

SOLUCIONES DEL EXAMEN

C1.- Se tiene un sistema ordenador donde el tamaño de las instrucciones y de los datos es siempre el mismo. El rendimiento ideal de este sistema supone un CPI de 3 ciclos. En la tabla adjunta, se indican las instrucciones que reducen este comportamiento ideal, señalando en cada caso el porcentaje de las mismas y el retardo promedio asociado.

Tipo	LD/ST	Salto	Aritmética. CF
Porcentaje uso	20%	15%	20%
RETARDO ^o	2	1	1

Se realizan dos mejoras consecutivas:

- 1) En primer lugar, se mejora la aritmética en CF, de tal forma que ahora las instrucciones de este tipo tardan la mitad.
 - 2) A continuación, se mejora el acceso a memoria en un 30% en relación al tiempo de acceso original.
- Se pide, aplicando **necesariamente la ley de Amdahl**, la ganancia tras la primera mejora y la aceleración final tras la aplicación de ambas.

SOLUCION

Si las instrucciones y los datos tienen el mismo tamaño, gastan lo mismo en el acceso a memoria (2 ciclos).

Mejora CF:

La mejora sólo afecta los ciclos de la ruta de datos para CF, es decir 2 ciclos, se mejoran a la mitad.

$$A_m = 2$$

$$F_m = 0,2 \cdot 2 / (3 + 0,2 \cdot 2 + 0,15 \cdot 1 + 0,2 \cdot 1) = 0.106$$

$$A^1_G = 1 / \{(1 - F_m) + F_m/A_m\} = 1 / \{0.894 + 0.053\} = 1,056. \quad A^1_G = 5,6 \%$$

Mejora Memoria:

Tras la mejora anterior, las instrucciones de CF tardaran la mitad en la ruta de datos, es decir $2+1=3$ ciclos, es decir no hay retardo de CF tras la mejora.

La mejora se aplica a todas las instrucciones en su captura mas las LD/ST en la operación con datos.

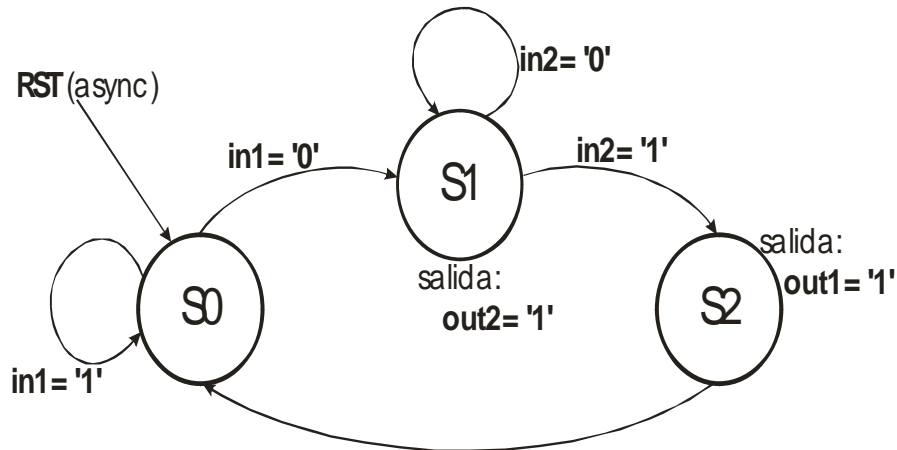
$$A_m = 1,3.$$

$$F_m = 1,2 \cdot 2 / (3 + 0,2 \cdot 2 + 0,15 \cdot 1) = 0.676$$

$$A^2_G = 1 / \{(1 - F_m) + F_m/A_m\} = 1 / \{0.324 + 0.52\} = 1,18. \quad A^2_G = 18 \%$$

$$A_G = A^2_G \times A^1_G = 1.056 \times 1.18 = 1.246 \Rightarrow A_G = 24,6\% \text{ de mejora total.}$$

