FUNDAMENTOS DE INFORMÁTICA
(Exámenes del 7 y del 17 de febrero de 2012)

---

1. Se desea adquirir un computador para realizar predicciones meteorológicas en Europa utilizando un modelo matemático que divide la atmosfera terrestre en 40 capas, y la superficie en cuadrados de 10 km de lado. Para cada punto de la retícula de paralelepípedos así formada se han de ejecutar 1.000 instrucciones máquina con datos de coma flotante, y, para obtener la evolución en el tiempo y resultado de la predicción, se han de iterar dichas instrucciones 20.000 veces. Suponer que cada paralelepípedo, por término medio, contribuye con un punto en la retícula y que la superficie de Europa a considerar es de 10.000.000 Km^2. La predicción hay que realizarla 4 veces al día. Por otra parte, el modelo tiene que calcular en cada punto cuatro parámetros (velocidad de viento, temperatura, presión y humedad) de forma que los cálculos anteriores han de ejecutarse 4 veces. Estimar:

1a) La potencia de cálculo que necesitaría un supercomputador para ejecutar satisfactoriamente el modelo.

La potencia de cálculo de un computador dedicado a cálculo científico-técnico se mide en el número de instrucciones con números reales (coma flotante) a ejecutar en un segundo FLOP/s o MFLOPS.

Obviamente, si hay que hacer 4 predicciones al día cada una de ellas deberá de hacerse como mucho en 24/4 = 6 horas.

$$t_{predicción} = \frac{24}{4} horas = \ 6 \cdot 60 \cdot 60 \ segundos = 21.600 \ s$$

Por otra parte, con los datos que nos da el problema podemos determinar el número de instrucciones a ejecutar en cada predicción, estas instrucciones se tienen que ejecutar en 6 o menos horas.

Como nos dan el número de instrucciones que hay que ejecutar en cada punto de la retícula, primero tenemos que calcular el número de puntos, que coincide con el de paralelepípedos (N_p), y esté número será:

$$N_p = nº \ de \ cuadrados \ en \ la \ base \ x \ nº \ de \ capas = \frac{10.000.000 \ Km^2}{5 \cdot 5 \ Km^2} \cdot 40 \ capas$$
$$= 16 \cdot 10^6 \ puntos$$

El número de instrucciones en cada punto para cada predicción ($NIP_{predición}$), de acuerdo con lo indicado en el enunciado, será:

$$NIP_{predición} = \ 1.000 \ x \ 40.000 \ x \ 4 = 16 \cdot 10^7 \ intrucciones/punto$$

Con lo que el nº total de instrucciones por predicción ($NI_{predicción}$) será:

$$NI_{predicción} = N_p \cdot \ NIP_{predicción} = 256 \cdot 10^{13}$$

Es decir, el número mínimo de instrucciones por segundo que tendría que ejecutar el procesador (potencia de cálculo, P) sería:

$$P = \frac{NI_{predicción}}{t_{predicción}} = \frac{256 \cdot 10^{13}\ instrucciones\ de\ coma\ flotante}{21.600\ segundos} = 1{,}18 \cdot\ 10^{11}\ FLOPS$$
$$= 118 \cdot\ 10^{9}\ FLOPS = 118\ GFLOPS$$

1b) El tiempo que tardaría en ejecutarse una predicción (de las 4 del día) con el modelo en un PC que tuviese un único procesador que, por termino medio, consumiese 2 ciclos por instrucción y con una frecuencia de reloj de 2 GHz.

Como es bien conocido el tiempo de ejecución de un programa se puede calcular como el producto del nº de instrucciones (NI) por el número medio de ciclos por instrucción (NCI) por el tiempo de duración de cada ciclo (periodo de reloj, T); es decir, en este caso:

$$t_{predicción} = NI_{predicción} \cdot NCI \cdot T = \frac{NI_{predicción} \cdot NCI}{F} = \frac{256 \cdot 10^{13} \cdot 2}{2 \cdot 10^{9}} = 1.024 \cdot 10^{4} s$$

