R-222 Arquitectura del Computador - LCC

Examen final condición regular/libre

2 de marzo de 2023

Nombre y apellido: _____

Legajo: _____

IMPORTANTE: Justificar las respuestas e indicar de manera clara y detallada los procedimientos realizados para llegar a las soluciones.

1. (20 puntos) Dado el siguiente código en Assembler x86-64:

```
movb $254, %al            # al = ...........
addb %al, %al             # al = ...........   CF=..... OF=.....
addb $0x80, %al           # al = ...........   CF=..... OF=.....
imulb %al                 # ax = ...........
```

   (a) Indicar el valor del subregistro `al`, luego de ejecutarse las líneas correspondientes, en decimal, en octal y en hexadecimal.

   (b) Indicar si hubo acarreo y desbordamiento luego de realizarse las operaciones indicadas.

   (c) Indicar si el resultado de las sumas es correcto, analizando tanto el caso `signed` como `unsigned`.

   (d) Indicar el valor del subregistro `ax` luego de ejecutarse la instrucción `imulb %al`.

2. (40 puntos) Escribir un código en Assembler x86-64 equivalente al siguiente código en lenguaje C:

```
#include <stdio.h>

void producto(int list1[], int list2[], int n);

int list1[5]={1, 2, 3, 4, 5};
int list2[5]={2, 3, 4, 5, 6};

int main(){
    int n=5;
    producto(list1, list2, n);
    return 0;
}

void producto(int list1[], int list2[], int n){
    for(int i=0; i<n; i++)
        printf("%d ", list1[i]*list2[i]);
}
```

   tal que imprima por pantalla el producto de ambos vectores componente a componente. Es decir, con los valores de los arreglos mostrados debería imprimir `2 6 12 20 30`.

3. (40 puntos) Dado el siguiente código en lenguaje c que implementa el procedimiento denominado "Método del campesino ruso" para multiplicar dos valores enteros:

```
unsigned int campesino_ruso(unsigned int i, unsigned int j)
{
    unsigned int res = 0;
    while (j > 1) {
        if (j & 1) {
            res += i;
            j -= 1;
        } else {
            i *= 2;
            j /= 2;
        }
    }
    return res + i;
}
```

Completar el siguiente código equivalente en Assembler ARM:

```
campesino_ruso:
    .....   r1, #1
    .....   .L6
    .....   r3, #0
.L5:
    .....   r1, #1
    .....   r3, r3, r0
    .....   r1, r1, #1
    .....   r0, r0, #1
    .....   r1, r1, #1
    .....   r1, #1
    .....   .L5
.L2:
    .....   r0, r0, r3
    bx      lr
.L6:
    .....   r3, #0
    .....   .L2
```

4. (40 puntos) Dado el siguiente código en Assembler x86-64:

```
    .data
    a:      .word 0x6000
            .word 100
            .word 649
            .word 200
    b:      .byte 0xff
            .byte 0xde
            .byte 0xaa
            .byte 127
    f:      .asciz "Hola mundo"

    .text
    .global main
    main:
    movq $a, %rbx
    ........(....), %eax       # eax = 0x7faadeff
    ........(....), %al        # al = 0x64
    ........(....), %ax        # ax = 0x7faa
    ........(....), %eax       # eax = 0xc8028900
    ............., %rbx       # rbx = 0x40400b
```

```
........(....), %rcx        # rcx = 0x40400d
retq
```

(a) Realizar el diagrama de memoria byte a byte, expresando los valores en hexadecimal e indicando las direcciones de memoria correspondientes. Asumir que la variable `a` está alojada en la dirección `0x404000`.

(b) Completar lo que falta en las líneas de puntos para que cada línea de código se corresponda con el valor del registro indicado en el comentario correspondiente. **IMPORTANTE:** No usar movimientos de valores inmediatos.

5. (30 puntos) Dados los siguientes números en punto flotante norma IEEE 754 simple precisión:

   - `A=0x5FBE4000`
   - `B=0x3FF80000`
   - `C=0xDFDE4000`

   (a) Realizar la suma `(A+B)+C`.

   (b) Convertir el resultado a doble precisión.

   (c) Indicar la instrucción para realizar dicha conversión, tanto en Assembler x86-64 como en ARM.

   (d) Analizar el error cometido por representación en el ítem (a) y cómo se modifica si en lugar de usar simple precisión se utiliza doble precisión.

6. (30 puntos) Dado un sistema que emplea un esquema de paginación con las siguientes características:

   - El número de páginas por proceso es como máximo de 8192.
   - El tamaño de página es de 32 KB.
   - La memoria física es de 16 MB (2^{24} bytes).
   - El tiempo de acceso a disco es de 120 ms, a memoria de 50 ns y a la TLB de 10 ns (en todo caso).

   (a) ¿Cuál es el formato de las direcciones virtuales? Especificar el tamaño de los campos y su significado.

   (b) Indicar el tamaño del espacio direccionable virtualmente.

   (c) ¿Qué es la TLB? ¿Cuál es su función?

   (d) Atendiendo al contenido de la TLB y de la tabla de páginas de un proceso mostradas a continuación, indicar la dirección de memoria accedida para las siguientes direcciones virtuales:

   1. 0x001A007
   2. 0x0007100
   3. 0x00140C2

   Tener en cuenta que las entradas de la tabla de paginación tienen 3 bits de control (los tres bits más significativos), de los cuales los dos más significativos se corresponden al bit de presencia y al bit de modificación, respectivamente.

TLB

2	0xA1C
5	0xC04
1	0x043

Tabla de paginación

0	0xFF1
1	0x043
2	0xA1C
3	0x203
4	0x76A
5	0xC04
6	0x60F
7	0x663
8	0x011

(e) Atendiendo a las direcciones virtuales del apartado anterior, indicar si se debe producir acceso a disco en cada caso. ¿Por qué?

(f) Calcular el tiempo de acceso para cada una de las direcciones virtuales del apartado (d).

Ejercicio	1	2	3	4	5	6	Total
Puntos	20	40	40	40	30	30	200
Puntos obtenidos							