C2.- Completar el código VHDL dado más abajo en los puntos marcados para que implemente la máquina de estados representada en la siguiente figura.



```

LIBRARY ieee;
USE ieee.std_logic_1164.ALL;

ENTITY maq IS PORT (
    clk, rst : IN std_logic;
    in1, in2 : IN std_logic;
    out2, out1: OUT std_logic);
END maq;

ARCHITECTURE archmoore1 OF maq IS

TYPE estado_T IS (S0, S1, S2);
SIGNAL estado: estado_T;

BEGIN
    fsm: PROCESS (clk, rst)(1)
    BEGIN
        IF rst = '1' THEN
            estado <= S0;

        ELSIF rising_edge(clk) THEN
            CASE estado IS (2)

                WHEN S0 => (3)
                    IF in1 = '0' THEN estado <= S1;
                    END IF;

                WHEN S1 => (4)
                    IF in2='1' THEN estado <= S2;
                    END IF;

                WHEN S2 => estado <= S0;

            END CASE;
        END IF;

    END PROCESS fsm;

    out2 <= '1' WHEN (estado = S1) ELSE '0'; (5)
    out1 <= '1' WHEN (estado = S2) ELSE '0'; (6)

END archmoore1;

```

C3.- Se define un sistema cache unificado de 128 bytes. La estructura de la cache es asociativa de 2 vías, con 8 bytes de bloque y una política de reemplazamiento del tipo LRU. El sistema tiene un bus de direcciones es de 20 bits y uno de datos de 32 bits y se sabe que no existen bits de control.

Se pide:

a) Los campos en los que se divide la dirección para el acceso a la cache.

b) Utilizando los esquemas adjuntos, el estado final de la cache tras la ejecución completa del programa que lee 2 palabras de memoria, las modifica y las reescribe en otra parte de la misma. Indicar expresamente la tasa de aciertos obtenida en la ejecución del programa.

DIRECCION	INSTRUCCION	DESCRIPCION
0FFE4	MOVE #0, R1	; 0 => R1
0FFE8	MOVE #2, R2	; 2 => R2
0FFEC	L1: LOAD 00F40H,R1,R4	; M{04000+(R1)} => R4
0FFF0	ADD R4, #1, R4	; R4 + 1 => R4
0FFF4	STORE 00F60H,R1, R4	; R4 => M{06000+(R1)}
0FFF8	ADD R1, #4, R1	; R1 + 4 => R1
0FFFC	SUB R2, #1, R2	; R2 - 1 => R2
10000	BZS FIN	; Salta si Z = 1
10004	JMPL L1	; Salto incondicional
1AB24	FIN: STOP	; Fin del programa

SOLUCION

Dirección	Etiqueta	Índice	B/Bq	A/F
	14 bits	3 bits	3 bits	
0FFE4	0FF 11	10 0	100	F
0FFE8	0FF 11	10 1	000	F
0FFEC	0FF 11	10 1	100	A
00F40	00F 01	00 0	000	F
0FFF0	0FF 11	11 0	000	F
0FFF4	0FF 11	11 0	100	A
00F60	00F 01	10 0	000	F
0FFF8	0FF 11	11 1	000	F
0FFFC	0FF 11	11 1	100	A
10000	100 00	00 0	000	F
10004	100 00	00 0	100	A
0FFEC	0FF 11	10 1	100	A
00F44	00F 01	00 0	100	A
0FFF0	0FF 11	11 0	000	A
0FFF4	0FF 11	11 0	100	A
00F64	00F 01	10 0	100	A
0FFF8	0FF 11	11 1	000	A
0FFFC	0FF 11	11 1	100	A
10000	100 00	00 0	000	A
1AB24	1AB 00	10 0	100	F

	VIA 2			VIA 1		
	TAG	W1	W0	TAG	W1	W0
000	100 00	M(10004)	M(10000)	00F 01	M(00F44)	M(00F40)
001						
010						
011						
100	00F 01	M(00F64)	M(00F60)	1AB 00	M(1AB24)	M(1AB20)
101				0FF 11	M(0FFEC)	M(0FFE8)
110				0FF 11	M(0FFF4)	M(0FFF0)
111				0FF 11	M(0FFFC)	M(0FFF8)

