subrutina llamante, `SUBRUTINA`, deja disponibles todos los registros excepto r1, r30 (SP) y r31 (FP).

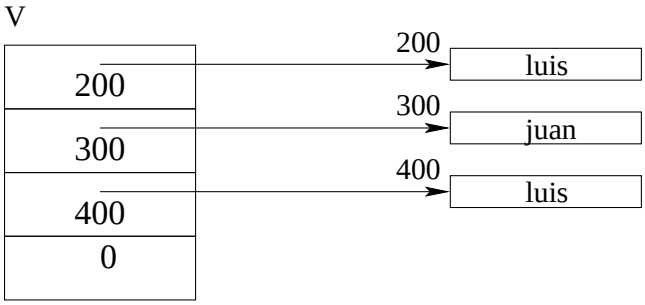
c) Fin: Retorno al programa llamante, incluyendo la destrucción del marco de pila.

5 Programe en ensamblador del 88110 la función `mapa` que recibe dos parámetros en la pila:

- `v`: Es un vector que contiene direcciones de comienzo de cadenas de caracteres. La dirección 0 indicará el final del vector cuyo tamaño nunca será superior a 32 elementos. Se pasa por dirección.
- `str`: Es una cadena de caracteres que se pasa por dirección.

Esta función devolverá en r29 una máscara que contendrá en el bit i-ésimo un 1 si la cadena `str` es igual a la cadena apuntada por el elemento i-ésimo del vector o 0 en caso contrario. Se recuerda que el formato de la instrucción para poner un campo de bits a 1 es `set rD,rS1,W5<O5>` o `set rD,rS1,rS2`. En este último caso los bits 9-5 y 4-0 de `rS2` se toman como los campos `W5` (ancho del campo de bit) y `O5` (bit origen del campo de bit) respectivamente.

Se supondrá que existe la función de librería `strcmp(str1,str2)` que compara las cadenas de caracteres `str1` y `str2` y devuelve un 0 en r29 si son iguales o 1 si no lo son. El resultado de invocar a la función `mapa` cuando el parámetro `str` contiene el valor `luis` y V es el vector que aparece en la figura es: `r29 = 0x00000005`.



6

a) Programe en ensamblador del 88110 la subrutina `INVERTIR` que recibe cuatro parámetros en la pila, todos de 1 palabra:

- `imagen`: Es la dirección de comienzo de una imagen de m×n píxeles.
- `m`: Es el número de filas de la imagen.
- `n`: Es el número de columnas de la imagen.
- `lg`: Es el tamaño en palabras de cada píxel de la imagen.

Esta subrutina transformará la imagen que se pasa como parámetro invirtiéndola tanto horizontal como verticalmente y dejándola en la misma zona de la memoria, es decir, a partir de la dirección `imagen`. El siguiente ejemplo ilustra en qué orden deben quedar los píxeles de una imagen de 3×4 tras la ejecución de la subrutina `INVERTIR`.

```
imagen: 1  2  3  4              imagen: 12 11 10  9
        5  6  7  8    ====>              8  7  6  5
        9 10 11 12                       4  3  2  1
```

La subrutina `INVERTIR` usará dos punteros, `p1` y `p2`: `p1` para recorrer la imagen partiendo del primer píxel hacia el último, y el `p2` para recorrerla en sentido inverso, es decir, partiendo del último píxel y hacia el primero.

La transformación de la imagen se realizará mediante un bucle que recorrerá la mitad de los píxeles. En cada paso del bucle, se intercambiarán los píxeles direccionados por `p1` y `p2`. Dicho intercambio se realizará mediante la subrutina `INTERCAMBIAR`, de tres parámetros de 1 palabra cada uno, que se pasan en la pila:

- `d1`: Dirección de memoria del primer píxel.
- `d2`: Dirección de memoria del segundo píxel.
- `lg`: Longitud en palabras del píxel.

Para realizar el intercambio de los píxeles, la subrutina `INTERCAMBIAR` se apoyará en una variable local, `aux`, de `lg` palabras de longitud: `píxel(d1) ==> aux; píxel(d2) ==> píxel(d1); aux ==> píxel(d2)`.

b) Programe en ensamblador del 88110 la subrutina `INTERCAMBIAR`. Suponga que está definida la Macro `COPIA_PIXEL (px1, px2, long)` que copia el píxel direccionado por el registro `px1` a partir del píxel direccionado por el registro `px2`, siendo `long` un registro que contiene la longitud del pixel.

