El examen está compuesto de dos bloques de 12 cuestiones
Para aprobar o compensar con el segundo parcial será necesario tener un mínimo de 4 cuestiones correctas en cada bloque
Las respuestas con valores no enteros deben tener 3 decimales
Todas las cuestiones tienen la misma puntuación ($10/24 = 0{,}4167$)

Bloque I

❑ Se tiene un procesador segmentado de 5 etapas con un CPI ideal de 1 y hueco de retardo de salto de 3 ciclos. Además, la dirección efectiva de salto no se conoce hasta el final del segundo ciclo del hueco. ¿Cual será el CPI real al ejecutar un programa con un 25 % de instrucciones de control, de las cuales un 60 % dan lugar a salto efectivo, si se sigue la estrategia de predicción de ...

A) ... salto NO efectivo?
B) ... salto efectivo?

❑ Tenemos los siguientes diseños alternativos para un procesador segmentado:

A) cauce de 6 etapas, 5 de duración t y otra de duración 3t
B) como el anterior, pero segmentando totalmente la etapa que hace de cuello de botella
C) como el primero, pero añadiendo 2 replicas de la etapa que hace de cuello de botella, de modo que las 3 trabajen en paralelo

Completar la siguiente tabla con las respectivas ganancias ideales respecto a un procesador como el A, pero NO segmentado:

G(A)	
G(B)	
G(C)	

❑ Completa la siguiente tabla con Xs en función de las características de los diferentes niveles RAID indicados:

Característica	0	1	3	5
disco de paridad				
paridad distribuida				
mirroring				
striping				

❑ Completa la siguiente tabla con las palabras MAT o VEC en función del tipo de arquitectura (matricial o vectorial) a la que mejor corresponden las características indicadas.

Basada en segmentación	
Rendimiento = f (tamaño de vectores o matrices)	
Basada en paralelismo	
Coste moderado	

❑ Siendo T1P el tiempo empleado en ejecutar un programa sobre un procesador, TNP el tiempo empleado en ejecutar el mismo programa sobre NP procesadores, G la ganancia proporcionada por el multiprocesador y E la eficiencia del mismo, rellenar la siguiente tabla con las expresiones que correspondan.

G(real)		G(ideal)	
E(real)		E(ideal)	

❑ Completa la siguiente tabla con Xs en función de si las características de fácil programación (FP) o elevada escalabilidad (EE) corresponden a cada uno de los tipos de computadores.

	FP	**EE**
Cluster de PCs		
Multiprocesador simétrico (SMP)		
Constelación		
Computador masivamente paralelo (MPP)		

❑ En Windlx se simula la ejecución de un bucle de 100 iteraciones sin y con adelantamiento, obteniendo una ganancia al aplicar la tecnica hardware de 1.2. Si la simulación sin adelantamiento consumía 2400 ciclos ¿cuántos ciclos de detección por iteración logra eliminar el adelantamiento?

❑ El alto consumo de cafés de Autobar por parte de algún profesor ha provocado que se tengan que hacer recortes en el presupuesto destinado a adquirir un nuevo servidor. Inicialmente se pensaba en una máquina con procesador MIPS R3000 (como la vista en clase) pero se ha tenido que optar por una versión similar, algo más barata, que presenta las siguientes deficiencias:

- No presenta cachés independientes para datos y código
- No implementa la técnica de adelantamiento

Se sabe que un programa que contiene el siguiente fragmento de código tarda en ejecutarse 150 ciclos de reloj.

```
1       addi  r30,  r0,   10
2 bucle:
3       sub   r2,   r3,   r4
4       and   r5,   r6,   r7
5       and   r8,   r9,   r10
6       or    r11,  r12,  r13
7       subui r30,  r30,  1
8       bnez  r30,  bucle
```

Nota: El registro r0 siempre contiene la constante 0.

— Si la instrucción 3 se sustituyese por **lw r2, (r20)** ¿cuántos ciclos tardaría en ejecutarse el citado fragmento?

— Si sustituimos el segundo operando de la instrucción **or** por **r2**, aparece una dependencia de datos entre dicha instrucción y la primera del bucle. ¿Cuántos ciclos se tardaría ahora en ejecutar el programa?

❑ En las gráficas del anexo se simula el comportamiento de un programa que rellena un array con los 12 primeros elementos de la secuencia de Fibonacci. Los 2 primeros son inicializados al declarar el array y un bucle calcula los 10 restantes. En función del comportamiento observado, responder a las tres cuestiones que siguen:

— Indica el nombre de la técnica o técnicas implícitas en dos de las simulaciones que justifican las mejora de sus tiempos de ejecución respecto a la tercera.

	Técnica 1	Técnica 2
Simulación A		
Simulación B		
Simulación C		

— ¿Cuántos ciclos se ahorran en cada iteración del bucle debido exclusivamente a la técnica hardware?

