

Apellido y nombre	L.U.

## ORGANIZACIÓN DEL COMPUTADOR 2 - Parcial

Segundo Cuatrimestre 2021

Ej.1	Ej.2	Ej.3	Ej.4	Nota

Correctorx:

### Aclaraciones

- Anote apellido, nombre, LU en *todos* los archivos entregados.
- El parcial es domiciliario y todos los ejercicios deben estar aprobados para que el parcial se considere aprobado. Hay dos fechas de entrega, en ambos casos el conjunto de ejercicios a entregar es el mismo. En la primera instancia deberán defender su trabajo frente a su tutorx, quien les ayudará también a encaminar el trabajo de los ejercicios pendientes, si los hubiera.
- El link de entrega es: <https://forms.gle/o7apfGWzwBkb8Ad36>. Ante cualquier problema pueden subirlo también al campus o comunicarse con la lista docente.
- La fecha límite de entrega es el lunes 8 de noviembre a las 21:00. El coloquio será el martes 16 de noviembre durante el horario de cursada (17:00 a 22:00). En caso de precisar re entregar el parcial se podrá hacer hasta el viernes 19 de noviembre a las 21:00. La devolución de la re entrega se hará el día el jueves 25 de noviembre en el horario de cursada (17:00 a 22:00).
- Todas las respuestas deben estar correctamente justificadas y escritas en formato digital (el que prefieran).

## Introducción

Este parcial está dividido en cuatro ejercicios. Los dos primeros tienen que ver con la convención de llamada para 64 y 32 bits respectivamente. En cada función deberán encontrar un único error en la implementación. El tercer ejercicio les presenta una pregunta de system programming como las que ya respondieron para los talleres. El cuarto ejercicio espera que diseñen y expliquen el funcionamiento del mecanismo pedido utilizando lo aprendido en los talleres de system programming como herramienta.

**Nota:** Intenten justificar sus respuestas con precisión y claridad, expresando no sólo la información mínima sino todo aquello que consideren necesario para guiar a la persona que corrija o lea su parcial de forma ordenada y cohesiva. De igual manera si desean realizar consultas por correo les recomendamos que utilicen el tiempo que sea necesario para ordenar y desarrollar sus inquietudes, ya que este proceso permitirá mejorar la comunicación por un lado pero también probablemente sea de gran ayuda a la hora de entender mejor los temas a evaluar/ejercitar.

## Ejercicios

### 1. Ejercicio 1 - Convención C (64 bits)

En el siguiente ejercicio deben encontrar un único error en cada función, uno en `alternate_sum_5` y uno en `alternate_sum_9` entendiendo que su implementación se ha hecho sobre 64 bits. Las funciones deben realizar una suma alternada de cinco o nueve parámetros respectivamente:

$$\text{alternate\_sum\_5}(x_1, x_2, x_3, x_4, x_5) = x_1 - x_2 + x_3 - x_4 + x_5$$

$$\text{alternate\_sum\_9}(x_1, x_2, x_3, x_4, x_5, x_6, x_7, x_8, x_9) = x_1 - x_2 + x_3 - x_4 + x_5 - x_6 + x_7 - x_8 + x_9$$

Pueden consultar los archivos y clases de la práctica y la teórica para recordar qué condiciones debe respetar una función para cumplir con la convención de llamada de 64 bits. En el correo que recibieron junto con el enunciado encontrarán un archivo `archivos_incisol.tar.gz` conteniendo las fuentes necesarias para compilar y diagnosticar el error. No hace falta entregar los archivos, sólo el diagnóstico del error.

