



ORGANIZACIÓN DE COMPUTADORAS
Departamento de Ciencias e Ingeniería de la Computación
Universidad Nacional del Sur
Segundo Cuatrimestre de 2017



Segundo Examen Parcial		
Lic. en Ciencias de la Computación – Ing. en Computación – Ing. en Sistemas de Información		
Apellido y Nombre: (en ese orden)	LU:	Hojas entregadas: (sin enunciado)
Profesor:		
NOTA: Resolver los ejercicios en hojas separadas. Poner nombre, LU y número en cada hoja.		

Apague cualquier dispositivo electrónico en su poder y manténgalo guardado. No puede utilizar auriculares. Lea todo el ejercicio antes de comenzar a desarrollarlo.

Ejercicio 1. Asumiendo que se cuenta en todos los casos con las instrucciones **add** y **mpy**. Encontrar una secuencia de instrucciones que resulte óptima en tiempo de ejecución (es decir, que minimice la cantidad de accesos a memoria), y cuya ejecución tenga como resultado la evaluación de la siguiente expresión aritmética:

$$B = (A \times (D + C)) + (A \times (D + C)^2)$$

Las etiquetas denotan las *direcciones de memoria* que contienen los valores sobre los que se quiere operar

- a) Asumiendo una arquitectura de 0–direcciones (tipo pila), con las instrucciones **push** y **pop** para acceder a memoria y la instrucción **dup** que duplica el tope de la pila. Determinar la cantidad de instrucciones y la profundidad de la pila alcanzada.
- b) Asumiendo una arquitectura estilo RISC con operaciones registro a registro, sin limitaciones en cuanto a los registros disponibles, y las instrucciones **ld** (load) y **st** (store) para acceder a memoria, y la instrucción **lda** (load address). Las operaciones aritméticas operan con dos operandos (dst/fte, fte). Indicar la cantidad de accesos a memoria realizados.
- c) Asumiendo una arquitectura tipo INTEL con operaciones 1–dirección más registros, sin limitaciones en cuanto a los registros disponibles, que en lugar de load y store cuenta con la instrucción **mov** para acceder a memoria y donde las operaciones aritméticas operan con tres operandos (dst, fte, fte). Indicar la cantidad de accesos a memoria requeridos.

Ejercicio 2. En el marco de la norma IEEE 754, considerando la representación en punto flotante de media precisión: mantisa fraccionaria en signo magnitud con hidden bit, exponente en exceso y base 2 y la siguiente distribución de bits:

Sig (1b)	Exponente (8 bits)	Mantisa (10 bits)
----------	--------------------	-------------------

Dados los números $X = (1\ 10110\ 0011111001)$ e $Y = (0\ 00111\ 1000111100)$, realizar el producto $X \times Y$ aplicando redondeo por proximidad hacia los pares y hacia $+\infty$, explicando cada uno de los pasos involucrados e indicando claramente qué se hace con los bits G, R y S del

resultado y con R y S al redondear. El resultado debe ser expresando según la representación enunciada.

Ejercicio 3. En el marco de la norma IEEE 754, considerando una representación simplificada de números en punto flotante con mantisa fraccionaria en signo magnitud con hidden bit, exponente en exceso y base 2 y la siguiente distribución de bits:

Sig (1b)	Exponente (5b)	Mantisa (10b)
----------	----------------	---------------

Dados los números:

$$X = (0 \ 01101 \ 0010110101) \quad Y = (0 \ 01110 \ 1101000110)$$

Realizar la suma $X + Y$ aplicando redondeo por *proximidad unbiased (hacia los pares)*, explicando cada uno de los pasos involucrados e indicando claramente qué se hace con los bits G, R y S del resultado y con R y S al redondear. **El resultado debe ser expresando según la representación enunciada.**

Ejercicio 4. Determinar cuál es el contenido final de cada uno de los registros y posiciones de memoria involucrados en la siguiente secuencia de instrucciones. Indicar en cada caso, el número de instrucción que origina cada cambio. Asumir que el primer operando es el destino y el segundo la fuente de información para la operación.

(1) mov R1, #0200	Interpretación
(2) mov (R1), #0100	#xxxx Inmediato
(3) mov 0100(R1), R1	R Registro
(4) mov R2, #0500	(R) Registro indirecto
(5) mov @0100(R1), #0500	xxxx Absoluto
(6) mov (0200), 0300	xxxx(R) Indexado
(7) mov R3, 0200	(xxxx) Memoria indirecto
(8) mov R3, @0100(R3)	@xxxx(R) Pre-indexado indirecto

Ejercicio 5. Considerando el siguiente programa para la arquitectura OCUNS, en la que toda lectura/escritura sobre la dirección FF es redireccionada a la E/S estándar:

LDA R0, FFh	
LOAD R1, 0(R0)	
LOAD R2, 0(R0)	
XOR R3, R3, R3	
LDA R4, 1b13	
JZ R1, 1b13	
JZ R2, 1b13	
SUB R5, R1, R2	
JG R5, 1b12	
1b11: ADD R3, R3, R2	
DEC R1	
JZ R1, 1b13	
JMP R4	
1b12: ADD R3, R3, R1	
DEC R2	
JZ R2, 1b12	
1b13: STORE R3, 0(R0)	
HLT	

OP.	DESCR.	FORM.	PSEUDOCÓDIGO
0	add	I	$R[d] \leftarrow R[s] + R[t]$
1	sub	I	$R[d] \leftarrow R[s] - R[t]$
2	and	I	$R[d] \leftarrow R[s] \& R[t]$
3	xor	I	$R[d] \leftarrow R[s] \wedge R[t]$
4	lsh	I	$R[d] \leftarrow R[s] \ll R[t]$
5	rsh	I	$R[d] \leftarrow R[s] \gg R[t]$
6	load	I	$R[d] \leftarrow \text{mem}[\text{offset} + R[s]]$
7	store	I	$\text{mem}[\text{offset} + R[d]] \leftarrow R[s]$
8	lda	II	$R[d] \leftarrow \text{addr}$
9	jz	II	if $(R[d] == 0)$ $PC \leftarrow PC + \text{addr}$
A	jg	II	if $(R[d] > 0)$ $PC \leftarrow PC + \text{addr}$
B	call	II	$R[d] \leftarrow PC; PC \leftarrow \text{addr}$
C	jmp	III	$PC \leftarrow R[d]$
D	inc	III	$R[d] \leftarrow R[d] + 1$
E	dec	III	$R[d] \leftarrow R[d] - 1$
F	hlt	III	exit

FORMATO	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
I	0	×	×	×	dest. d				src. s				src. t / off.			
II	1	0	×	×	dest. d				address addr							
III	1	1	×	×	dest. d				-							

- Ensamblar el programa a partir de la dirección 00h.
- Si se reubicara el código máquina obtenido en el inciso (a) a partir de la dirección 20h, ¿qué referencias a memoria requieren ser ajustadas? Justificar adecuadamente.
- Suponiendo que los valores ingresados por teclado son 1Ah y 04h, realice una traza mostrando la evolución del contenido de cada registro, para luego, describir el propósito del programa en su conjunto.
- ¿Qué sucede con el resultado retornado si los valores ingresados fueran 04h y 1Ah? ¿Cuál es la diferencia? ¿Existe alguna restricción para los datos de entrada en cuanto al correcto funcionamiento del programa?