Estructura de Computadores

(Grado MI)

Programación en ensamblador

Enunciados de problemas

abril 2019

---

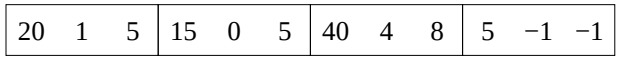
1 Programe en ensamblador del 88110 la función `compacta` que recibe cuatro parámetros en la pila:

- `v`: Es una matriz de enteros de m filas por n columnas almacenada por filas. Se pasa por dirección.
- `p`: Es una lista compacta que contiene elementos de una matriz. Se pasa por dirección. Cada uno de los elementos de la lista se compondrá de tres campos enteros:
  - `elemento`: Contiene el elemento que pertenece a la matriz.
  - `fila`: Indica la fila a la que pertenece el elemento. Las filas comienzan a numerarse en la posición 0.
  - `columna`: Indica la columna a la que pertenece el elemento. Las columnas comienzan a numerarse en la posición 0.

  El final de la lista está indicado con el el valor -1 contenido en el campo fila.
- `m`: Es un entero que contiene el número de filas de la matriz v. Se pasa por valor.
- `n`: Es un entero que contiene el número de columnas de la matriz v. Se pasa por valor.

Esta función rellenará la lista apuntada por p con los elementos no nulos de la matriz v. Suponga que los valores n y m son mayores que 0.

El resultado del ejemplo que se muestra en la figura proviene de una matriz en la que su elemento (0, 5) contiene el valor 15, el (1, 5) el valor 20 y el (4, 8) el valor 40. El resto de los elementos de la matriz son 0.



2 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 $0 < x$. 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.

3 Programe las siguientes subrutinas en ensamblador de MC88110:

a) (20 %) La función `extrae_palabra(char *texto,char *resultado)` extrae la primera palabra que se encuentra a partir del puntero `texto` y la copia a partir de la dirección `resultado`. Se considerará una palabra un conjunto de caracteres delimitado por espacios en blanco, tabuladores, puntos o comas, cuyos códigos ASCII son 0x20, 0x09, 0x2e y 0x2c. La función devuelve en r29 el tamaño en caracteres de la palabra copiada excluyendo el separador.

b) (80 %) La función `contar(char *palabra, char *texto)` que cuenta el número de ocurrencias de la palabra apuntada por el primer parámetro en el texto que se pasa en el segundo. La función `contar` recibe los siguientes parámetros en la pila:

- `palabra`: Es una cadena de caracteres. Se pasa por dirección y contiene la palabra (terminada por el carácter NUL, cuyo código ASCII es 0) que se desea contar.

- `texto`: Es una cadena de caracteres. Se pasa por dirección y contiene el texto en el que se desea contar el número de veces que aparece `palabra`. Supóngase que todas las palabras del texto están separadas por un único separador y que la longitud de todas las palabras es menor que 100 caracteres.

Para la realización de este ejercicio 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 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). Además se supondrá que la subrutina de biblioteca `strcmp(s1,s2)` compara las cadenas de caracteres `s1` y `s2` y devuelve en r29 en valor 0 si son iguales y 1 si no lo son. `s1` y `s2` son dos cadenas de caracteres que acaban con el carácter NUL. Se puede invocar a la subrutina `extrae_palabra` del apartado anterior.

4 Se desea realizar la trasposición de una matriz de enteros cuadrada almacenada por filas. Las filas y las columnas de la matriz comienzan a numerarse en el elemento 0. Para ello se pide realizar las siguientes subrutinas:

a) `calc_dir_efectiva`: Devuelve en el registro r29 la dirección que ocupa un elemento de la matriz. Recibe los siguientes parámetros en la pila:

- `matriz`: Es una matriz de enteros. Se pasa por dirección.
- `dimension`: Es un valor entero que indica la dimensión de la matriz cuadrada. Se pasa por valor.
- `fila`: Es un valor entero que contiene el número de fila del elemento del que se desea calcular la dirección que ocupa en memoria. Se pasa por valor.
- `columna`: Es un valor entero que contiene el número de columna del elemento del que se desea calcular la dirección que ocupa en memoria. Se pasa por valor.

b) `trasponer`: Realiza la trasposición de la matriz que se pasa como primer parámetro y almacena el resultado de dicha operación en la propia matriz. Para la realización de esta función puede utilizar la función del apartado anterior. Recibe los siguientes parámetros en la pila:

- `matriz`: Es una matriz de enteros. Se pasa por dirección.
- `dimension`: Es un valor entero que indica la dimensión de la matriz cuadrada. Se pasa por valor.

