

[Página Principal](#) / [Mis cursos](#) / [SistOp20](#) / [Prácticos](#) / [Parcial 1: Virtualización](#)

Comenzado el Tuesday, 20 de October de 2020, 14:00

Estado Finalizado

Finalizado en Tuesday, 20 de October de 2020, 15:59

Tiempo empleado 1 hora 59 minutos

Calificación 6,25 de 10,00 (63%)

Pregunta **1**

Finalizado

Sin calificar

El fork es:

Seleccione una:

- a. Ricky.
- b. Una syscall de UNIX que crea un proceso con la misma imagen de memoria pero en un espacio disjunto.
- c. Tenedor.

Respuesta correcta

La respuesta correcta es: Una syscall de UNIX que crea un proceso con la misma imagen de memoria pero en un espacio disjunto.

Pregunta 2

Sin contestar

Puntúa como 2,00

Se cambió ligeramente el *trapframe* donde se mantiene el estado de los registros dentro del *process control block* de **xv6** (1ra columna). Dada la función `mcd()` (2da columna), se la compila a i386 con `-O0` (3ra columna) y con `-O1` (4ta columna).

La opción `-O0` significa sin optimizaciones.

La opción `-O1` significa con optimizaciones básicas.

Para las dos versiones en ensamblador, diga si el *context switch* funciona **siempre, a veces** o **nunca**.

<pre> struct trapframe { uint edi; uint esi; uint ebp; uint oesp; uint ebx; uint ecx; uint eax; uint gs; uint fs; uint es; uint ds; uint trapno; uint err; uint eip; uint cs; uint eflags; uint esp; uint ss; } </pre>	<pre> int mcd(int a, int b) { while(a!=b) { if (a<b) b = b - a; else a = a - b; } return a; } </pre>	<pre> mcd: jmp L2 L4: movl 4(%esp), %eax cmpl 8(%esp), %eax jge L3 movl 4(%esp), %eax subl %eax, 8(%esp) jmp L2 L3: movl 8(%esp), %eax subl %eax, 4(%esp) L2: movl 4(%esp), %eax cmpl 8(%esp), %eax jne L4 movl 4(%esp), %eax ret </pre>	<pre> mcd: movl 4(%esp), %edx movl 8(%esp), %eax cmpl %eax, %edx jne L5 L2: ret L3: subl %eax, %edx L4: cmpl %eax, %edx je L2 L5: cmpl %eax, %edx jge L3 subl %edx, %eax jmp L4 </pre>
--	---	--	--

Seleccione una o más de una:

- a. Con `-O0` (3ra columna): Siempre.
- b. Con `-O0` (3ra columna): A veces.
- c. Con `-O0` (3ra columna): Nunca.
- d. Con `-O1` (4ta columna): Siempre.
- e. Con `-O1` (4ta columna): A veces.
- f. Con `-O1` (4ta columna): Nunca.

Respuesta incorrecta.

Las respuestas correctas son: Con `-O0` (3ra columna): Siempre., Con `-O1` (4ta columna): A veces.

Pregunta **3**

Correcta

Puntúa 1,00 sobre 1,00

Para los siguientes procesos CPU-bound completar la secuencia temporal de como se planifican los procesos en un monoprocesador para una política de planificación RR con *quanto* de 2.

Proceso	$T_{arrival}$	T_{CPU}
A	0	4
B	1	2

Ejemplo de respuesta: AACCAAABB.

Respuesta: 

La respuesta correcta es: AABBAA

Pregunta **4**

Correcta

Puntúa 1,00 sobre 1,00

Para los siguientes procesos CPU-bound completar la secuencia temporal de como se planifican los procesos en un monoprocesador para una política de planificación STCF.

Proceso	$T_{arrival}$	T_{CPU}
A	0	4
B	1	2
C	3	1
D	3	4

Ejemplo de la respuesta: ABBBADD.

Respuesta: 

La respuesta correcta es: ABBCAAADDD

Pregunta **5**

Correcta

Puntúa 0,50 sobre 0,50

Al ejecutar este programa *assembler* i386 usando **memoria segmentada** se produce la siguiente secuencia de accesos a la memoria física: 519, 524, 2044, 527, 529, SegFault, 532.

Deducir el valor del registro de segmentación **base** para el **segmento de código**.

```
7: movl $1020,%ebx
12: movl (%ebx),%eax
15: addl $4, %ebx
17: movl (%ebx),%eax
20: retq
```

Respuesta: 

La respuesta correcta es: 512

Pregunta **6**

Correcta

Puntúa 0,50 sobre 0,50

Al ejecutar este programa *assembler* i386 usando **memoria segmentada** se produce la siguiente secuencia de accesos a la memoria física: 519, 524, 2044, 527, 529, SegFault, 532.

Deducir el valor del registro de segmentación **base** para el **segmento de heap**.

```
7: movl $1020,%ebx
12: movl (%ebx),%eax
15: addl $4, %ebx
17: movl (%ebx),%eax
20: retq
```

Respuesta: 

La respuesta correcta es: 1024

Pregunta 7

Sin contestar

Puntúa como 0,50

Al ejecutar este programa *assembler* i386 usando **memoria segmentada** se produce la siguiente secuencia de accesos a la memoria física: 519, 524, 2044, 527, 529, SegFault, 532.

