

Evaluación del Laboratorio Número 1

Sistemas con Microprocesadores y Microcontroladores

10/09/2025

Instrucciones

- La presente evaluación es individual y podrá utilizar todo el material escrito del que disponga.
- La misma consta de 2 ejercicios y la duración será de 90 minutos.
- En cada apartado se indica el puntaje que otorga el mismo.
- No se permitirá intercambiar ni prestar ningún tipo de elemento entre los estudiantes.
- Cada ejercicio funcionando deberá ser aprobado por un docente para que se compute como válido antes de comenzar con el siguiente ejercicio.
- El código de los ejercicios debe ser enviados por correo electrónico a las direcciones *mgio-ri@herrera.unt.edu.ar* y *ltek@herrera.unt.edu.ar* antes de finalizar la evaluación.

Ejercicios para resolver

1) 25 Minutos - 30 Puntos

Escriba un programa para encontrar un número en un bloque de datos. El tamaño de cada dato es de 8 bits. El resultado debe ser la posición del número (respecto al inicio del bloque de datos) o -1 en caso de no encontrarlo, el cual se debe almacenar en la dirección **base**. La longitud del bloque está en la dirección **base+1** y el bloque comienza a partir de la dirección **base+2**. El dato a buscar se encuentra en la dirección **base**. Por ejemplo:

Datos	Resultado
(base) = 0x3A	(base) = 0x02
(base + 1) = 0x03	
(base + 2) = 0x2A	
(base + 3) = 0x10	
(base + 4) = 0x3A	

2) 25 Minutos - 30 Puntos

Escriba un programa que cuente el número de bits que están en '1' en una secuencia de datos de 8 bits. La longitud de la secuencia expresada en bytes se encuentra en la dirección **base+1**, y la secuencia de datos comienza en **base+2**. El resultado total (el conteo de bits establecidos de toda la secuencia) debe ser almacenado en la dirección **base**. Por ejemplo:

Datos	Resultado
	(base) = 0x0C
(base + 1) = 0x03	
(base + 2) = 0x2A	
(base + 3) = 0x1F	
(base + 4) = 0x3C	

3) 40 Minutos - 40 Puntos

Escriba un programa para multiplicar dos matrices de 3x3. Las matrices A y B están almacenadas en la memoria de la siguiente manera: la matriz A comienza en la dirección **base**, y la matriz B en **base+9**. El resultado, la matriz $C = A \times B$, debe almacenarse a partir de la dirección **base+18**. Cada elemento de la matriz es un número signado de 8 bits.

Como un recordatorio de Álgebra, la multiplicación de dos matrices 3x3 A y B se define con la siguiente fórmula: $c_{ij} = a_{i1} \cdot b_{1j} + a_{i2} \cdot b_{2j} + a_{i3} \cdot b_{3j}$

Para facilitar la comprobación de los resultados:

$$\begin{bmatrix} 3 & -1 & 2 \\ 2 & 0 & -4 \\ -3 & 5 & 2 \end{bmatrix} \times \begin{bmatrix} 5 & 3 & -3 \\ -4 & 1 & 2 \\ 2 & -1 & 0 \end{bmatrix} = \begin{bmatrix} 23 & 6 & -11 \\ 2 & 10 & -6 \\ -31 & -6 & 19 \end{bmatrix}$$

Soluciones completas

1)

```

.data
base: .byte 0x3A, 0x03, 0x2A, 0x10, 0x3A

.text
.globl main
main:
    la s0, base+2      # Cargo la dirección base + 2 para ahorrar operaciones
    lb s1, -2(s0)      # Cargo el número a buscar
    lb s2, -1(s0)      # Cargo la cantidad de elementos
    li a0, 0           # Cargo 0 en a0 para contar movimientos de memoria
loop:
    addi s2, s2, -1     # Resto 1 a la cantidad de elementos
    lb t0, 0(s0)        # Cargo el elemento
    beq s1, t0, fin     # Si se lo encuentra, ir al final
    addi s0, s0, 1      # Si no se lo encuentra, sumar uno a la dirección
    addi a0, a0, 1      # Sumar uno al contador
    bnez s2, loop       # Si siguen habiendo elementos, volver
noEncontrado:
    li a0, -1          # Si no se lo encuentra, cargar a0
fin:
    la s0, base        # Cargar de vuelta la dirección base
    sb a0, 0(s0)        # Guardar -1 o el contador
    li a0, 10           # Selección del servicio terminar el programa sin error
    ecall              # Termina el programa y retorna al sistema operativo

