HOJA DE PROBLEMAS DEL TEMA 4.
PROGRAMACIÓN EN ENSAMBLADOR DE MIPS
CARACTERES Y CADENAS DE CARACTERES

---

1. Diseñar un programa en ensamblador de MIPS que encuentre el número de veces que aparece un carácter dado en una cadena de caracteres ASCII de longitud desconocida. Se podrá partir de la siguiente versión de la solución en C:

```
int veces;
char string[32], caracter;
register int i;
int main (void) {
      veces = 0;
      i = 0;
      while (string[i] != '\0') {
            if (string[i] == character)
                  veces++;
            i++;
      }
      return 0;
}
```

Las variables ***veces***, ***string*** y ***caracter*** existirán en la solución con el mismo nombre, y tendrán un contenido inicial desconocido. Excepcionalmente, para hacer pruebas se podrá dar un contenido inicial conocido a ***string*** y ***caracter***. Escribir la solución en el fichero **strcharacter.asm**.

---

2. Diseñar un programa en ensamblador de MIPS que, convierta las letras minúsculas de una cadena de caracteres a mayúsculas. Se puede partir del siguiente código en C:

```
char string[32];
register int i;
register char caracter;
int main (void) {
      i = 0;
      while ( (caracter = string[i]) != '\0' ) {
            if ( (caracter >= 'a') && (caracter <= 'z') )
                  string[i] = caracter -'a' + 'A';
            i++;
      }
      return 0;
}
```

En la solución la variable ***string*** existirá en la solución con el mismo nombre, y tendrá un contenido inicial desconocido. Excepcionalmente, para realizar las pruebas se la podrá dar un contenido inicial. Escribir la solución en el fichero **strtoupper.asm**.

---

3. Diseñar un programa en ensamblador de MIPS que, dada una cadena de caracteres, cuente cuántos son dígitos del '0' al '9'. Se puede partir del siguiente código en C:

```
int digitos;
register int i;
char string[32];
register char caracter;
int main (void) {
      digitos = 0;
      i = 0;
      while ( (caracter = string[i]) != '\0' ) {
            if ( (caracter >= '0') && (caracter <= '9') )
                  digitos++;
            i++;
      }
      return 0;
}
```

En la solución las variables ***digitos*** y ***string*** existirán en la solución con el mismo nombre, y tendrán un contenido inicial desconocido. Excepcionalmente, para realizar las pruebas se podrá dar un contenido inicial conocido a la variable ***string***. Escribir la solución en el fichero **strdigits.asm**.

---

4. Diseñar un programa en ensamblador de MIPS que determine si una cadena de caracteres ASCII es o no palíndroma. Una cadena es palíndroma si se lee igual de derecha a izquierda que de izquierda a derecha. Se puede partir del siguiente código en C:

```
register char *ini, *fin;
char string[32];
char palin;
int main (void) {
      // Buscar el final de la tira
      ini = fin = string;
      while (*fin != '\0')
            fin++;
      fin--;
      // Comparar los caracteres
      palin = 'S';
      while (ini < fin) {
            if (*ini == *fin) {
                  ini++;
                  fin--;
            }
            else {
                  palin = 'N';
                  break;
            }
      }
      return 0;
}
```

En la solución las variables ***palin*** y ***string*** existirán en la solución con el mismo nombre, y tendrán un contenido inicial desconocido. Excepcionalmente, para realizar las pruebas se podrá dar un contenido inicial conocido a la variable ***string***. Escribir la solución en el fichero **strpalin.asm**.

---

5. Diseñar un fragmento de programa en ensamblador de MIPS que convierta una tira de caracteres a un valor numérico de tipo entero. La tira representa una cantidad numérica, y sólo contiene caracteres entre el '0' y el '9' (aparte del carácter nulo que marca el final de la misma). Se puede partir de la siguiente versión en C:

```
int numero;
char string[10];
register int i, cifra; // cifra tiene el ordinal del carácter leído
int main (void) {
      numero = 0;
      i = 0;
      while ( (cifra = string[i]) != '\0' ) {
            numero = numero * 10 + (cifra-'0');
            i = i + 1;
      }
      return 0;
}
```

En la solución las variables ***numero*** y ***string*** existirán en la solución con el mismo nombre, y tendrán un contenido inicial desconocido. Excepcionalmente, para realizar las pruebas se podrá dar un contenido inicial conocido a la variable ***string***. Escribir la solución en el fichero **atoi.asm**.