A continuación vamos a pasar a meses, días y horas esos segundos:

$$1.024 \cdot 10^{4} s = \frac{1.024 \cdot 10^{4}}{30 \cdot 24 \cdot 60 \cdot 60}\ meses = 3{,}950617284\ meses$$
$$= 3\ meses, 28\ día, s\ 12\ horas, 26\ minutos, y\ 40\ segundos$$

1c) **Sabiendo que la superficie de Europa es un 2% de la superficie mundial; ¿qué potencia debería tener el supercomputador para hacer la predicción en toda la atmosfera terrestre?**

Si Europa es el 2% de la superficie mundial, quiere decir que esta última es 100/2 = 50 veces mayor; con lo que la potencia de cálculo ahora debería ser:

$$P_{mundial} = P_{Europa} \cdot 50 = 118 \cdot 50 = 5.900\ GigaFLOPS = 5{,}9\ TFLOPS$$

2. Dado el texto:

Caña: 8€

obtener su codificación en:

2a) UNICODE (*Nota:* recuerde que en ASCII € es 80, y ñ es F1).

	C	a	ñ	a	:	SP	8	€
ASCII →	43	61	F1	61	3ª	20	38	80
UNICODE →	0043	0061	00F1	0061	003ª	0020	0038	0080

2b) UTF-8

Tenemos que tener en cuenta las siguientes reglas de codificación:

Rango UNICODE	*UTF-8*	*Nº de Bytes*
0000 -00007F (US-ASCII)	**0xxxxxxx**	**1**
000080 - 0007FF	**110xxxxx 10xxxxxx**	**2**
000800 - 00FFFF	**1110xxxx 10xxxxxx 10xxxxxx**	**3**
010000 - 10FFFF	**11110xxx 10xxxxxx 10xxxxxx 10xxxxxx**	**4**

Lo que quiere decir, que todos los caracteres, excepto ñ y €, se codifican con un sólo byte, en cambio, tanto ñ como € hay que codificarlos con dos bytes, añadiendo al inicio de ellos los bits de relleno 110 y 10, respectivamente. Es decir:

	C	**a**	**ñ**	**a**	**:**	**SP**	**8**	**€**
UNICODE →	0043	0061	00F1	0061	003A	0020	0038	0080
			1100 0011 **10**11 0001					**110**0 0010 **10**00 0000
UTF-8 →	43	61	C3 B1	61	3A	20	38	C2 80

3c) ¿Qué factor de compresión se obtiene al codificarlo en UTF-8 en lugar de UNICODE)?

En UNICODE el nº de bits utilizado es:

32 cifras HEX · 4 bits/HEX = 128 bits

En UTF-8 el nº de bits utilizado es:

20 cifras HEX · 4 bits/HEX = 80 bits

Con lo que el factor de compresión será:

$$f_c = \frac{C_{antes}}{C_{después}} = \frac{128}{80} = 1{,}6$$

Es decir, el factor de compresión es de 1,6 a 1.

3. La sentencia de lenguaje C++:

For (i=vi; i<vf; i++) A(i)=k;

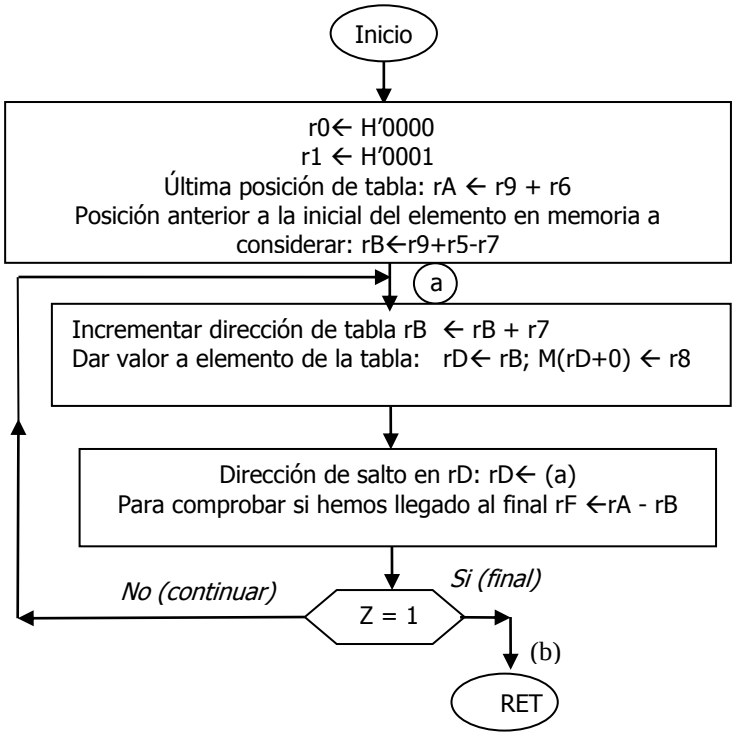
representa un bucle que da el valor k a los elementos sucesivos de una tabla desde la posición vi a la vf. Obtener el código que se generaría si esta instrucción se compilase para un procesador CODE-2. Realizar el programa suponiendo que los distintos parámetros ya están almacenados en los siguientes registros: r5: valor del índice i; r6: valor final del índice i; r7: valor del incremento del índice i; r8: valor a dar a cada elemento de la tabla, y r9: posición inicial de la tabla A en la memoria principal.