**Nota:** Los errores pueden o no tener que ver con la convención de llamada. Pueden suponer que la función `save_result(char *filename, uint32_t result)` guarda el resultado `result` en el archivo ubicado en `filename`.

```
##### LISTA DE FUNCIONES IMPORTADAS
extern save_result

##### SECCION DE DATOS
section .data
```

```

filename_5: db 'results_5',0
filename_9: db 'results_9',0

;##### SECCION DE TEXTO (PROGRAMA)
section .text

;##### LISTA DE FUNCIONES EXPORTADAS
global alternate_sum_5
global alternate_sum_9

;##### DEFINICION DE FUNCIONES
; la funcion debe devolver el resultado de  $x1 - x2 + x3 - x4 + x5$ 
; uint32_t alternate_sum_5(uint32_t x1, uint32_t x2, uint32_t x3, uint32_t x4, uint32_t
    x5);
; registros: x1[rdi], x2[rsi], x3[rdx], x4[rcx], x5[r8]
alternate_sum_5:
    ;prologo
    ;recordar que si la pila estaba alineada a 16 al hacer la llamada
    ;con el push de RIP como efecto del CALL queda alineada a 8
    push rbp
    mov rbp, rsp

    sub rsp, 8
    xor rax, rax    ; limpiamos rax
    add eax, edi    ; realizamos las sumas alternadas
    sub eax, esi

    add eax, edx
    sub eax, ecx
    add eax, r8d

    mov r12d, eax
    mov rdi, filename_5
    mov esi, eax
    call save_result
    mov eax, r12d

    ;epilogo
    add rsp, 8

    pop rbp
    ret

; la funcion debe devolver el resultado de  $x1 - x2 + x3 - x4 + x5 - x6 + x7 - x8 + x9$ 
; uint32_t alternate_sum_9(uint32_t x1, uint32_t x2, uint32_t x3, uint32_t x4, uint32_t
    x5, uint32_t x6, uint32_t x7, uint32_t x8, uint32_t x9);
; registros y pila: x1[rdi], x2[rsi], x3[rdx], x4[rcx], x5[r8], x6[r9], x7[rbp + 0x10],
    x8[rbp + 0x18], x9[rbp + 0x20]
alternate_sum_9:
    ;prologue
    push rbp
    mov rbp, rsp
    push rbx
    push r12
    push r13
    push r14
    push r15

    sub rsp, 8

    lea rbx, [rbp + 0x10]    ; rbx <- rbp + 16
    mov r10d, [rbx]         ; r10d <- x7
    lea rbx, [rbp + 0x18]    ; rbx <- rbp + 24
    mov r11d, [rbx]         ; r11d <- x8
    lea rbx, [rbp + 0x20]    ; rbx <- rbp + 32
    mov r12d, [rbx]         ; r12d <- x9

    xor rax, rax    ; limpiamos rax
    add eax, edi    ; sumamos todos los elementos
    sub eax, esi
    add eax, edx
    sub eax, ecx

```

```
add eax, r8d
sub eax, r9d
add eax, r10d
sub eax, r11d
add eax, r12d    ; el resultado queda en eax

mov r13d, eax
mov rdi, filename_9
mov esi, eax
call save_result
mov eax, r13d

;epilogue
add rsp, 8

pop r14
pop r15
pop r13
pop r12
pop rbx

pop rbp
ret
```

---

## 2. Ejercicio 2 - Convención C (32 bits)

En el siguiente ejercicio deben encontrar un único error en cada función, uno en `alternate_sum_5` y uno en `alternate_sum_9` entendiendo que su implementación se ha hecho sobre **32 bits**. Las funciones deben realizar una suma alternada de cinco o nueve parámetros respectivamente:

$$\text{alternate\_sum\_5}(x_1, x_2, x_3, x_4, x_5) = x_1 - x_2 + x_3 - x_4 + x_5$$

$$\text{alternate\_sum\_9}(x_1, x_2, x_3, x_4, x_5, x_6, x_7, x_8, x_9) = x_1 - x_2 + x_3 - x_4 + x_5 - x_6 + x_7 - x_8 + x_9$$