La tasa de aciertos es de $12/20 = 0,6$, es decir de $H = 60 \%$

C4.- Se tiene un sistema ordenador, cuyos buses de direcciones y datos son ambos de 32 bits. El sistema tiene dos unidades cache independientes para instrucciones y datos, ambas con 8 palabras por bloque. porcentaje de instrucciones que deben acceder a la memoria de datos es del 25%. Se dispone de un esquema de post escritura (PE), en donde la estadística de bloque modificados es del 50%. Se conoce que el tiempo de acceso para ambas unidades cache es de 10 nsec y que el tiempo de acceso para la memoria es de 50 nsec. Se sabe que el porcentaje de aciertos en la cache de instrucciones es del 96% y se quiere conocer cual debe ser el máximo porcentaje de fallos en la cache de datos para que el tiempo de acceso medio por instrucción sea a) ncomo máximo 35 nsec y b) como máximo 25 nsec.

NOTA: La cache no cuenta con bits de validez.

SOLUCION:

Como el bloque tiene 8 palabras, la penalización por bloque es $t_B = 8 \times t_m = 400$ nsec

Los accesos a la cache de datos y de instrucciones deben ser considerados de forma independiente.

$$1/t_{acc} = \%Acc^{INS} \{t_c + (1-H_I) t_B\} + \%Acc^{DAT} \{t_c + (1-H_D)(1+w_M)t_B\}$$

$$F_D = (1-H_D) = [1/t_{acc} - \%Acc^{INS} \{t_c + (1-H_I) t_B\} - \%Acc^{DAT} t_c] / \{\%Acc^{DAT} (1+w_M)t_B\}$$

$$F_D = [35 - 1 \cdot \{10 + 0,04 \cdot 400\} - 0,25 \cdot 10] / (0,25 \cdot 1,5 \cdot 400) = 0,043.$$

Es decir el porcentaje de fallos debe ser inferior al 4,3%

En el caso que $1/t_{acc} = 25$ nsec.

$$F_D = [25 - 1 \cdot \{10 + 0,04 \cdot 400\} - 0,25 \cdot 10] / (0,25 \cdot 1,5 \cdot 400) = - 0,023$$

El signo menos implica que no es posible encontrar un tiempo promedio de 25 nsec sin cambiar alguno de los otros parámetros.

C5.- Se dispone de un ordenador con un procesador cuyas direcciones virtuales son de 24 bits y que dispone de una memoria real de 1 Mbyte. Se opta por un sistema de memoria paginado, con un tamaño de página de 256 bytes y longitud del descriptor de página de 2 bytes en el que se incluye, el marco de página, un bit de presencia y otros bits de control. Como la tabla de páginas es muy grande, ésta a su vez está paginada en tres niveles y sólo se guarda una parte en memoria real (los dos últimos niveles). La unidad de manejo de memoria (MMU) del sistema dispone de una unidad TLB completamente asociativa de 8 entradas. Además del TLB, dispone de una memoria de sustitución directa, que contiene todos los descriptores del primer nivel de las páginas en las que se ha dividido la tabla de páginas. El tamaño de la ruta de datos y por tanto el tamaño de las lecturas es de 16 bits. Se pide:

- 1) El tamaño del TLB y de la memoria de sustitución directa presente en la MMU, así como la longitud y campos de los descriptores de página en ella guardados.
- 2) Indicar los pasos que debe seguir la MMU cuando el procesador solicite las siguientes lecturas: EF2C58h y CC25B4h y que resultado se devolverá al procesador.
- 3) ¿Cuántas excepciones por falta de página se pueden producir como máximo en un acceso a memoria en este sistema?. Justificar brevemente.

Contestar Exclusivamente en los lugares asignados a tal efecto

El tamaño de la ruta de datos y por tanto el tamaño de las lecturas es de 16 bits. En la figura se representa el contenido de la MMU y de algunas posiciones de memoria relevantes para el problema. Los datos están organizados "big endian" (byte de mayor peso en dir más baja). Y La información del marco está en los bits de mayor peso de las palabras. Suponer además que en los bits de control la página que se busca está presente.