Deducir el valor del registro de segmentación **límite** para el **segmento de heap**.

```
7: movl $1020,%ebx
12: movl (%ebx),%eax
15: addl $4, %ebx
17: movl (%ebx),%eax
20: retq
```

Respuesta: ✘

La respuesta correcta es: 1024

Pregunta 8

Incorrecta

Puntúa 0,00 sobre 0,50

Considere un tamaño de página de 8 KiB (8192 bytes) y la tabla de páginas lineal de abajo.

Determine que dirección física que se corresponde a la dirección virtual 24579.

Expresar en decimal.

Consejo: no pasar a binario, dividir por el *pagesize*.

VPN	PFN	¿Presente?
15	000	0
14	111	1
13	000	0
12	000	0
11	001	1
10	101	1
9	000	0
8	100	1
7	000	0
6	000	0
5	011	1
4	000	0
3	100	1
2	000	1
1	000	0
0	110	1

Respuesta: ✘

La respuesta correcta es: 32771

Pregunta 9

Incorrecta

Puntúa 0,00 sobre 0,75

Considere un tamaño de página de 8 KiB (8192 bytes) y la tabla de páginas lineal de abajo.

Determine **alguna** dirección virtual que se corresponde a la dirección física 32772.

Expresar en decimal.

Consejo: no pasar a binario, dividir por el *pagesize*.

VPN	PFN	¿Presente?
15	000	0
14	111	1
13	000	0
12	000	0
11	001	1
10	101	1
9	000	0
8	100	1
7	000	0
6	000	0
5	011	1
4	000	0
3	100	1
2	000	1
1	000	0
0	110	1

Respuesta:



La respuesta correcta es: 24580

Pregunta **10**

Correcta

Puntúa 0,75 sobre 0,75

Para un esquema de paginación de i386 (10,10,12), con $PDBR=CR3=0x00000$ y dada la siguiente configuración de memoria, transformar de virtual a física la dirección $0x\text{FFFFFF0C0}$.

Expresar en hexadecimal y con todos los dígitos. Ejemplo: $0x0023ABCD$.

PFN = 0x00000			PFN = 0x00001			PFN = 0x00002		
	PFN	Valid?		PFN	Valid?		PFN	Valid?
0	0x00001	1	0	0x00000	1	0	0xF0F05	0
1	0xFAFAF	0	1	0xC0CA5	0	1	0xF0FA5	0
⋮			⋮			⋮		
1022	0xDEAD0	0	1022	0xCACA5	0	1022	0xF1F0C	0
1023	0x00002	1	1023	0xCAC05	0	1023	0x0000C	1

Respuesta:

La respuesta correcta es: 0x0000C0C0

Pregunta **11**

Correcta

Puntúa 0,75 sobre 0,75

Para un esquema de paginación de i386 (10,10,12), con $PDBR=CR3=0x00000$ y dada la siguiente configuración de memoria, transformar de virtual a física la dirección $0x\text{FFFFFFACA}$.

Expresar en hexadecimal y con todos los dígitos. Ejemplo: $0x0023ABCD$.

PFN = 0x00000			PFN = 0x00001			PFN = 0x00002		
	PFN	Valid?		PFN	Valid?		PFN	Valid?
0	0x00001	1	0	0x00000	1	0	0xF0F05	0
1	0xFAFAF	0	1	0xC0CA5	0	1	0xF0FA5	0
⋮			⋮			⋮		
1022	0xDEAD0	0	1022	0xCACA5	0	1022	0xF1F0C	0
1023	0x00002	1	1023	0xCAC05	0	1023	0x0000C	1

Respuesta:

La respuesta correcta es: 0x0000CACA

Pregunta **12**

Correcta

Puntúa 0,75 sobre 0,75

Para un esquema de paginación de i386 (10,10,12), con $PDBR=CR3=0x00000$ y dada la siguiente configuración de memoria, transformar de virtual a física la dirección **0x00400BA4**.

Expresar en hexadecimal y con todos los dígitos. Ejemplo: 0x0023ABCD.

PFN = 0x00000		
	PFN	Valid?
0	0x00001	1
1	0xFAFAF	0
⋮		
1022	0xDEAD0	0
1023	0x00002	1

PFN = 0x00001		
	PFN	Valid?
0	0x00000	1
1	0xC0CA5	0
⋮		
1022	0xCACA5	0
1023	0xCAC05	0

PFN = 0x00002		
	PFN	Valid?
0	0xF0F05	0
1	0xF0FA5	0
⋮		
1022	0xF1F0C	0
1023	0x0000C	1

Respuesta: ✓

La respuesta correcta es: fault

Pregunta **13**

Correcta

Puntúa 1,00 sobre 1,00

Para un esquema de paginación de i386 (10,10,12), con $PDBR=CR3=0x00000$ y dada la siguiente configuración de memoria, transformar **de física a virtual** la dirección **0x0000CAA**.

Expresar en hexadecimal y con todos los dígitos. Ejemplo: 0x0023ABCD.

PFN = 0x00000		
	PFN	Valid?
0	0x00001	1
1	0xFAFAF	0
⋮		
1022	0xDEAD0	0
1023	0x00002	1

PFN = 0x00001		
	PFN	Valid?
0	0x00000	1
1	0xC0CA5	0
⋮		
1022	0xCACA5	0
1023	0xCAC05	0

PFN = 0x00002		
	PFN	Valid?
0	0xF0F05	0
1	0xF0FA5	0
⋮		
1022	0xF1F0C	0
1023	0x0000C	1

Respuesta: ✓

La respuesta correcta es: FFFFFAAA

[← Entrenamiento de Virtualización](#)

[Ir a](#)