Pueden consultar los archivos y clases de la práctica y la teórica para recordar qué condiciones debe respetar una función para cumplir con la convención de llamada de 32 bits. En el correo que recibieron junto con el enunciado encontrarán un archivo `archivos_inciso2.tar.gz` conteniendo las fuentes necesarias para compilar y diagnosticar el error. No hace falta entregar los archivos, sólo el diagnóstico del error.

```
##### LISTA DE FUNCIONES IMPORTADAS
extern save_result

##### SECCION DE DATOS
section .data

filename_5: db 'results_5',0
filename_9: db 'results_9',0

##### SECCION DE TEXTO (PROGRAMA)
section .text

##### LISTA DE FUNCIONES EXPORTADAS
global alternate_sum_5
global alternate_sum_9

##### DEFINICION DE FUNCIONES
; la funcion debe devolver el resultado de x1 - x2 + x3 - x4 + x5
; uint32_t alternate_sum_5(uint32_t x1, uint32_t x2, uint32_t x3, uint32_t x4, uint32_t
    x5);
; registros: x1[ebp+0x08], x2[ebp+0x0C], x3[ebp+0x10], x4[ebp+0x14], x5[ebp+0x18]
alternate_sum_5:
    ;prologo
    push ebp
    mov ebp, esp

    xor eax, eax    ; limpiamos eax
    add eax, [ebp+0x08]    ; realizamos las sumas alternadas
    sub eax, [ebp+0x0C]

    add eax, [ebp+0x10]
    sub eax, [ebp+0x14]
    add eax, [ebp+0x18]

    mov ebx, eax
    push eax
    push filename_5
    call save_result
    add esp, 8
    mov eax, ebx

    ;epilogo

    pop ebp
    ret

; la funcion debe devolver el resultado de x1 - x2 + x3 - x4 + x5 - x6 + x7 - x8 + x9
; uint32_t alternate_sum_9(uint32_t x1, uint32_t x2, uint32_t x3, uint32_t x4, uint32_t
    x5, uint32_t x6, uint32_t x7, uint32_t x8, uint32_t x9);
; registros: x1[ebp+0x08], x2[ebp+0x0C], x3[ebp+0x10], x4[ebp+0x14], x5[ebp+0x18], x6[
    ebp+0x1C], x7[ebp+0x20], x8[ebp+0x24], x9[ebp+0x28]
alternate_sum_9:
    ;prologue
    push ebp
    mov ebp, esp
    push ebx
```

```
push esi
push edi

xor eax, eax    ; limpiamos eax
add eax, [ebp + 0x08]    ; sumamos todos los elementos
sub eax, [ebp + 0x0C]
add eax, [ebp + 0x10]
sub eax, [ebp + 0x14]
add eax, [ebp + 0x18]
sub eax, [ebp + 0x1C]
add eax, [ebp + 0x20]
sub eax, [ebp + 0x24]
add eax, [ebp + 0x28]    ; el resultado queda en eax

mov ebx, eax
push eax
push filename_9
call save_result
add esp, 8
mov eax, ebx

;epilogue

pop esi
pop edi
pop ebx

pop ebp
ret
```

---

### 3. Ejercicio 3 - Temas de system programming

Este ejercicio les presenta una pregunta del mismo estilo que ya respondieron durante los talleres de la materia.

**Nota:** Tanto en este ejercicio como en el próximo pueden suponer que el contexto de uso/ejecución de los elementos involucrados es análogo al del kernel desarrollado durante los talleres.

**Pregunta:** ¿Cuál es la unidad de direccionamiento en 32 bits? ¿Y en 64 bits?

### 4. Ejercicio 4 - Mecanismos de system programming

Este ejercicio describe un mecanismo que deseamos implementar o utilizar. Supone que cuentan con un entendimiento suficientemente bueno de los temas de system programming como para diseñar una solución o mecanismo que satisfaga las características presentadas. En este punto se espera que utilicen los temas estudiados como herramientas del diseño de sistemas entendido como un proceso creativo.

**A resolver:**

Suponiendo que las siguientes funciones se utilizan para implementar un mecanismo de **page swapping**:

- `void pedir_pagina_menos_usada(uint16_t *tss_sel, paddr_t *pagina_menos_usada)` devuelve por referencia el selector de tarea y dirección física de la página de usuario menos usada (según algún criterio) dentro del kernel.
- `void escribir_a_disco(uint16_t tss_sel, paddr_t pagina_desalojada)` escribe en el disco el contenido de la página que comienza en `pagina_desalojada` asociada a la tarea cuyo selector es `tss_sel`.
- `paddr_t escribir_a_memoria(uint16_t tss_sel, vaddr_t pagina_pedida)` escribe en memoria el contenido asociado a `pagina_pedida`, que ya había sido guardada en disco, para la tarea cuyo selector es `tss_sel` y devuelve la dirección física de la página.

¿Cómo implementarían la función `pedir_pagina_menos_usada`? ¿Qué estructuras utilizarían? ¿Cómo podrían representar qué tanto se ha usado la página? ¿Cuándo se inicializan las estructuras auxiliares?

---

Esperamos que la experiencia de resolución del examen sea grata y formativa, saludos y buenos deseos de parte del equipo docente.