Objetivo: Comprender la arquitectura MIPS (programación utilizando el ISA de MIPS).

Ejercicio 0.

a. Se encuentra en el segmento de texto de un programa en memoria las siguientes instrucciones en lenguaje máquina (representadas aquí en base 16). Traduzca desde lenguaje máquina de MIPS esas cinco instrucciones. Indique para cada una, de qué formato es, y cómo es su decodificación (desemsamblado) en lenguaje ensamblador.
   0x3c0c0041
   0x258c00e0
   0x918e0004
   0x00000000
   0x25ce001e
b. Explique el proceso (pasos) que realiza mipsx cuando se realiza “click” en “compilar y cargar”. Detalle en su explicación que función cumplen los programas as, ld y gdb (de donde provienen esos programas?), y que relación existe entre mipsx ejecutado en la PC de laboratorio con la máquina MIPS en la red.

Ejercicio 1. Escriba un programa que realice la sumatoria del siguiente arreglo con elementos de tipo byte y almacene su resultado en “resultado”. La longitud es variable, se utiliza un byte `NULL` (con valor cero) para determinar su fin.

```
.data
arreglo1:
	.byte 8, 12, 3, … , 5, 0
resultado:
	.byte 0
.text
#Completar
```

Ejercicio 2. Implemente la multiplicación de “N”x”M”, guardando el resultado en “resultado”, sin utilizar instrucciones que hagan uso del coprocesador matemático (utilizando solo instrucciones de suma y saltos, y sin utilizar mult).

```
.data
N:
	.word 10
M:
	.word 7
resultado:
	.word 0
.text
#Completar
```

Ejercicio 3. Crear un programa que calcule el en enésimo elemento de la sucesión fibonacci, y guarde el resultado en una variable “resultado”. El número del elemento a ser calculado debe ser tomado de la variable “N”. Recuerde que la secuencia de fibonacci está definida como:

F(1) = 1
F(2) = 1
F(n) = F(n-1) + F(n-2), para “n” distinto de 1 y 2.

Los primeros 10 de la sucesión: 1, 1, 2, 3, 5, 8, 13, 21, 34, 55…

Ejercicio 4. Dadas las siguientes declaraciones de la sección de datos, y sabiendo que el segmento de datos se carga a partir de la dirección de memoria 0x0400 0000, y el segmento de texto a partir de la dirección 0x0100 0000.

```
.data
lastLogin:
      .word 1631216321
datos:
      .byte 37
      .half 0x7FFF
      .float 0
nick:
      .asciiz "peperina"
#password (esto es un comentario, no una etiqueta)
      .word 0x68756E74, 0x65723200, 0x00000000
saludo:
.asciiz "Buen dia <nick>!"
longitud:
      .word 0
```

a. Utilizando las instrucciones `lb`, `sw`, instrucciones de salto y cualquier otra que considere necesaria, almacene en `longitud` la cantidad de caracteres del nick (sin incluir el byte cero que marca el fin de la cadena). El código debe seguir funcionando aunque el usuario cambie la longitud del nick.

b. Utilizando las instrucciones `lb`, `sb`, instrucciones de salto y cualquier otra que considere necesaria, copiar la contraseña guardada en `password` a `saludo`. Como la contraseña está representada en ASCII, copie los bytes hasta el primer byte en cero. El código debe seguir funcionando aunque el usuario cambie la longitud de la contraseña ¿Cuál es la contraseña?

Ejercicio 5. Traducir al lenguaje ensamblador de **MIPS** el siguiente fragmento de código en alto nivel, sabiendo que el vector V es un vector de halfs, y el resto de las variables son enteras.

```
for (i=0; i<n-1; i++)
{
   for (j=i+1; j<n; j++)
  {
    if(V[i]>V[j])
    {
      aux = V[i];
      V[i] = V[j];
      V[j] = aux;
    }
  }
}
```