Solución:

Asignación de registros y de memoria:

00E0	Posición de memoria de inicio del programa
r0	H'0000
r1	H'0001
r5	Valor inicial del índice (i0)
r6	Valor final del índice (if)
r7	Incremento del índice
r8	Valor a dar a los elementos de la tabla
r9	Posición inicial de la tabla en la memoria
RA	Dirección final de la tabla
rB	Dirección de memoria del elemento de la tabla en consideración (puntero de la tabla)

Organigrama



Programa en nemónico y hexadecimal:

	dirección	Nemónico	Código HEX	Explicación
	00E0	LLI r0, 00	**2000**	r0 ←H'0000
	00E1	LLI r1, 01	**2101**	r1←H'0001
	00E2	ADDS rA,r9,r6	**6A96**	rA←rA+r6
	00E3	ADDS rA,rA,r5	**6AA5**	rA←rA+r5
	00E4	SUBS rB,r9,r7	**7B97**	rB←r9-r7
(a)	00E5	ADDS rB,rB,r7	**6BB7**	Posicionar el puntero en la dirección a considerar
	00E6	ADDS rD,rB,r0	**6DB0**	rD ← rB (ubicar puntero en rD)
	00E7	ST [00], r8	**1800**	Almacenar el contenido de r8 en la Tabla
	00E8	SUBS rF,rA,rB	**7FAB**	para comprobar si se ha llegado al final
	00E9	LLI rD,ED	**2DE4**	Dirección de salto al final
	00EA	BZ	**B100**	Salto si final
	00EB	LLI rD, E5	**2DE4**	Dirección de salto si no final
	00EC	BR	**B000**	Salto a (a)
(b)	00ED	HALT	**F000**	Final del programa

4. Se desea realizar un programa en lenguaje máquina de CODE-2 que, teniendo en los registros r2 y r3 dos números positivos, realice la multiplicación de ambos números, almacene el resultado en la posición de memoria contenida en r4, y lo visualice en OP1. Suponer que el programa se cargará a partir de la dirección H'0000. Efectuar la multiplicación sumando el multiplicando tantas veces como indique el multiplicador, y el programa debe prever que uno de los multiplicandos pueda ser 0, y si se produce desbordamiento debe proporcionarse por el puerto de salida el mensaje FFFF.

   La cuestiones a que se debe responder son las siguientes:

   3a) Organigrama del programa
   3b) Asignación de registros y memoria
   3c) Redactar el programa en nemónicos (código máquina).
   3d) Escribir las últimas 10 instrucciones en hexadecimal
   3e) Obtener el tiempo que tardaría en ejecutarse el programa si R2=8 y R3= 5, y si la frecuencia de reloj de CODE-2 fuese de 10 GHz

Nota: Esta forma de realizar la multiplicación es muy poco eficiente, entre otros motivos porque el tiempo de ejecución del programa va a depender del valor del multiplicador

SOLUCIÓN

a. Organigrama del programa.

La multiplicación la realizamos sumando el multiplicando tantas veces como indique el multiplicador. Al hacer cada suma parcial comprobamos si se produce desbordamiento (*V=1*). El organigrama se muestra en la figura.

b. Asignación de registros y de memoria.

