

APELLIDOS		NOMBRE		Grupo
DNI		Firma		

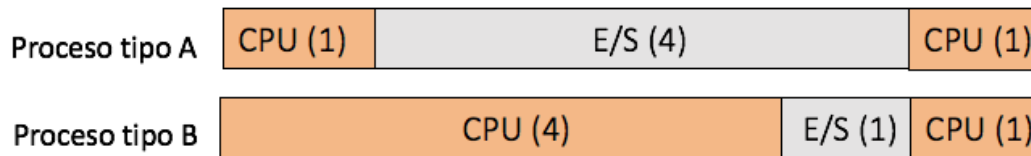
- No desgrape las hojas.
- Conteste exclusivamente en el espacio reservado para ello.
- Utilice letra clara y legible. Responda de forma breve y precisa.
- El examen consta de 7 cuestiones, cuya valoración se indica en cada una de ellas.

**1.** De los siguientes datos almacenados en la PCB de un proceso, indique si pueden cambiar o no desde que empieza el proceso hasta que acaba, justificando la respuesta.

(1,2 puntos=0,3+0,3+0,3+0,3)

1a	<b>PID</b> <b>No cambia nunca.</b> El PID es una atributo que identifica al proceso, el sistema asigna un PID a cada proceso al crearlo y se mantiene invariante durante toda la vida del proceso.
1b	<b>PPID</b> <b>PPID es el identificador de su proceso padre y puede cambiar si su proceso padre termina antes.</b> En este caso el proceso es adoptado por INIT (proceso 1) y el nuevo valor de PPID será 1.
1c	<b>Copia del contador de programa y registros de la CPU</b> <b>Cambia cada vez que se produce una excepción.</b> El sistema operativo guarda copia de estos registros en el PCB. Posteriormente cuando tenga que restaurar el contexto la utilizará.
1d	<b>Estado</b> <b>Cambia.</b> Aquí se indica el estado en el que se encuentra el proceso, por tanto se modifica siempre que cambie el estado (ejecución, espera, suspendido, ...)

**2.** Sea un sistema con una única CPU y la E/S constituida por un disco duro, que permite una **conurrencia máxima de 2 procesos**. A este sistema pueden llegar procesos de dos tipos que alternan ráfagas de CPU y de E/S (DISCO) de acuerdo a la siguiente figura (entre paréntesis se indica la duración de esas ráfagas):



Responde a las siguientes cuestiones brevemente:

(1,7 puntos=0,6+0,6+0,5)

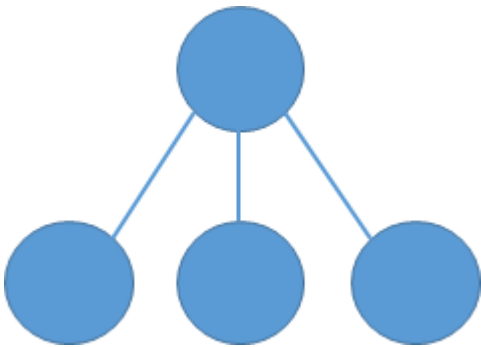
2a	<p>Describa brevemente qué es un proceso limitado por CPU o por E/S, e identifique entre los procesos anteriores de qué tipo son.</p> <p><b>Proceso limitado por CPU:</b> aquel proceso cuyas ráfagas de CPU son de mayor duración que las de E/S, y por lo tanto está limitado por la rapidez de la CPU.</p> <p><b>Proceso limitado por E/S:</b> aquel proceso cuyas ráfagas de E/S son de mayor duración que las de CPU, y por lo tanto está limitado por la rapidez de la E/S.</p> <p>Proceso tipo A: limitado por E/S</p> <p>Proceso tipo B: limitado por CPU.</p>
2b	<p>Indique qué combinación y orden de llegada de procesos permite obtener el máximo rendimiento (productividad), y la máxima utilización de la CPU. Justifíquelo gráficamente y calcule la productividad y utilización de la CPU.</p> <p>Nota: los procesos pueden ser todos de un mismo tipo.</p> <p>La combinación óptima será de 1 proceso de tipo A y otro de tipo B, entrando primero en CPU el A. De esta forma mientras el A está en E/S el B puede estar en CPU, y así se solapa el uso de la CPU y la E/S. La productividad es de 2/7 y la utilización de la CPU del 100%.</p> <pre> PA  +-----+ PB  .++++-+ CPU  ++++++ E/S  .-----.</pre> <p>+ : representa CPU, - : representa E/S, . : representa espera ó ocioso. (Desarrollo completo al final del examen)</p>
2c	<p>En el sistema utilizado, se cambia el disco duro por uno de estado sólido (SSD), reduciendo el tiempo de E/S al 25% (es decir, lo que antes costaba 4, ahora cuesta 1). Indique en este caso, qué combinación de procesos obtiene un máximo rendimiento.</p> <p>Al reducir el coste de E/S, los procesos que salen más beneficiados son los limitados por E/S. En concreto, el proceso de tipo A, que tenía una ráfaga de 4 u.t. de E/S, se reduce a 1 u.t. E/S. Por lo tanto, la mejor combinación será 2 procesos de tipo A, que consigue una productividad de 2/4 y una utilización de CPU del 100%.</p> <pre> PA1  +-+ PA2  .+-+ CPU  ++++ E/S  .--.</pre> <p>(Desarrollo completo al final del examen)</p>