A

— Considerando que las gráficas representan la simulación de la última iteración ejecutada y que el último ciclo de la instrucción de salto es el ciclo final del programa ¿qué ganancia se consigue debido exclusivamente a la técnica software?

Bloque II

❑ El Servicio Nacional de Metereología ha implantado un acceso a sus previsiones metereológicas utilizando las RPC de Sun Microsystem. La API consiste en dos procedimientos, uno de los cuales es de uso libre por los usuarios y el otro es de acceso restringido a los administradores.

El procedimiento `ObtenPrev`, de acceso libre, recibe como parámetro el número de días para los que se desea la previsión (0 = hoy, 1 = hoy y mañana, etc) y devuelve una lista encadenada de estructuras, cada una de las cuales contiene la previsión para cada uno de los días. Internamente, `ObtenPrev` llama al procedimiento `PrevisionDia` que recibe como parámetro el día para el que se desea la previsión (0 = hoy, 1= mañana, 2=pasado mañana, etc) y devuelve un puntero a una estructura `infomet` con la previsión metereológica para ese día. La propia función reserva el espacio necesario para esa estructura.

El procedimiento `ActualizarPrev` es un procemiento administrativo restringido que únicamente se puede ejecutar por clientes desde una determinada máquina. El procedimiento actualiza las bases de datos con la previsión metereológica para un día determinado, para ello recibe como parámetro una pareja `<día, información metereológica>` y retorna cero si todo ha ido bien. Internamente, `ActualizarPrev` invoca el precedimiento `actualiza` que tiene el siguiente interface:

```
int actualiza( int dia, char *prevision,
               int precip, int tmax, int tmin)
```

dónde `dia` es el número del día a actualizar, `prevision` es una cadena de texto con la previsión del tiempo, `precip` es la probabilidad de lluvia y `tmax` y `tmin` son las temperaturas máxima y mínima previstas. Así mismo, en el caso de que hubiera previsión de nieve, tendría también que invocar el procedimiento `cotadenieve` que tiene el interface:

```
int cotadenieve( int dia, int cota)
```

dónde `dia` es el número del día a actualizar y `cota` la altura mínima a la que se prevé nieve. Tanto `actualiza` como `cotadenieve` retornan un cero si todo ha salido bien.

El listado 1 muestra el fichero de definición del interface necesario para acceder a los servicios y el listado 2 la implementación de uno de los servicios de la aplicación.

```
1  union CotaNieve switch (int hay) {
2    case 0:
3      void;
4    case 1:
5      int cota;
6  };
7
8  struct infomet {
9    string prev<20>;
10   int ProbPrec;
11   CotaNieve nieve;
12   int TMax;
13   int TMin;
14 };
15
16 struct parInfoDia {
17   int dia;
18   infomet info;
19 };
20
21 struct listPrev {
22   infomet prevision;
23   listPrev *sigdia;
24 };
25
26 program PrevMeteo {
27   version MeteoVER {
28     listPrev ObtenPrev( int ndias) = 1;
29     int ActualizarPrev ( parInfoDia info) = 2;
30   } =    ;
31 }= 0x20501501;
32
33
```

Listado 1. `Interface.x`

```
1  int * actualizarprev_4 (parInfoDia *dat,
2                          struct svc_req *r)
3  {
4     static int result;
5     authunix_parms *ucred;
6
7     result=1;
8     if ( r->rq_cred.oa_flavor == AUTH_UNIX ) {
9       ucred = ( struct authunix_parms *)r->rq_clntcred;
10    if (                                ,MIMAQ)) {
11     result= actualiza(
12                       
13                                        );
14       if (dat->info.nieve.hay ==1 )
15          result= cotadenieve(
16                                                  );
17       }
18    }
19    return &result;
20 }
21
22
```

Listado 2.

— ¿Qué falta en la línea 10 del listado 2?

— Completa las llamadas a `actualiza` y `cotadenieve` de las líneas 11 y 15 del listado 2.

— Escribe la implementación del servicio `ObtenPrev` utilizando el interface simplificado de las RPC de Sun.

— ¿Qué falta en el hueco de la línea 30 del listado 1?.

— El comando `ls` nos ha generado la siguiente salida en la máquina `sirio.edv.uniovi.es` (la misma que hemos usado en las prácticas durante el curso) y que sabemos que tiene un sistema operativo SunOS. En esa máquina se pretende generar los archivos ejecutables del cliente y del servidor de la aplicación, `cliente.exe` y `servidor.exe` respectivamente. ¿Cuáles son las instrucciones de compilación necesarias para generar en esa máquina esos ficheros correctamente?

```
client.c    inter.h     inter.x  inter_clnt.c
inter_svc.c inter_xdr.c server.c
```

— Antes de la invocación del servicio `ActualizarPrev` (utilizando el interface que facilita `rpcgen`) ¿Qué función/es del API de las RPC de Sun Microsystem ha tenido necesariamente que ejecutar el cliente? Escribe el código de la llamada a esa/s función/es declarando todas las variables que utilices.