- El programa se carga a partir de la dirección M(0000)
- En M(r4) se deja el resultado

R0	Valor 0	0000
R1	Valor uno	0001
R2	Multiplicando	
R3	Multiplicador	

R4	Dirección resultado	
R5	Patrón overflow	0F0F
R6	Resultado multiplicación	0000

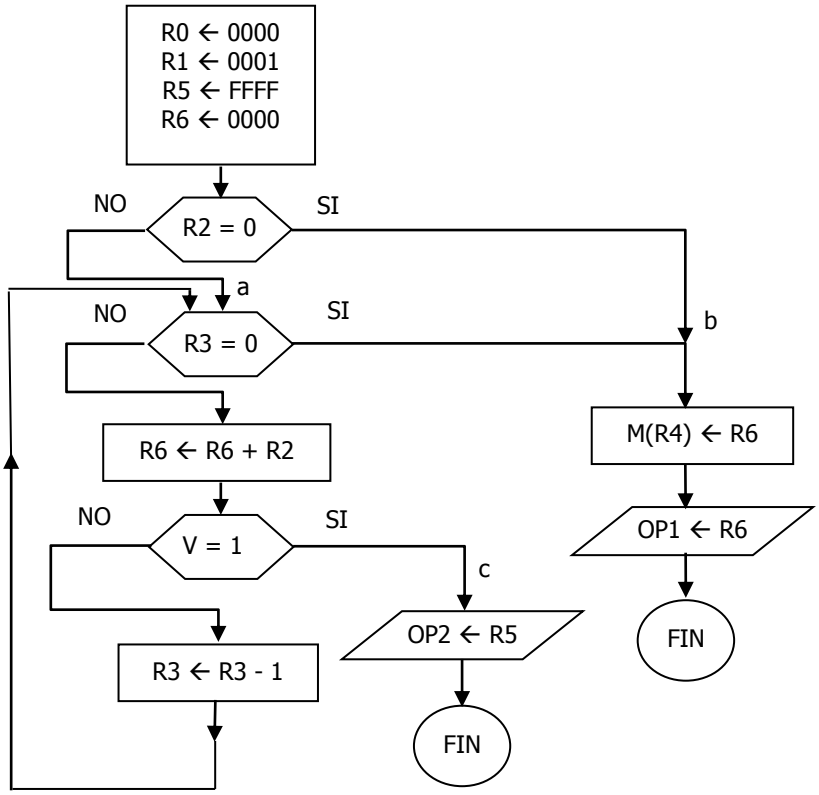


Figura Organigrama del programa de multiplicar

a) Programa en nemónicos y código máquina. Puede verse en la siguiente tabla:

Programa de multiplicar enteros positivos

Salto	Dirección	Nemónico	Hexadecimal	Comentarios
	0000	LLI R0,00	2000	*Inicializaciones*
	0001	LLI R1,01	2101	
	0002	LLI R5,FF	250F	
	0003	LHI R5,FF	350F	
	0004	LLI R6,00	2600	
	0005	SUBS RF,R2,R0	7F20	*¿R2 = 0?*
	0006	LLI RD,11	2D11	
	0007	BZ	C100	*Salto a (b)*
(a)	0008	SUBS RF,R3,R0	7F30	*¿R3 = 0?*
	0009	LLI RD,11	2D11	
	000A	BZ	C100	*Salto a (b)*
	000B	ADDS R6,R6,R2	6662	*Paso de multiplicación*
	000C	LLI RD,14	2D14	
	000D	BV	C400	*Salto a (c) si overflow*
	000E	SUBS R3,R3,R1	7331	
	000F	LLI RD,08	2D08	
	0010	BR	C000	*Salto a (a)*
(b)	0011	ST [0+R4],R6	1600	*Memorizar resultado*
	0012	OUT OP1,R6	5601	*Sacar el resultado*
	0013	HALT	F000	*Fin*

(c)	0014 0015	OUT OP2,R5 HALT	5502 F000	*Salida de overflow* *Fin*

b) Tiempo que tardaría en ejecutarse el programa si r2=5 y r3=2 si la frecuencia de reloj de CODE-2 fuese de 10 GHz