En la figura que se muestra a continuación aparece un ejemplo de la matriz original y del resultado de aplicar a esta matriz (de dimensión 4) la subrutina `trasponer(matriz,4)`.

$$Original = \begin{pmatrix} 1 & 2 & 3 & 4 \\ 5 & 6 & 7 & 8 \\ 9 & 10 & 11 & 12 \\ 13 & 14 & 15 & 16 \end{pmatrix} \qquad Resultado = \begin{pmatrix} 1 & 5 & 9 & 13 \\ 2 & 6 & 10 & 14 \\ 3 & 7 & 11 & 15 \\ 4 & 8 & 12 & 16 \end{pmatrix}$$

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

```
funcion (int p1, int p2, int vector[])
{
   /* Fragmento 1 */
   int  i, j;         /* Var. locales enteras */
   int matriz [7][7]; /* Matriz de enteros de 7 filas y 7 columnas almacenada por filas*/

   /* Se inicializan todos los elementos de la matriz con su número de fila */
   for (i=0; i < 7; i = i + 1) {
       for (j=0; j < 7; j = j + 1) {
           matriz[i][j] = i;
       }
   }

           ....

   /* Fragmento 2 */
   i = seleccionar (matriz, vector);
   j = manipular (matriz, i, p1);
   return i+j; // El valor de retorno (i+j) se devuelve en r29
   /* Fin */
}
```

a) Fragmento 1: Creación del marco de pila incluyendo la inicialización de la variable local `matriz`.

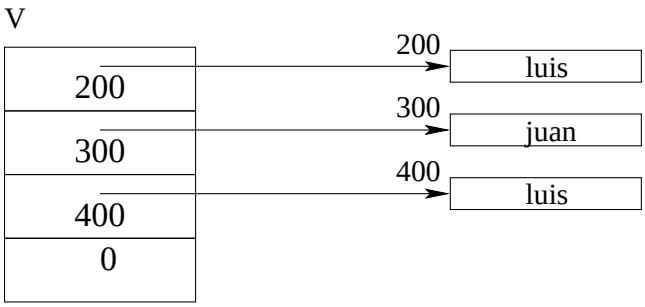
b) Fragmento 2: Llamada a la subrutina `seleccionar` incluyendo el paso de los parámetros por dirección y en la pila. Llamada a la subrutina `manipular`, que recibe el primer parámetro por dirección y los dos últimos por valor. Suponga que la subrutina llamante, `funcion`, deja disponibles todos los registros excepto r1, r30 (SP) y r31 (FP) y que todas las subrutinas del enunciado devuelven el valor de retorno en r29.

6 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`.



7 (2 puntos) 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 2 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.

8 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.

9 Sea una lista doblemente encadenada ordenada de forma creciente. Cada elemento de la lista se compone de tres palabras: la primera contiene un entero que es el campo de información de la lista, la segunda contiene un puntero (dirección) al siguiente elemento de la lista (0 si es el último) y la tercera contiene un puntero (dirección) al elemento anterior de la lista (0 si es el primero). La dirección de comienzo de la lista se almacena en una palabra en memoria y apunta al primer elemento (0 si está vacía).

Realice una subrutina `INSERTA(elem,lista)` que inserta el elemento `elem`, que es un elemento de la lista tal y como se ha definido anteriormente, en la `lista` que se pasa como segundo parámetro. Ambos parámetros se pasan por dirección.

10 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.

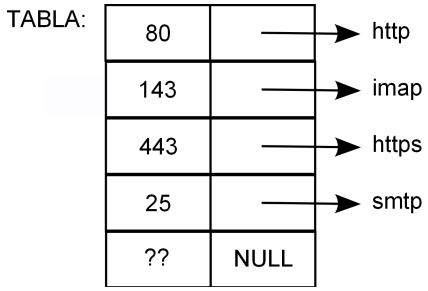
Se supondrá que existe la función de librería `strcmp(str1,str2)` que compara las cadenas de caracteres `str1` y `str2` y devuelve un 0 en r29 si son iguales o 1 si no lo son. Los parámetros se le pasan por dirección en la pila, y cada cadena de caracteres finaliza con un carácter NUL.

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 supondrá 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.