Contenido del TLB			Memoria de sust directa (directorio de primer nivel)		
Etiqueta	M.P.R.	Control	Posición	M.P.R.	Control
2345	CCE	X	0	CCC	X
F356	011	X	1	321	X
EF2C	CCD	X	2	132	X
0001	F2E	X	3	3B2	X
112D	001	X			
0303	123	X			
3454	2D4	X			
ABC1	DED	X			

MMU

Contenido de la Memoria Principal

Pos(hex)	Val(H)	Pos(hex)	Val(H)	Pos(hex)	Val(H)	Pos(hex)	Val(H)
00000	XX	1B001	A2	55C4A	CC	CCDB6	56
...	...	1B002	AA	55C4B	DD	CCDB7	78
00100	12	1B003	22
00101	13	8010F	45	DED09	40
00102	14	3B22E	31	80110	67	DED0A	20
00103	24	3B22F	32	80111	88	DED0B	06
...	...	3B230	55	80112	98
01130	22	3B231	CF	E0E11	22
01131	23	3B232	32	CCD57	01	E0E12	DE
01132	24	CCD58	33	E0E13	D3
01133	25	55400	22	CCD59	32
...	...	55401	23	CCD60	56	F1A0B	BB
132D4	E0	55402	24	CCD61	78	F1A0C	DE
132D5	E3
...	...	55C48	CC	CCDB4	20	FFFFE	XX
1B000	A0	55C49	CC	CCDB5	05	FFFFF	XX

Respuestas:

1) El tamaño del TLB y de la memoria de sustitución directa presente en la MMU, así como la longitud y campos de los descriptores de página en ella guardados:

Campos de la Dirección Virtual				Campos del TLB		
Nivel 1	Nivel 2	Nivel 3	Offset	Etiqueta	MPR	Control
2	7 bits	7 bits	8 bits	16 bits	12 bits	4 bits

Dirección Real		Tamaño de campos del descriptor	
M.P.R	Offset	M.P.R	control
12 bits	8 bits	12 bits	4 bits

Breve justificación del tamaño en que se dividen las direcciones:

DV: 24 bits; DR: 20 bits; Páginas: 256 Bytes = 28 Bytes => 8 bits de offset

En las tablas de páginas alojadas en páginas pueden guardar $256/2 = 128 = 27$ descriptores

Si hay 2 niveles de tabla en memoria queda para el primer nivel:

$$24(\text{total}) - 8(\text{offset}) - 7(\text{nivel3}) - 7(\text{nivel2}) = 2 \text{ bits para el nivel 1.}$$

Tamaño Total TLB: $8 \times 32 \text{ bits} = 256 \text{ bits} = 32 \text{ Bytes}$

Tamaño Total primer nivel en MMU:

La tabla de sustitución directa posee $2^2 = 4$ posiciones de 16 bits = 64 bits = 8 bytes-

2) Indicar los pasos que debe seguir la MMU cuando el procesador solicite las siguientes lecturas: EF2C58h y CC25B4h y que resultado se devolverá al procesador.

2.i) Dir Virtual: EF2C58h ; Dir Real: CCD58 h ; Dato Obtenido: 3332 h
Breve descripción:

Dada la dirección: EF2C58h; Página Virtual: EF2Ch; Offset: 58h.

La página virtual EF2Ch se encuentra en el TLB. Luego la DR=CCD58. El contenido de memoria es 3332h

2.ii) Dir Virtual: CC25B4h ; Dir Real: CCDB4 h ; Dato Obtenido: 2005 h
Breve descripción:

Dada la dirección: CC25B4h; Página Virtual: CC25h; Offset: B4h. La P.V. no está en el TLB.

$1100\ 1100\ 0010\ 0101 \Rightarrow L1=11\ (3h); L2 = 0011000\ (18h); L3 = 0100101\ (25h);$

En la mem de sustitución directa el índice 3 devuelve un marco 3B2h.