### 3. Considerad los dos códigos siguientes:

Código F.c	Código E.c
<pre> 1 #include &lt;stdlib.h&gt; 2 #include &lt;stdio.h&gt; 3 int main(int argc, char *argv[]){ 4     int pid,i,status; 5     for(i=0;i&lt;3;i++){ 6         pid=fork(); 7         if (pid==0){ 8             printf("exit\n"); 9             exit(i); 10        } 11    } 12    i=0; 13    while(wait(&amp;status)&gt;0) i++; 14    printf("wait(%d)\n",i); 15 }</pre>	<pre> 1 #include &lt;unistd.h&gt; 2 int main(int argc, char *argv[]){ 3     execl("./F","F",NULL); 4     execl("./F","F",NULL); 5 }</pre>

Suponed que ambos códigos están compilados como F y E en el directorio de trabajo.

(1,6 puntos=0,4+0,4+0,4+0,4)

<b>3a</b>	<p>En total, ¿cuántos procesos crea la orden ./F? Dibuje el grafo que los relaciona.</p> <p>Cuatro procesos en total: un padre y tres hijos en abanico</p>  <pre> graph TD     A(( )) --- B(( ))     A --- C(( ))     A --- D(( ))     </pre>
<b>3b</b>	<p>Describe la salida que produce la orden ./F</p> <p>Serán las siguientes cuatro líneas:</p> <pre> exit exit exit wait(3) </pre>

<b>3c</b>	De los procesos creados con la orden ./F, ¿cuántos quedan huérfanos? Justifique la respuesta  Ninguno. Todos los procesos que crean hijos esperan a que estos acaben
<b>3d</b>	¿Cuántos procesos crea la orden ./E? Justifique la respuesta  Los mismos que en el apartado 3a). La primera llamada Exec cambia el ejecutable E por el ejecutable F

**4.** Queremos analizar diferentes planificadores a corto plazo para el sistema operativo de un computador. Los planificadores elegidos han sido SJF (Shortest-Job-First), SRTF (Shortest-Remaining-Time-First), y RR (Round-Robin) con un *quantum* de 1 u.t. ( $q=1$ ). El análisis lo queremos basar en los tiempos medios de retorno y de espera de los siguiente procesos:

Proceso	Instante de llegada	Ráfagas de CPU y E/S
A	0	4CPU + 1E/S + 1CPU
B	1	2CPU + 2E/S + 1CPU
C	4	1CPU

Rellene los diagramas siguientes para cada uno de los planificadores y calcule los valores medios del tiempo de retorno y espera de los procesos, así como la utilización de la CPU. En caso de eventos simultáneos considerar la siguiente ordenación: 1) llegada de proceso nuevo, 2) terminación de proceso, 3) abandono estado de suspensión y 4) fin del quantum. ¿Cuál sería el mejor planificador considerando el tiempo medio de retorno? ¿Y cuál considerando el tiempo medio de espera? ¿Y si consideramos la utilización de la CPU?

