Arquitectura del Computador - LCC - FCEIA-UNR

Examen final condición libres - Mesa 19 de febrero de 2026

Nombre y apellido: _____

Legajo: _____

IMPORTANTE: Justificar las respuestas e indicar de manera clara y detallada los procedimientos realizados para llegar a las soluciones.
Horario máximo de entrega: 11:00 hs.

1. (10 puntos) Considere una arquitectura de 32 bits con representación de números enteros en complemento a dos. Dados los siguientes operandos:
   - `A = 0xfffe1c7b`
   - `B = 0x1e306`

   (a) Calcule `A+B` y exprese el resultado en binario, decimal y hexadecimal. Indique si se produce desbordamiento y/o acarreo. Justifique.

   (b) Calcule `A - B` utilizando la representación en complemento a dos (esto es, `A+(-B)`) y exprese el resultado en binario, y hexadecimal. Indique si se produce desbordamiento y/o acarreo.

   (c) Realice un corrimiento a la derecha de `A` (`A>>16`) y un corrimiento a la derecha de `B` (`B>>12`), y exprese el resultado en binario, decimal y hexadecimal.

   (d)) Calcule las operaciones `A&B`, `A|B` y `A^B`, y exprese cada resultado en binario, decimal y hexadecimal.

2. (10 puntos) Dados los siguientes números decimales: $x = 15.25$, $y = 15.5$.

   (a) Representarlos en formato IEEE 754 de precisión simple. Indique claramente el signo, el exponente y la fracción en cada caso.

   (b) Realizar la operación $z = x \ominus y$ en aritmética IEEE 754 de precisión simple.

   (c) Convertir el resultado a su representación en formato IEEE 754 de doble precisión.

3. (30 puntos) Escribir un programa en lenguaje Assembler x86-64 que calcule la suma parcial de una serie utilizando exclusivamente aritmética entera. En particular, se desea calcular la siguiente serie geométrica:

$$S = \sum_{n=0}^{N-1} x^n$$

donde x es un número entero de 32 bits y N la cantidad de términos de la serie.

Requerimientos

1. El programa debe recibir por línea de comandos:
   - Un número entero x de 32 bits.
   - Un número entero $N \geq 1$.
2. El cálculo debe realizarse utilizando únicamente instrucciones de aritmética entera.

3. La suma debe calcularse de forma iterativa:

$$S = 1 + x + x^2 + \cdots + x^{N-1}$$

4. No se permite utilizar rutinas externas de potencia.
5. Cada término debe obtenerse a partir del anterior:

$$T_n = T_{n-1} \cdot x$$

6. Implementar:
   - Una subrutina que calcule la suma de la serie.
   - Control de overflow (indicarlo por pantalla si ocurre).
7. Mostrar el resultado final por pantalla en formato entero. Por ejemplo, si se ejecuta

   `./serie 5 6`

   se debe imprimir lo siguiente:

```
x: 5
Cantidad de terminos: N=6
1 5 25 125 625 3125
Suma=3906
```

Ayuda:

- El primer término de la serie es 1. Puede inicializarse el acumulador con dicho valor y actualizar cada término mediante multiplicaciones sucesivas.
- Puede utilizarse la función `atoi` para convertir los argumentos de entrada. Dicha función recibe como parámetro un puntero a `char` (por ejemplo, `argv[1]`) y devuelve el valor entero correspondiente.

4. (30 puntos) Implementar en lenguaje Assembler ARM un programa que invoque una función encargada de realizar una conversión de escala lineal sobre un arreglo de números en punto flotante de precisión simple. La conversión de escala está dada por la expresión:

$$y_i = a \cdot x_i + b$$

donde:

- x_i representa el elemento original del arreglo en formato de punto flotante.
- y_i representa el valor escalado en formato de punto flotante.
- $a = 2056$ es el factor de escala entero.
- $b = 11.25$ es el desplazamiento (*offset*) en punto flotante.

Requerimientos:

1. El programa principal debe invocar la función de conversión de escala.
2. La función debe recibir como argumentos (según la convención de llamadas):
   - La dirección base del arreglo de entrada X.
   - La dirección base del arreglo de salida Y.
   - La cantidad de elementos N.
   - El factor de escala a.
   - El desplazamiento b.
3. La función debe:

- Recorrer el arreglo elemento a elemento.
- Aplicar la conversión $y_i = a \cdot x_i + b$ a cada elemento.
- Almacenar cada resultado en el arreglo de salida.
- Imprimir cada resultado por pantalla.

5. (20 puntos) Se tiene un sistema de memoria virtual con paginación de dos niveles que puede direccionar un total de 2^{32} bytes. El primer nivel utiliza 9 bits y el segundo nivel 11 bits. El sistema cuenta con 4 GB de memoria principal y páginas físicas de 4 KB.
   (a) ¿Cuántos bits tienen las direcciones virtuales?
   (b) Indicar cómo están compuestas las direcciones virtuales.
   (c) ¿Cuántos bits tienen las direcciones físicas?
   (d) Indicar cómo están compuestas las direcciones físicas.
   (e) ¿Cuál es el número máximo de páginas virtuales en el sistema?
   (f) ¿Cuál es el número máximo de páginas físicas en el sistema?
   (g) Calcular el tamaño de la tabla de páginas de primer nivel y el de cada tabla de segundo nivel. Suponer que cada entrada de las tablas de páginas, en ambos niveles, ocupa 4 bytes.
   (h) Indicar el tamaño total de las tablas de paginación de este proceso y compararlo con el correspondiente al caso de paginación de un solo nivel. En caso necesario, redondear el resultado a una cantidad entera de bytes.
   (i) Suponiendo una TLB con 32 entradas totalmente asociativas, calcular su tamaño total. Cada entrada almacena el número de página virtual, el número de marco físico y un bit de validez. En caso necesario, redondear el resultado a una cantidad entera de bytes.
   (j) Atendiendo al contenido de la TLB y de las tablas de páginas de un proceso mostradas a continuación, indicar la dirección de memoria física accedida para las siguientes direcciones virtuales:
      1. `0x7fff028`
      2. `0xffffeb98`
      3. `0x2002060`

TLB

V	Nro. Pág. Virtual	Nro. Pág. Física
0		
1	0x7fff	0x5000
⋮	⋮	⋮
1	0x7ffe	0x5100
0		

Tabla de páginas de primer nivel

V	Dirección	
1	0x8000	0x1ff
	⋮	⋮
1	0x8400	
		0x0

Tablas de páginas de segundo nivel

V	Nro. Pág. Física	
		0x7ff
1	0x410	
	⋮	⋮
1	0x405	
		0x0

0x8400 →

V	Nro. Pág. Física	
		0x7ff
1	0x3ff	
	⋮	⋮
1	0x402	
		0x0

0x8000 →

Ejercicio	1	2	3	4	5	Total
Puntos	10	10	30	30	20	100
Puntos obtenidos						