```

2)

```

.data
base: .byte 0x00, 0x03, 0x2A, 0x1F, 0x3C

.text
.globl main
main:
    la s0, base+2      # Cargo la dirección base + 2 para ahorrar operaciones
    lb s1, -1(s0)      # Cargo la cantidad de elementos
    li a0, 0           # Cargo 0 en a0 para contar 1s
loop:
    addi s1, s1, -1     # Resto 1 a la cantidad de elementos
    lb t0, 0(s0)        # Cargo el elemento actual
    li a1, 8           # Cargo 8 para hacer el loop interno
loopBit:
    addi a1, a1, -1     # Resto 1 al contador
    andi t1, t0, 1      # AND con 1 para ver si está o no en 1
    add a0, a0, t1       # Sumo el resultado de AND al contador
    srli t0, t0, 1      # Shift right uno
    bnez a1, loopBit
    addi s0, s0, 1      # Sumar uno a la dirección
    bnez s1, loop       # Si siguen habiendo elementos, volver
fin:
    la s0, base        # Cargar de vuelta la dirección base
    sb a0, 0(s0)        # Guardar el contador
    li a0, 10           # Selección del servicio terminar el programa sin error
    ecall              # Termina el programa y retorna al sistema operativo

```

3)

```

.data
base: .byte 3, -1, -2
      .byte 2, 0, -4

```

```

.byte -3, 5, 2

.byte 5, 3, -3
.byte -4, 1, 2
.byte 2, -1, 0

.byte 0,0,0,0,0,0,0,0,0

.text
.globl main
main:
    la s0, base          # Cargo la dirección de la matriz A
    la s1, base+9        # Cargo la dirección de la matriz B
    la s2, base+18       # Cargo la dirección de la matriz C
    li s3, 3             # Cargo 3 para multiplicar

    li a0, 0             # Índice de fila actual

# cij = ai1 b1j + ai2 b2j + ai3 b3j
loopI:
    li a1, 0             # Índice de columna actual
loopJ:
    li a3, 0             # Resultado cij

    mul a2, a0, s3        # Multiplico la fila por 3
    add t0, s0, a2        # Dirección de ai1
    lb t1, 0(t0)          # Cargo ai1

    add t2, s1, a1        # Dirección de b1j
    lb t3, 0(t2)          # Cargo b1j

    mul t4, t1, t3        # ai1 b1j
    add a3, a3, t4        # cij = ai1 b1j

    lb t1, 1(t0)          # Cargo ai2
    lb t3, 3(t2)          # Cargo b2j

    mul t4, t1, t3        # ai2 b2j
    add a3, a3, t4        # cij = ai1 b1j + ai2 b2j

    lb t1, 2(t0)          # Cargo ai3
    lb t3, 6(t2)          # Cargo b3j

    mul t4, t1, t3        # ai3 b3j
    add a3, a3, t4        # cij = ai1 b1j + ai2 b2j + ai3 b3j

    sb a3, 0(s2)          # Guardo cij
    addi a1, a1, 1        # Aumento 1 al índice de columna
    addi s2, s2, 1        # Sumo 1 a la dirección de resultado
    bne a1, s3, loopJ     # Si no llegó a 3, seguir a loopJ
    addi a0, a0, 1        # Aumento 1 al índice de fila
    bne a0, s3, loopI     # Si no llegó a 3, saltar a loopI

fin:
    li a0,10             # Selección del servicio terminar el programa sin error
    ecall                # Termina el programa y retorna al sistema operativo

```