La dirección en la tabla de nivel 2 será: $3B200h + 18h \times 2 = 3B230h$. La entrada de la tabla es el valor 55CFh. El marco de la tabla de nivel 2 es 55Ch.

La entrada en la tabla de nivel 3 es: $55C00h + 25h \times 2 = 55C4Ah$. El contenido de ésta entrada es: CCDDh. Luego el marco es CCD.

La dirección real es DR: CCDB4. El contenido 2005h.

3) ¿Cuántas excepciones por falta de página se pueden producir como máximo en un acceso a memoria en este sistema?. Justificar brevemente.

El máximo de ocurrencias de fallos es tres. Si la dirección no está en el TLB no se produce excepción. Solo cuando en el descriptor de páginas el bit de presencia está a 0, se produce una excepción para traer la página de Nivel 2. Si al acceder a ésta el bit de presencia nuevamente ésta a cero se traerá la página que contiene la Tabla de páginas de nivel 3. Por último al componer la DR puede que no éste en memoria y se producirá el tercer y último fallo de página.

WR Escribe resultado en un registro.

a) Determinar el **CPI Real** si la predicción es estática i) efectiva o ii) no efectiva. En ambos casos, cuando se detecta la instrucción de salto, el procesador no se detiene. Para justificar su respuesta utilice las tablas adjuntas.

	T1	T2	T3	T4	T5	T6	T7	T8	T9
BCC	F1	F2	DE	RE	EX	M1	M2	WR	
N		F1	F2	DE	RE	EX	M1	M2	WR
N+1			F1	F2	DE	RE	EX	M1	M2
N+2				-	-	F1	F2	DE	RE
...
T				F1	F2	-	-	-	-
T+1					F1	-	-	-	-

	T1	T2	T3	T4	T5	T6	T7	T8	T9
BCC	F1	F2	DE	RE	EX	M1	M2	WR	
N		F1	F2	-	-	-	-		
N+1			F1	-	-	-	-		
...
T				F1	F2	DE	RE	EX	M1
T+1					F1	F2	DE	RE	EX
T+2						F1	F2	DE	RE

$$\text{CPI} = \text{CPI}_{\text{insaltos}} + \text{CPI}_{\text{saltos}} = 1,2 + 0,3 = 1,5 \text{ ciclos}$$
[illegible]

	T1	T2	T3	T4	T5	T6	T7	T8	T9
BCC	F1	F2	DE	RE	EX	M1	M2	WR	
N		F1	F2	DE	M1	-			
N+1			F1	F2	DE	-			
N+2				F1	F2	-			
N+3					F1	-			
...
T						F1	F2	DE	RE
T+1							F1	F2	DE

a.ii) Total CPI agregado con la estrategia de predicción NO efectiva:

Si salto no efectivo agrega 0 ciclos, si salto efectivo agrega 4 ciclos. Luego CPI adicional:

$$CPI_{\text{saltos}} = 0,15 * (0,25 * 0 + 0,75 * 4) = 0,45 \text{ ciclos}$$

$$CPI = CPI_{\text{sin saltos}} + CPI_{\text{saltos}} = 1,2 + 0,45 = 1,65 \text{ ciclos}$$

b) Determinar el CPI real si la predicción es dinámica con dos bits de control donde la predicción cambia tras dos fallos consecutivos. Suponer que por defecto se predice no efectivo.

La secuencia de saltos es de la forma E-E-E-E-NE-E-E-NE-E-E-NE-E-NE-E...(E: efectivo, NE: no efectivo)

b) Total CPI agregado con la estrategia de predicción dinámica:

Hay 2 predicciones NE con salto E que penaliza 4 ciclos. Luego con predicción efectiva hay 4/15 fallos y 9/15 aciertos que penalizan 2 ciclos

$$CPI_{\text{saltos}} = 0,15 * (2/15 * 4 + 4/15 * 2 + 11/15 * 2) = 0,34$$

$$CPI = CPI_{\text{sin saltos}} + CPI_{\text{saltos}} = 1,2 + 0,34 = 1,54 \text{ ciclos}$$

c) Suponga el siguiente trozo de código, y que el procesador no dispone de hardware para el adelantamiento de datos. Tenga en cuenta además que todas las etapas funcionan con flanco de subida, es decir en el mismo ciclo que se escribe un registro dado (etapa WR), éste no puede ser leído por otra instrucción en la etapa RE.

c.1) Reescriba el código agregando las instrucciones NOP necesarias y reordenando el código para que se ejecute en el menor tiempo posible. Cual sería el CPI para este código.

c.2) Si existiese adelantamiento de datos y respetando la emisión en orden. ¿Cuántos ciclos tardaría en ejecutarse este trozo de código?