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

Figura 1

11 La sucesión de Fibonacci es una sucesión que comienza con los números 0 y 1, que corresponden a los índices 0 y 1, y a partir de estos, cada término se calcula como la suma de los dos anteriores:

$$F_n = F_{n-1} + F_{n-2} \text{ si } n \geq 2 \; ; \; F_0 = 0; F_1 = 1$$

Para acelerar el cálculo de un elemento de la sucesión, se han almacenado en una lista encadenada **algunos** términos de la misma. La lista está ordenada de forma creciente por el valor del índice de los términos almacenados y cada elemento contiene los siguientes campos, ocupando cada uno una palabra:

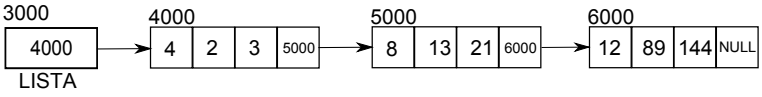
- **i:** Contiene el índice del término de la sucesión. Es un valor mayor que 1.
- F_{i-1}**:** Contiene el valor del término de la sucesión para el índice **i-1**.
- F_i**:** Contiene el valor del término de la sucesión para el índice **i**.
- **siguiente:** Es un puntero al siguiente elemento de la lista. Si es NULL indica que es el último.

Programe en ensamblador del MC88110 las subrutinas que se describen a continuación, sabiendo que todas ellas reciben sus parámetros en la pila.

a) (20 %) `busca(n,lista)`: Selecciona entre los elementos de la `lista` cuyo campo `i` sea menor o igual al parámetro `n`, el que tenga un valor mayor en dicho campo. Devuelve en r29 un puntero al elemento seleccionado. Ambos parámetros se pasan **por valor**. Si la lista está vacía o el campo `i` del primer elemento es mayor que `n` devuelve el valor NULL. Ejemplo: si se pasa la lista que aparece en la figura (4000 en el parámetro `lista`) y el valor 10 en `n`, devolverá la dirección del elemento cuyo campo `i` tiene el valor 8 (r29 contendrá 5000).

b) (80 %) `Fibonacci(elem,lista)`: Calcula el término de la sucesión de Fibonacci para un índice determinado, devolviéndolo en r29. El parámetro `elem` es un entero que corresponde al índice del término que se desea calcular. El parámetro `lista` contiene la dirección de la lista en la que están almacenados algunos términos de la sucesión. Para realizar el cálculo la solución tiene que apoyarse en la subrutina anterior, tal y como se describe a continuación:

- Si `elem` contiene 0 ó 1, devuelve el mismo valor.
- Se debe buscar el valor `elem` en `lista`.
- Se debe calcular el valor de la sucesión a partir del resultado devuelto por `busca` (si no ha devuelto el valor NULL)



Si se invoca a `Fibonacci` con el valor 10 en `elem` y la `lista` que aparece en la figura, devolverá en r29 el valor 55, calculado a partir del término de la sucesión cuyo índice es 8 que está almacenado en la lista.

Para la realización de este ejercicio 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 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.

12 Sea una matriz de enteros, `matriz`, cuya estructura de datos contiene en la primera palabra el valor de la primera dimensión (número de filas), en la siguiente palabra está el valor de la otra dimensión (número de columnas), y en el resto de palabras consecutivas los elementos enteros de la matriz almacenados por filas.

Se desea contabilizar el número de valores negativos, valores igual a cero y valores estrictamente positivos que contine cada fila de la matriz. Para ello, se definen tres vectores consecutivos de dimensión igual al número de filas, `vneg`, `vceros` y `vpos`.

- El vector `vneg` tendrá en cada elemento i-ésimo el contador de elementos negativos de la fila i-ésima de la `matriz`.
- El vector `vceros` tendrá en cada elemento i-ésimo el contador de elementos iguales a 0 de la fila i-ésima de la `matriz`.
- El vector `vpos` tendrá en cada elemento i-ésimo el contador de elementos estrictamente positivos de la fila i-ésima de la `matriz`.

Programe en ensamblador del MC88110 la subrutina `signos(matriz,vneg,vceros,vpos)` que completa los valores de los vectores `vneg`, `vceros` y `vpos` para la matriz de entrada `matriz`. Todos los parámetros se pasan en pila por dirección.

Un ejemplo de una invocación a esta subrutina se muestra en la figura.

Para la realización de este ejercicio no es necesario el tratamiento de marco de pila y se supondrá 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).

`matriz`

3	4		
1	-1	1	-1
0	2	3	-4
1	0	3	0

`vneg`

2
1
0

`vceros`

0
1
2

`vpos`

2
2
2