(1,7 puntos=0,5+ 0,5+0,5+0,2)

**4a) SJF (Shortest-Job-First)**

T	Preparados	CPU	Cola E/S	E/S	Evento
0	(A)	A			Llega A
1	B	A			Llega B
2	B	A			
3	B	A			
4	B, (C)	C	(A)	A	Llega C
5	B, (A)	A			Fin C
6	(B)	B			Fin A
7		B			
8			(B)	B	
9				B	
10		B			
11					Fin B
12					
13					

	A	B	C	Tiempo Medio
Tiempo Retorno	6	10	1	$17/3 = 5,67$ u.t
Tiempo Espera	0	5	0	$5/3 = 1,67$ u.t.
Utilización de CPU	$9/11 = 81.8\%$			

**4b) SRTF (Shortest-Remaining-Time-First)**

T	Preparados	CPU	Cola E/S	E/S	Evento
0	(A)	A			Llega A
1	A, (B)	B			Llega B
2	A	B			
3	(A)	A	(B)	B	
4	A, (C)	C		B	Llega C
5	A, (B)	B			Fin C
6	(A)	A			Fin B
7		A			
8			(A)	A	
9	(A)	A			
10					Fin A
11					
12					
13					

	A	B	C	Tiempo Medio
Tiempo Retorno	10	5	1	$16/3 = 5,33$ u.t.
Tiempo Espera	4	0	0	$4/3 = 1,33$ u.t.
Utilización de CPU	$9/10 = 90\%$			

**4c) RR (Round-Robin) q=1 u.t.**

T	Preparados	CPU	Cola E/S	E/S	Evento
0	(A)	A			Llega A
1	A, (B)	B			Llega B
2	B, (A)	A			
3	A, (B)	B			
4	C, (A)	A	(B)	B	Llega C
5	A, (C)	C		B	
6	B, (A)	A			Fin C
7	(B)	B	(A)	A	
8	(A)	A			Fin B
9					Fin A
10					
11					
12					
13					

	A	B	C	Tiempo Medio
Tiempo Retorno	9	7	2	$18/3 = 6 \text{ u.t.}$
Tiempo Espera	3	2	1	$6/3 = 2 \text{ u.t.}$
Utilización de CPU	100%			

#### 4d) Selección de planificadores.

Mejor planificador desde el punto de vista de ...	Planificador	Tiempo Medio / Utilización
... tiempo de retorno:	SRTF	$16/3 = 5,33 \text{ u.t.}$
... tiempo de espera:	SRTF	$4/3 = 1,33 \text{ u.t.}$
... utilización de la CPU	RR	100%

**5.** El programa ThreadsAdd.c de la práctica de hilos, cuyo listado aparece a continuación, realiza la suma de una matriz bidimensional matrix.

<pre> 1 #include &lt;stdio.h&gt; 2 #include &lt;pthread.h&gt; 3 4 #define DIMROW 1000000 5 #define NUMROWS 20 6 7 typedef struct row{ 8     int vector[DIMROW]; 9     long addition; 10 } row; 11 12 struct row matrix[NUMROWS]; 13 long total_addition=0; 14 15 void *AddRow( void *ptr ) 16 { 17     int k; 18     row *fi; 19     fi = (row *)ptr; 20 21     fi-&gt;addition=0; 22     for(k=0;k&lt;DIMROW;k++) 23         fi-&gt;addition += 24             exp((k*(fi-&gt;vector[k])+(k+1)* 25                 (fi-&gt;vector[k]))/ 26                 (fi-&gt;vector[k]+2*k))/2; 27     //APDO.D 28 } </pre>	<pre> 26 int main() 27 { 28     int i,j; 29     pthread_t  threads[NUMROWS]; 30     pthread_attr_t atrib; 31 32     // Vector elements are initialized to 1 33     for(i=0;i&lt;NUMROWS;i++) 34         for(j=0;j&lt;DIMROW;j++) 35             matrix[i].vector[j]=1; 36 37     // Thread attributes initialization 38     pthread_attr_init(&amp;atrib ); 39 40     for(i=0;i&lt;NUMROWS;i++){ 41         pthread_create(&amp;threads[i],&amp;atrib, 42             AddRow, (void *)&amp;matrix[i]); 43     } 44 45     for(i=0;i&lt;NUMROWS;i++) 46         pthread_join(threads[i],NULL); 47 48     for(i=0;i&lt;NUMROWS;i++) 49         total_addition += matrix[i].addition; 50     printf("Total addition is: %ld \n", 51         total_addition); 52     pthread_exit(0); 53 } </pre>
--	---