I1:	ADD R3, R1, R2	;R3 <- R1+R2
I2:	LD R1, 100(R6)	;R1 <- M[100+R6]
I3:	AND R5, R3, R4	;R5 <- R3 and R4
I4:	NAND R6, R4, R2	;R6 <- R4 nand R2
I5:	OR R1, R1, R6	;R1 <- R1 or R6
I6:	ST R1, 4(R4)	;M[4+R4] <- R1
I7:	ST R3, 104(R4)	;M[104+R4] <- R3
I8:	SUB R3, R5, R6	;R3 <- R5 - R6

Respuesta c1: Reescritura del código

I1:	ADD R3, R1, R2	I9:	nop	I17:	
I2:	LD R1, 100(R6)	I10:	nop	I18:	
I3:	NAND R6, R4, R2	I11:	SUB R3, R5, R6	I19:	
I4:	nop	I12:	nop	I20:	
I5:	nop	I13:	ST R1, 4(R4)	I21:	
I6:	AND R5, R3, R4	I14:		I22:	
I7:	ST R3, 104(R4)	I15:		I23:	
I8:	OR R1, R1, R6	I16:		I24:	

CPI código C1:

$$\text{Total ciclos en ejecutarse el código: } N^{\circ} \text{ Inst} + \text{profPipe} - 1 = 13 + 8 - 1 = 20$$

$$CPI = \text{TotalCiclos} / N^{\circ} \text{ Inst.} = 20/8 = 2,5$$

c.2) Si existiese adelantamiento de datos y respetando la emisión en orden. ¿Cuántos ciclos tardaría en ejecutarse este trozo de código?

Respuesta c.2

Si existe adelantamiento de datos, lo único que puede hacer perder ciclos es la instrucción LD (en este caso cuando en los dos ciclos sucesivos se utilice el registro donde escribe el LD). Como no es el caso tendremos:

$$\text{Total Ciclos} = N^{\circ} \text{ Instrucciones} + \text{profPipe} - 1 = 8 + 8 - 1 = 15.$$

Esta respuesta no es necesaria. Solo se agrega para aclarar el resultado

		T1	T2	T3	T4	T5	T6	T7	T8	T9	T10	T11	T12	T13	T14	T15	T16	T17	T18	T19	T20
I1	ADD R3, R1, R2	F1	F2	DE	RE	EX	M1	M2	WR ¹												
I2	LD R1, 100(R6)		F1	F2	DE	M1	EX	M1	M2	WR ²											
I3	NAND R6,R4,R2			F1	F2	DE	M1	EX	M1	M2	WR ^{2,3}										
I4	Nop				F1	-	-	-	-	-	-	-									
I5	Nop					F1	-	-	-	-	-	-									
I6	AND R5, R3, R4						F1	F2	DE	RE ¹	EX	M1	M2	WR ³							
I7	ST R3, 104(R4)							F1	F2	DE	RE	EX	M1	M2	WR						
I8	OR R1, R1, R6								F1	F2	DE	RE ²	EX	M1	M2	WR ⁴					
I9	nop									F1	-	-	-	-	-	-	-				
I10	nop										F1	-	-	-	-	-	-	-			
I11	SUB R3, R5, R6											F1	F2	DE	RE ³	EX	M1	M2	WR		
I12	nop												F1	-	-	-	-	-	-	-	
I13	ST R1, 4(R4)													F1	F2	DE	RE ⁴	EX	M1	M2	WR

Notas:

1. Al siguiente ciclo de la escritura del reg R3 por I1, puedo leerlo en la I6
2. LD escribe R1 en T9 y NAND R6 en T10. Luego OR lo puede leer en el siguiente ciclo (T10)
3. La instrucción SUB debe esperar la escritura de R5 y R6. Al R5 lo escribe I6 en T13.
4. ST R1, se puede ejecutar después que OR de la I8 haya escrito el R1 en T15.