❏ En la siguiente tabla se muestran las acciones que pueden realizar tanto el cliente como el servidor respecto a la tolerancia a fallos de una RPC. Cada una de esas acciones dará lugar a una semántica concreta. Rellena los huecos de la tabla de manera adecuada.

Cliente	**Servidor**		
Reintenta RPC	**Filtra duplicados**	**Acción**	**Semántica RPC**
No	No aplicable		
		Reejecución Servicio	
	SI		

❏ Se desea crear un par de claves RSA para poder realizar criptografía de clave pública. Para ello se han escogido como generadores los números primos $P = 79$ y $Q = 47$. Durante el cálculo de la pareja de claves, para la clave pública se escogió el número más pequeño de los posibles, y para la secreta la que resulta del algoritmo RSA.

— ¿Cuál es la pareja de claves $\langle K_P, N\rangle$ y $\langle K_S, N\rangle$ calculada?

— Se quieren utilizar las claves calculadas para firmar un mensaje. Si el mensaje está compuesto por los siguientes bytes (en hexadecimal):

`23 7F A0 DD E6`

¿Cuál ha sido la primera fórmula matemática que se ha aplicado para realizar la firma? (se supone que se firma el mensaje directamente, no su *hash*) Contestar con todos los elementos de la fórmula en **base diez**.

❏ Como parte de un algorimo de cifrado, es necesario obtener el resultado de $53^{34} \bmod 77$. Para ello se aplica el algoritmo de la exponenciación binaria aplicando el módulo tras cada paso, para mantener los números manejables. ¿Cuál es el resultado final de la operación y cuántas multiplicaciones se han tenido que realizar?

Resultado:	Multiplicaciones:

❏ El siguiente código pretende poner en hora el reloj de un computador que forma parte de un sistema distribuído. Para ello hace uso de las siguientes funciones:

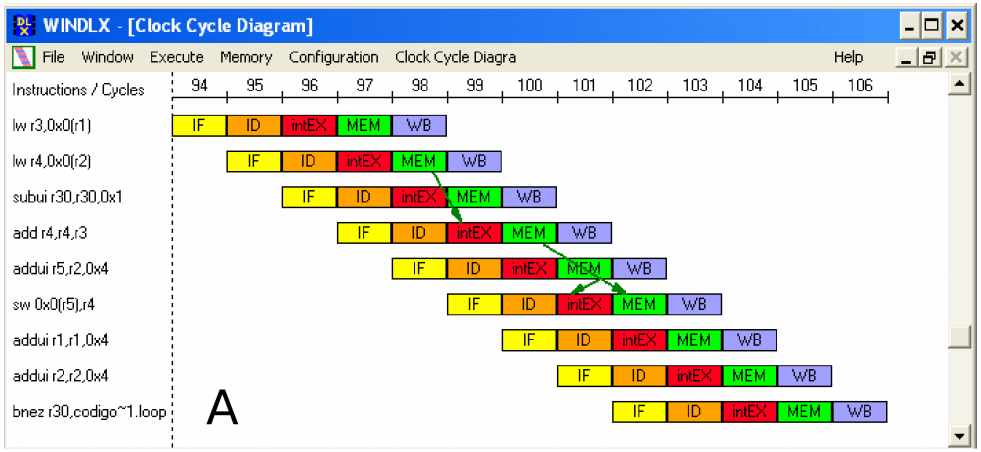
- `time(NULL)` Esta función devuelve el contador de segundos de la máquina local, con origen de tiempos en el 1 de Enero de 1970.
- `servicio_time(maquina)` Esta función le pide la hora a la máquina que recibe como parámetro, usando el protocolo `time`. La función devuelve el valor respondido por la máquina (ya con el orden de bytes correcto), que tiene como origen de tiempos el 1 de Enero de 1900.
- `poner_hora(contador)` Cambia la hora local de la máquina, de forma que la nueva hora sea la especificada en `contador`, que debe tener origen de tiempos en 1970. En esta implementación no nos preocupamos por evitar cambios bruscos o retrocesos en dicho reloj local.