Conteste, justificando sus respuestas, a las siguientes cuestiones acerca del programa ThreadsAdd.c. Todos los supuestos se refieren a variaciones independientes sobre el mismo código original del listado anterior y suponiendo en todos los casos que la ejecución se realiza con un procesador multinúcleo.

(1,2 puntos=0,3+0,3+0,3+0,3)

<b>5a</b>	<p>¿Cuál sería el efecto sobre el resultado final obtenido si se eliminaran las líneas 45 y 46?</p> <p>El resultado podría no ser correcto porque se calcula e imprime la suma total sin esperar a que todos los hilos finalicen la obtención de la suma parcial de cada fila.</p>
<b>5b</b>	<p>Cuál sería el efecto sobre el resultado final obtenido y el tiempo de ejecución del programa si se eliminaran las líneas 45, 46 y 51.</p> <p>Todos los hilos acabarían rápidamente porque el proceso main finalizaría sin esperar la finalización del resto de hilos. El resultado sería incorrecto. El tiempo de ejecución muy breve.</p>
<b>5c</b>	<p>¿Cuál sería el efecto sobre el resultado final obtenido y el tiempo total de ejecución si se eliminaran las líneas 45 y 46 y se incluyera <code>pthread_join(threads[i], NULL)</code> en la línea 42?</p> <p>Los hilos encargados de sumar cada fila no se ejecutarían en paralelo, sino que se esperaría la finalización de cada uno antes de lanzar el siguiente. El resultado no se vería afectado pero el tiempo de ejecución sería mayor, equivalente a una ejecución secuencia</p>
<b>5d</b>	<p>¿Qué implicaciones tendría sobre la corrección del resultado si se eliminaran las líneas 48 y 49 y se añadiera <code>total_addition += fi-&gt;addition;</code> en la línea 24 ?</p> <p>El resultado podría ser incorrecto. Aparecería una condición de carrera, porque los hilos accederían a la variable compartida <code>total_addition</code> sin ningún protocolo de acceso a sección crítica.</p>

**6.** Dadas las funciones agrega y resta vistas en prácticas, contesta de forma justificada a las siguientes cuestiones

**NOTA:** Asuma que previa la declaración de las funciones se definen las constantes y variables siguientes:

```
#define REPEAT 20000000
long int V = 100;
int llave = 0;
```

1	void *agrega (void *argumento) {	15	void *resta (void *argumento) {
2	long int cont;	16	long int cont;
3	long int aux;	17	long int aux;
4		18	
5	for (cont=0; cont<REPEAT; cont++) {	19	for (cont=0; cont<REPEAT; cont++) {
6		20	
7	V = V + 1;	21	V = V - 1;
8		22	
9	}	23	}
10		24	
11	printf("--> Fin AGREGA (V=%ld) \n", V);	25	printf("--> Fin RESTA (V=%ld) \n", V);
12	pthread_exit(0);	26	pthread_exit(0);
13	}	27	}
14		28	
29	int main (void) {		
30	pthread_t hiloSuma, hiloResta,		
31	pthread_attr_t attr;		
32			
33	pthread_attr_init(&attr);		
34	pthread_create(&hiloSuma, &attr, agrega, NULL);		
35	pthread_create(&hiloResta, &attr, resta, NULL);		
36			
37	pthread_join(hiloSuma, NULL);		
38	pthread_join(hiloResta, NULL);		
39			
40	fprintf(stderr, "-----> VALOR FINAL: V = %ld\n\n", V);		
41	exit(0);		
42	}		