Suponiendo que R2 = 5 y R3 = 2, el programa ejecutará las siguientes instrucciones:

Ciclos	Instrucción	Nº de veces	Total ciclos
6	LLI	5	30
7	SUBS	1	7
6	LLI	1	6
6	BZ	1	6
7	SUBS	R2 = 2, 1, 0 (3 veces)	21
9	LLI	R2 = 2, 1, 0 (3 veces)	27
6	BZ	R2 = 2, 1, 0 (3 veces)	18
7	ADDS	R2 = 2, 1 (2 veces)	14
6	LLI	R2 = 2, 1 (2 veces)	12
6	BV	R2 = 2, 1 (2 veces)	12
7	SUBS	R2 = 1, 0 (2 veces)	14
6	LLI	R2 = 1, 0 (2 veces)	12
6	BR	R2 = 1, 0 (2 veces)	12
9	ST	1	9
8	OUT	1	8
6	HALT	1	6
		TOTAL	214

El periodo de reloj (tiempo de ciclo) es:

$$T = \frac{1}{F} = \frac{1}{10 \times 10^9} = 100 \mu s$$

con lo que el tiempo de ejecución del programa es:

$$t = T \cdot N = 100 \mu s \times 214 = 21 ms$$

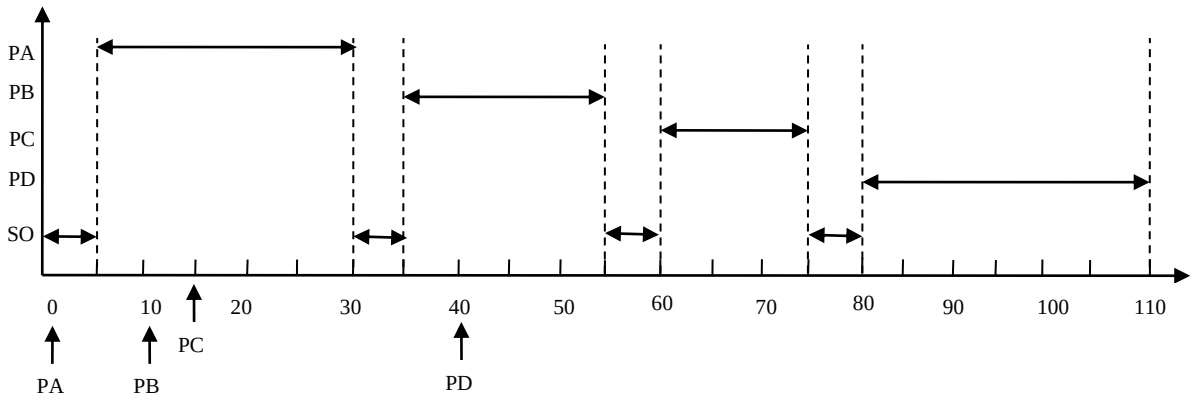
5. Suponga que en un computador se presentan cuatro procesos, cuyas prioridades, instantes de llegada y tiempos de utilización del procesador se indican en la tabla. Suponga que el sistema es no apropiativo ya que el sistema operativo sólo invierte (5 ms) para realizar la planificación cuando concluye cada proceso.

Proceso	*Prioridad (1 la mayor)*	*Instante de llegada (ms)*	*Tiempo de procesador (ms)*
PA	3	0	25
PB	4	10	20
PC	2	15	15
PD	1	40	30

Obtener el coeficiente de respuesta del proceso PB en los siguientes casos:

a) Planificación FCFS (*First Come, First Served*)

El orden de ejecución de los procesos será: PA → PB → PC → PD, siempre que hayan llegado en el momento que les llegue su turno