7 Se dispone de un texto almacenado en memoria que finaliza con el carácter NUL (código ASCII 0x00). El texto contiene una URL que se divide en protocolo, host y path. El protocolo está separado del host por la cadena de caracteres "://" y el host está separado del path por el carácter '/' que forma parte del path. Un ejemplo de una URL de este formato es: `http://www.datsi.fi.upm.es/docencia/Estructura_09`. En este caso el protocolo es `http`, el host es `www.datsi.fi.upm.es` y el path es `/docencia/Estructura_09`

Adicionalmente se dispone de una tabla no ordenada, que es una variable global, en que cada una de sus entradas contiene dos palabras:

- `puerto`: Es un entero que contiene el puerto asociado al protocolo del campo posterior.
- `str`: Es un puntero a una cadena de caracteres que contiene el nombre de un protocolo.

Una entrada que contenga el valor NULL (0) en el campo `str`, indica el final de la tabla.

Se pide programar en ensamblador del MC88110:

a) una subrutina `BUSCA (Tabla, Protocolo)` que busca la cadena de caracteres `Protocolo` en `Tabla`. Esta rutina devuelve un valor entero que contiene el puerto asociado al protocolo o el valor -1 si el protocolo no se encuentra en la tabla. Los parámetros se pasan por dirección en la pila.

Se supondrá que existe la función de librería `strcmp(str1,str2)` que compara las cadenas de caracteres `str1` y `str2` y devuelve un 0 en r29 si son iguales o 1 si no lo son. Los parámetros se le pasan por dirección en la pila, y cada cadena de caracteres finaliza con un carácter NUL.

Adicionalmente la función de librería `memcmp(s1,s2,tam)` compara dos zonas de memoria apuntadas por `s1` y `s2` de tamaño `tam` bytes. Devuelve un 0 en r29 si son iguales o 1 si no lo son. Los parámetros se le pasan por dirección en la pila.

b) la subrutina `SEPARA(URL, Host, Path, Puerto)` que separa cada uno de los componentes de `URL` que se han descrito al comienzo del enunciado. `Host` y `Path` son parámetros de salida y son cadenas de caracteres que se pasan por dirección. El parámetro `Puerto` es un entero que se pasa por dirección y es un parámetro de salida que contendrá el valor del puerto asociado al protocolo contenido en `URL`. Suponga que a partir de la posición de memoria `separador` está contenida la cadena de caracteres "://" y que el código ASCII de '/' es el 0x2F.

Para la realización de este ejercicio se llevará a cabo el tratamiento del Marco de Pila descrito en clase y se s upondrá que están definidas todas las macros que se han explicado en la parte teórica de la asignatura; que las subrutinas llamantes dejan disponibles todos los registros excepto r1, r30 (SP) y r31 (FP); que la pila crece hacia direcciones de memoria decrecientes y el puntero de pila apunta a la última posición ocupada (de la misma forma que se ha utilizado en el proyecto).

A modo de ejemplo se muestra el resultado de una llamada a `SEPARA` con la URL indicada al comienzo del enunciado y suponiendo que el contenido de la tabla es el que se indica a continuación.

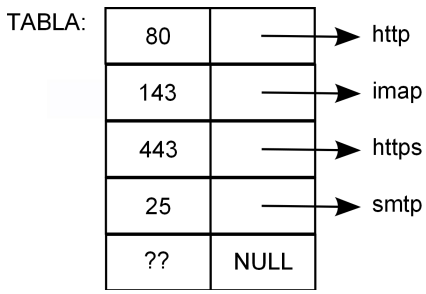


Figura 1

- `Puerto`: 80
- `Host`: www.datsi.fi.upm.es
- `Path`: /docencia/Estructura_09

8 Programe en ensamblador del MC88110 las siguientes subrutinas:

a) Subrutina `CEROS(x, vector)` que calcula el número de elementos que son 0 en un vector de x elementos enteros de una palabra, siendo $x > 0$. Los parámetros se pasan en la pila: `x` por valor y `vector` por dirección. El resultado se devuelve en el registro r29. Para programar esta subrutina, no es necesario que cree un marco de pila.

b) Subrutina `DIAGONAL(m, matriz, resultado)` que determina cuántos elementos distintos de cero contiene la diagonal principal de una matriz de enteros de una palabra, almacenada en memoria por filas. La matriz es cuadrada de orden m, siendo $0 < m < 256$. Los parámetros se pasan en la pila: m por valor y los otros por dirección.

Esta subrutina extrae los elementos de la diagonal principal de la matriz almacenándolos en un vector como variables locales y hace uso de la subrutina `CEROS` descrita en el apartado anterior. Para su realización se llevará a cabo el tratamiento del marco de pila descrito en clase y se supondrá que están definidas todas las macros que se han explicado en la parte teórica de la asignatura, que la subrutina llamante deja disponibles todos los registros (excepto r1, r30 (SP) y r31 (FP)), que la pila crece hacia direcciones de memoria decrecientes y que el puntero de pila apunta a la última posición ocupada de la cima de la pila.