(1,6 puntos=0,4+0,4+0,4+0,4)

6a	<p>¿Qué es una sección crítica? Identifique, si las hay, <b>las líneas del código (indique los números)</b> que representan las secciones críticas e indique en qué <b>líneas del código</b> se debería introducir el código del protocolo de entrada (<b>indique los números</b>) y del protocolo de salida (<b>indique los números</b>) para que la sección crítica esté protegida.</p> <p>La sección crítica es aquella parte del código de los hilos donde se accede a variables compartidas En el código las secciones críticas están en las líneas 7 y 21</p> <p><b>PROTOCOLO DE ENTRADA</b></p> <p>líneas 6 y 20</p> <p><b>PROTOCOLO DE SALIDA</b></p> <p>líneas 8 y 22</p>
----	--



6b	<p>Si queremos proteger las sección críticas utilizando la solución hardware (Test&amp;Set), escriba el código que implemente el protocolo de entrada y el protocolo de salida.</p> <p><b>PROTOCOLO DE ENTRADA</b></p> <pre>while(test_and_set(&amp;llave));</pre> <p><b>PROTOCOLO DE SALIDA</b></p> <pre>llave=0;</pre>
6c	<p>Justifique si la solución hardware que utiliza Test&amp;Set, como la del código anterior, cumple las tres condiciones de los protocolos de acceso a las secciones críticas.</p> <p>Las soluciones básicas HW basadas en Test&amp;Set cumplen las condiciones de Exclusión Mutua y de progreso, ya que al ser una operación atómica sólo se deja que un proceso acceda a la variable llave para cambiar su valor, por tanto sólo hay un proceso accediendo a la sección crítica y si no hay nadie en la sección crítica la variable llave tiene un valor igual a cero, y por tanto en cuanto un hilo intente acceder a la sección crítica, lo podrá hacer.</p> <p>Por contra, la condición de espera limitada puede no cumplirse, debido a que la decisión sobre qué proceso accede a la sección crítica no es del protocolo, sino del planificador y si éste es no equitativo, puede no cumplirse</p>
6d	<p>Indique la relación entre solución básica basada en Test&amp;Set y la espera activa. ¿En qué condiciones es adecuado utilizar espera activa en los protocolos de acceso a la sección crítica?</p> <p>Las soluciones básicas HW basadas en Test&amp;Set utilizan la espera activa, es decir bucles vacíos, para comprobar el cambio de la variable llave, y así poder acceder a la sección crítica.</p> <p>No es muy recomendable el uso de la espera activa en los protocolos de acceso a la sección crítica, el motivo fundamental es la dependencia que existe con la política de planificación, ya que en el mejor de los casos (utilizando una política R-R) malgastamos tiempo de CPU preguntando por el valor de una variable que puede no cambiar su valor en todo el quantum. Pero en el peor de los casos podemos llevar al sistema a una situación de deadlock (interbloqueo), ya que si el hilo que accede al bucle vacío de la espera activa, es más prioritario que el que está ejecutando la sección crítica, nunca le dará paso para que acabe su acceso y le permita entrar. Si no se puede dar el caso de interbloqueo por prioridades, puede resultar conveniente utilizar espera activa cuando la sección crítica sea muy breve y además haya pocos hilos que la comparten, dando lugar a una baja probabilidad de que haya un consumo ocioso de CPU</p>

7. Sean tres semáforos: A, B y C inicializados del siguiente modo: A=0, B=1, C=0. Dados 3 procesos que se ejecutan concurrentemente y acceden a dichos semáforos de la siguiente forma:

Proceso 1	Proceso 2	Proceso 3
. . P(A); . . . V(C);	. . P(B); . . . V(A); V(B);	. . P(B); P(C); . . V(B);

(1 punto=0,5+0,5)