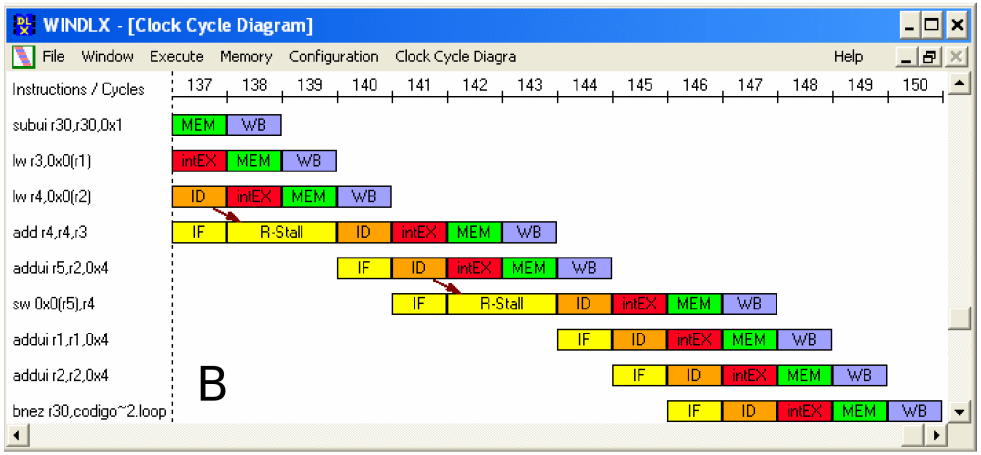
```
1 #define OFFSET 2208988800LU
2 long int hora1, hora2, lahora;
3 hora1  = time(NULL);
4 lahora = servicio_time("hora.uniovi.es");
5 hora2  = time(NULL);
6 poner_hora(                                  );
```

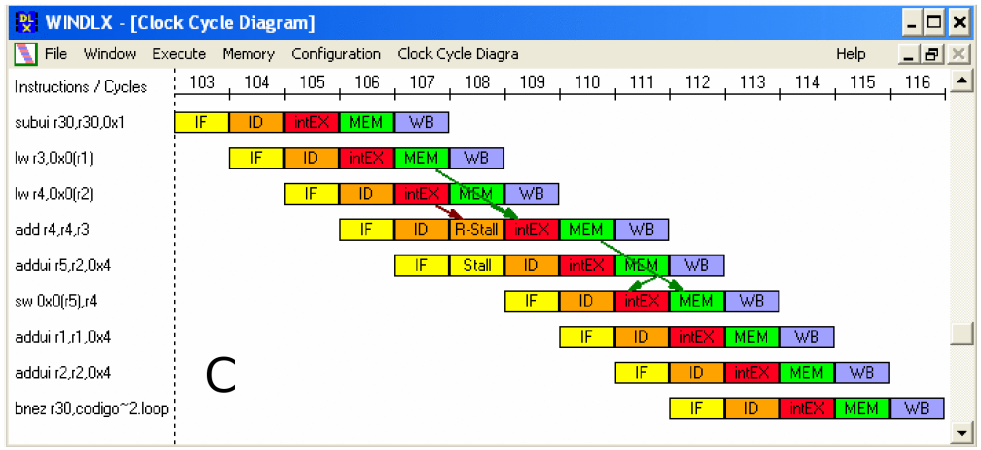
— Para estimar la hora correcta a partir de la respuesta del servidor, utiliza el algoritmo de Christian. ¿Qué falta entonces en el hueco?

— Si en lugar de una sincronización de relojes físicos se deseara simplemente una sincronización de relojes lógicos con el algoritmo de Lamport ¿que habría que poner en vez de la última línea del programa?

Anexo


WINDLX - [Clock Cycle Diagram]
File Window Execute Memory Configuration Clock Cycle Diagra Help
Instructions / Cycles
94 95 96 97 98 99 100 101 102 103 104 105 106
lw r3,0x0(r1)
lw r4,0x0(r2)
subui r30,r30,0x1
add r4,r4,r3
addui r5,r2,0x4
sw 0x0(r5),r4
addui r1,r1,0x4
addui r2,r2,0x4
bnez r30,codigo~1.loop
IF ID intEX MEM WB
A


WINDLX - [Clock Cycle Diagram]
File Window Execute Memory Configuration Clock Cycle Diagra Help
Instructions / Cycles
137 138 139 140 141 142 143 144 145 146 147 148 149 150
subui r30,r30,0x1
lw r3,0x0(r1)
lw r4,0x0(r2)
add r4,r4,r3
addui r5,r2,0x4
sw 0x0(r5),r4
addui r1,r1,0x4
addui r2,r2,0x4
bnez r30,codigo~2.loop
IF ID intEX MEM WB R-Stall
B


WINDLX - [Clock Cycle Diagra]
File Window Execute Memory Configuration Clock Cycle Diagra Help
Instructions / Cycles
103 104 105 106 107 108 109 110 111 112 113 114 115 116
subui r30,r30,0x1
lw r3,0x0(r1)
lw r4,0x0(r2)
add r4,r4,r3
addui r5,r2,0x4
sw 0x0(r5),r4
addui r1,r1,0x4
addui r2,r2,0x4
bnez r30,codigo~2.loop
IF ID intEX MEM WB R-Stall Stall
C