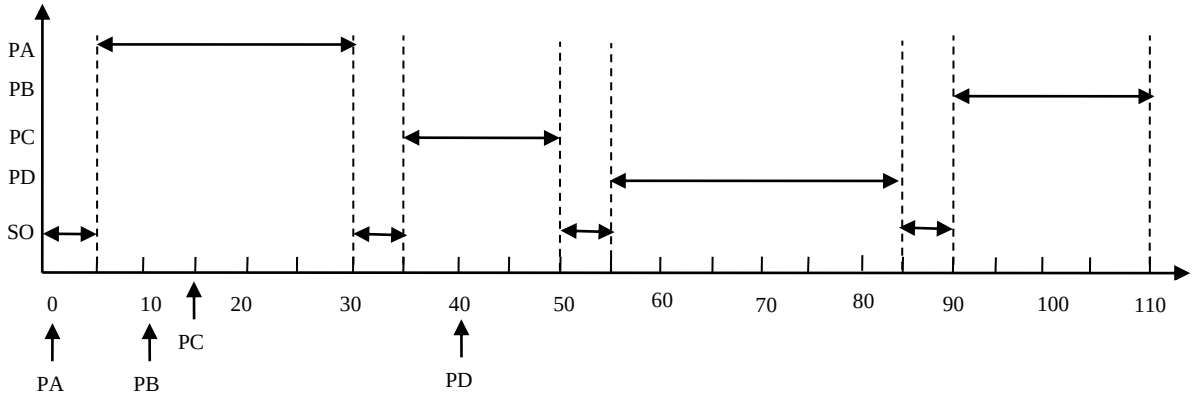
Por definición el tiempo de respuesta (t_r), es el tiempo comprendido entre el instante que llega el proceso (t_0) y el tiempo en que finaliza el mismo (t_f); y el coeficiente de respuesta (R) es el tiempo de respuesta dividido por el tiempo de procesamiento (t_{CPU}); es decir, para el proceso B se tiene:

$$R = \frac{t_{rB}}{t_{CPU,B}} = \frac{t_{fB} - t_{0B}}{t_{CPU,B}} = \frac{55 - 10}{20} = 2{,}25$$

b) Planificación por prioridades

El orden de ejecución de los procesos será:

PA (en t = 0ms sólo está él) → PC (por ser más prioritario que PB) → PD (por ser más prioritario que PB) → PB,

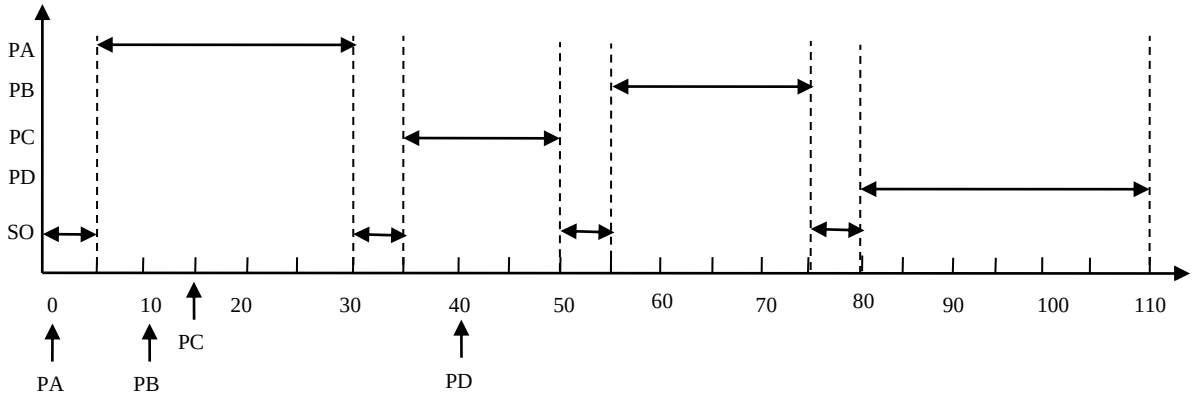


En este caso el coeficiente de respuesta del proceso PB será:

$$R = \frac{t_{rB}}{t_{CPU,B}} = \frac{t_{fB} - t_{0B}}{t_{CPU,B}} = \frac{110 - 10}{20} = 5$$

c) Planificación SPN (*Shortest Process Next*)

PA (en t = 0ms sólo está él) → PC (por durar menos que PB) → PB (por durar menos que PD) → PD



En este caso el coeficiente de respuesta del proceso PB será:

$$R = \frac{t_{rB}}{t_{CPU,B}} = \frac{t_{fB} - t_{0B}}{t_{CPU,B}} = \frac{75 - 10}{20} = 3{,}25$$