7a	<p>Indique una secuencia de operaciones P y V de los tres procesos que les permita terminar su ejecución.</p> <table border="1"> <thead> <tr> <th>PROCESO</th><th>OPERACIÓN DE SEMÁFORO</th></tr> </thead> <tbody> <tr><td>Proceso 1</td><td>P (A)</td></tr> <tr><td>Proceso 2</td><td>P (B)</td></tr> <tr><td>Proceso 2</td><td>V (A)</td></tr> <tr><td>Proceso 2</td><td>V (B)</td></tr> <tr><td>Proceso 1</td><td>V (C)</td></tr> <tr><td>Proceso 3</td><td>P (B)</td></tr> <tr><td>Proceso 3</td><td>P (C)</td></tr> <tr><td>Proceso 3</td><td>V (B)</td></tr> </tbody> </table>	PROCESO	OPERACIÓN DE SEMÁFORO	Proceso 1	P (A)	Proceso 2	P (B)	Proceso 2	V (A)	Proceso 2	V (B)	Proceso 1	V (C)	Proceso 3	P (B)	Proceso 3	P (C)	Proceso 3	V (B)
PROCESO	OPERACIÓN DE SEMÁFORO																		
Proceso 1	P (A)																		
Proceso 2	P (B)																		
Proceso 2	V (A)																		
Proceso 2	V (B)																		
Proceso 1	V (C)																		
Proceso 3	P (B)																		
Proceso 3	P (C)																		
Proceso 3	V (B)																		
7b	<p>Indique una secuencia de operaciones P y V de los tres procesos que les conduzca a una situación de interbloqueo.</p> <table border="1"> <thead> <tr> <th>PROCESO</th><th>OPERACIÓN DE SEMÁFORO</th></tr> </thead> <tbody> <tr><td>Proceso 3</td><td>P (B)</td></tr> <tr><td>Proceso 3</td><td>P (C)</td></tr> <tr><td>Proceso 2</td><td>P (B)</td></tr> <tr><td>Proceso 1</td><td>P (A)</td></tr> </tbody> </table>	PROCESO	OPERACIÓN DE SEMÁFORO	Proceso 3	P (B)	Proceso 3	P (C)	Proceso 2	P (B)	Proceso 1	P (A)								
PROCESO	OPERACIÓN DE SEMÁFORO																		
Proceso 3	P (B)																		
Proceso 3	P (C)																		
Proceso 2	P (B)																		
Proceso 1	P (A)																		

