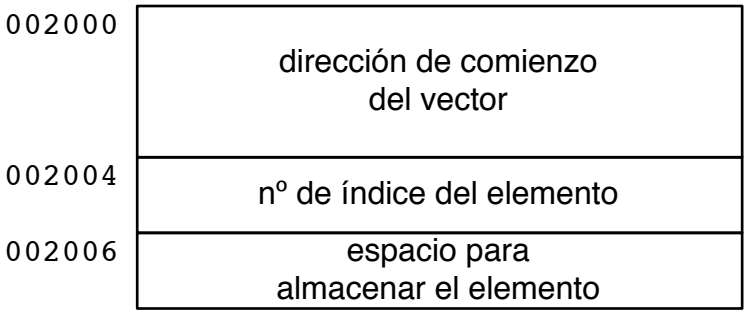
Arquitectura de computadores
Práctica 2: Empezando a programar

1. Modos de direccionamiento.

En la primera parte de esta práctica se trata de familiarizarnos con el uso de algunos modos de direccionamiento. Para ello se plantea el siguiente ejercicio:

Supongamos que disponemos de un vector (o *array*) en alguna zona de la memoria, cuyos elementos son de tamaño word (16 bits). Por otro lado, se cuenta con la siguiente estructura en memoria (por ejemplo, en la dirección 2000_{16}:



El primer componente de esa estructura (de tamaño long word) contiene la dirección del primer elemento del vector; el segundo elemento contiene el índice (tamaño word) del elemento al que se desea acceder; por último, el tercer componente es un espacio donde se almacenará una copia del elemento del vector especificado con los dos campos anteriores.

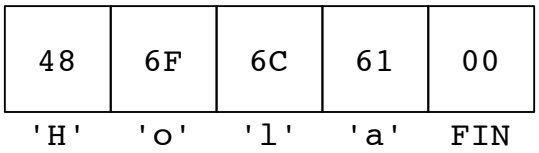
Se trata de escribir las instrucciones necesarias para copiar el elemento indicado en dicha estructura al espacio reservado en ella.

Se recomienda inicializar la estructura y el mismo vector (con los elementos que se desee) empleando las pseudoinstrucciones `DC` y `DS` apropiadas.

2. Ciclos.

A continuación realizaremos un sencillo ejercicio en el que se emplea un ciclo con final por doble condición.

En algunos lenguajes de programación (como por ejemplo en lenguaje C) las cadenas de caracteres se representan como secuencias de bytes (con una codificación de caracteres como ASCII, ISO-8859 o UTF-8), indicando el final de la cadena con un byte a cero. Ejemplo: la cadena `'Hola'` se representaría así:



Se trata de escribir un programa que, partiendo de una cadena previamente almacenada en memoria, la copie a otra zona de memoria reservada para ello, zona que tendrá un tamaño máximo de 64 bytes. Nótese que, en caso de que la cadena a copiar sea más larga, deberán copiarse sus primeros 64 bytes; es decir, el ciclo de copia debe terminar al alcanzar el final de la cadena o el límite de longitud máxima.

3. Subrutinas.

En la última parte de esta práctica realizaremos unos sencillos ejercicios que impliquen el uso de subrutinas.

Servicios del monitor.

Para empezar, vamos a ver unos SERvicios del MONitor (o *sermones*) del 68fil que nos resultarán de utilidad en esta práctica. El monitor del 68fil ofrece trece sermones, todos ellos accesibles mediante la instrucción `TRAP #15` seguida de una word, la cual determina el número de sermón. Hasta ahora sólo se ha hecho uso del sermón nº 0: el de final de programa.

El sermón `GETS` (nº 3) permite leer una cadena por teclado (a través del mismo puerto serie por el que se conecta el 68fil al PC). Este sermón necesita que se le pase como parámetro en el registro `A0` la dirección de comienzo de un área en memoria donde se almacenará la cadena leída, terminada con un byte 0. El programador debe asegurarse de que exista suficiente espacio reservado en ese área de memoria.

El sermón `PUTS` (nº 4) muestra una cadena en pantalla (a través del mismo puerto de serie de comunicación con el PC). También necesita que se le pase como parámetro en el registro `A0` la dirección de comienzo de la cadena (ya almacenada en memoria y terminada con un byte a 0).

Similar al anterior es el sermón `LNPUTS` (nº 11), que se diferencia de `PUTS` en que envía automáticamente un salto de línea (CR+LF) antes de la cadena apuntada por `A0`.

Otro sermón que resulta útil en muchas ocasiones es `ITOA` (*Integer To ASCII*). Éste toma como parámetro (en el registro `D7`, como long word) un número y construye su representación en forma de cadena de caracteres (y en hexadecimal), almacenando dicha cadena en la zona de memoria apuntada por `A0`. En combinación con `PUTS`/`LNPUTS` podemos así mostrar fácilmente por pantalla valores numéricos. (Importante: el sermón `ITOA` modifica el contenido del registro A0, el cual queda apuntando al byte siguiente al de final de cadena.)

Para utilizar más fácilmente los sermones se puede incluir un archivo que contiene las definiciones de todos los números de sermones así como una macro `SERMON`.

Para incluir dicho archivo en nuestros programas basta con escribir al principio la siguiente línea:

```
INCLUDE /usr/local/68k/sermones.inc
```

/share/

Ejercicios con subrutinas.

No. STRLEN está ya definido aquí
Hay que usar otra etiqueta

1. Escríbase una subrutina ~~STRLEN~~ que, tomando como parámetro de entrada (a través del registro `A0`) la dirección de una cadena, calcule y retorne su longitud en bytes (sin contar el 0 de final de cadena). El valor de retorno también se entregará en un registro (por ejemplo, `D7`). Así mismo, escríbase un programa principal que pida por teclado una cadena y muestre por pantalla su longitud, llamando para ello a la subrutina `STRLEN` y haciendo uso de los sermones `GETS`, `PUTS`/`LNPUTS` e `ITOA`. Ejemplo de ejecución:

   ```
   Teclee una cadena: Hola
   Su longitud es 4
   ```

2. Escríbase una subrutina `STRINV` que, dada una cadena, genere otra en sentido inverso (por ejemplo, a partir de 'Hola' generará la cadena 'aloH'). Dicha subrutina tendrá dos parámetros de entrada: la dirección de comienzo de la cadena a invertir en `A0`, y la dirección a partir de la cual se almacenará la cadena invertida en `A1`. Así mismo, escríbase un programa principal que pida por teclado una cadena y muestre por pantalla la cadena resultante de invertirla, llamando para ello a `STRINV` y haciendo uso de los sermones necesarios. Ejemplo:

   ```
   Teclee una cadena: Hola
   aloH
   ```

3. Escríbase una subrutina `MAXMIN` que determine y retorne los valores máximo y mínimo de un vector de números de 16 bits con signo. Sus parámetros de entrada serán la dirección de comienzo del vector (long) y su número de elementos (word), y se pasarán a través de la pila. El valor máximo se retornará en la word baja del registro `D7`, y el mínimo en la word alta del mismo registro. Escríbase también un programa principal que se limite a llamar a `MAXMIN` empleando un vector ya almacenado en memoria, y que muestre por pantalla los valores máximo y mínimo.