## Desarrollo pregunta 2:

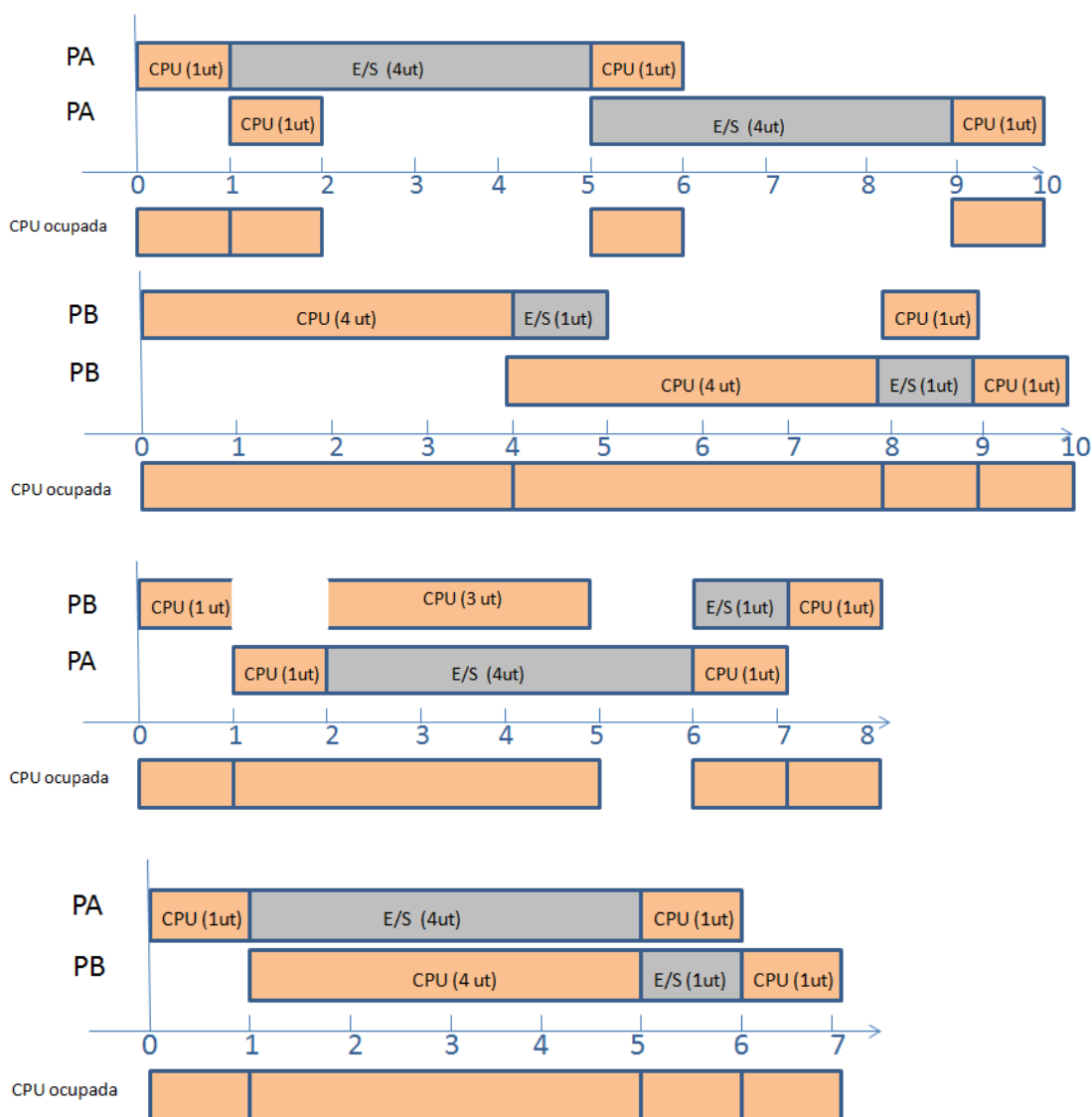
Indique qué combinación y orden de llegada de procesos permite obtener el máximo rendimiento (productividad), y la máxima utilización de la CPU. Justifíquelo gráficamente y calcule la productividad y utilización de la CPU.

Nota: los procesos pueden ser todos de un mismo tipo.

La productividad se define como el número de procesos finalizados por unidad de tiempo, mientras que utilización de CPU viene dada por las unidades de tiempo que la CPU está ocupada dividido por el total de unidades de tiempo.

Dado que sólo se dispone de una CPU y un dispositivo de E/S, no se pueden ejecutar concurrentemente ráfagas de CPU ni de E/S.

Las posibles combinaciones son dos procesos PA, dos procesos PB o un proceso PA y un PB. De forma gráfica las opciones son:



La combinación óptima es un proceso PA y otro PB, asignando primero la CPU a PA. De esta forma mientras PA está en E/S, PB puede estar en CPU, y así se solapa el uso de la CPU y la E/S. La productividad en este caso es  $2/7$  y la utilización de la CPU del 100%.

En el sistema utilizado, se cambia el disco duro por uno de estado sólido (SSD), reduciendo el tiempo de E/S al 25% (es decir, lo que antes costaba 4, ahora cuesta 1). Indique en este caso, qué combinación de procesos obtiene un máximo rendimiento.

Al reducir el coste de E/S, los procesos que salen más beneficiados son los limitados por E/S. En concreto, el proceso de tipo A, que tenía una ráfaga de 4 u.t. de E/S, se reduce a 1 u.t. de E/S. Por lo tanto, la mejor combinación será 2 procesos de tipo A, que consigue una productividad de 2/4 y una utilización de CPU del 100%.