I1, I2 e I3: se puede ejecutar ya que no depende de ninguna otra instrucción.

I7: puedo ejecutar ST R3, 104(R4) ya que R3 está disponible desde la ejec de I1 (desde T9). Se puede ejecutar un ciclo antes, pero retrasar el AND retrasa toda la ejecución.

El enunciado fijaba la primera instrucción, aunque existe otra solución del mismo tiempo de ejecución iniciando con LD, NAND y ADD.

P2.- Se dispone de un procesador VLIW con 4 unidades de ejecución. Una de load/store y saltos que necesita 2 ciclos de ejecución, una unidad de enteros (ALU) de 2 ciclos y dos unidades de punto flotante que requiere de 4 ciclos. Todas las unidades de ejecución son segmentadas (en cada ciclo de reloj puede empezar una nueva ejecución). Los ciclos se refieren a la latencia total del de la segmentación (pipeline). Se utiliza este procesador para calcular el siguiente programa que multiplica cada elemento de un vector de 6 elementos por una constante y además calcula la suma de los elementos del arreglo: Los elementos del arreglo (A[i]) son en coma flotante de tamaño Word (4 bytes):

```
for (i = 5; i > 0; i--)
{ Suma = Suma + A[i]
  A[i] = A[i] * cte; }
}
```

El código ensamblador (al margen de las inicializaciones) correspondiente es:

```
Loop:   Load R2, 100(R1)      ; R1 es el apuntador al arreglo
        Addf R3, R3, R2      ; R3 lleva la suma
        Mulf R4, R2, R5      ; En R5 la constante
        Store 100(R1), R4    ; guarda el Valor A[i]
        Sub R1, R1, 4        ; El P. flotante es de 32 bits
        Bnez R1, loop
```

a) ¿Cuántos ciclos necesita para ejecutarse este código si no se aplica ninguna optimización?. Utilice la tabla adjunta que representa las 4 unidades de ejecución:

Ciclo	LD/ST saltos	ALU enteros	Coma Flotante 1	Coma Flotante 2
1	Load R2, 100(R1)	Nop	Nop	Nop
2	Nop	Nop	Nop	Nop
3	Nop	Nop	Addf R3, R3, R2	Mulf R4, R2, R5
4	Nop	Nop	Nop	Nop
5	Nop	Nop	Nop	Nop
6	Nop	Nop	Nop	Nop
7	Store 100(R1), R4	Sub R1, R1, 4	Nop	Nop
8	Nop	Nop	Nop	Nop
9	Bnez R1, loop	Nop	Nop	Nop
10	Nop	Nop	Nop	Nop

Tiempo total de ejecución P2.a: Tiempo: 10*6 = 60 ciclos

b) Utilizando Desenrollamiento de bucles, Actualización de referencias y Renombramiento de registros en cuántos ciclos se puede ejecutar el código anterior. Suponga que no tiene límite en la cantidad de registros y nombres de la forma Rxx (xx números). Utilice la tabla adjunta

Ciclo	LD/ST saltos	ALU enteros	Coma Flotante 1	Coma Flotante 2
1	Load R2, 100(R1)	Nop	Nop	Nop
2	Load R12, FC(R1)	Nop	Nop	Nop
3	Load R22, F8(R1)	Nop	Nop	Mulf R4, R2, R5
4	Load R32, F4(R1)	Nop	Addf R3, R2, R12	Mulf R14, R12, R5
5	Load R42, F0(R1)	Nop	Nop	Mulf R24, R22, R5
6	Load R52, EC(R1)	Nop	Addf R13, R22, R32	Mulf R34, R32, R5
7	Store 100(R1), R4	Nop	Nop	Mulf R44, R42, R5
8	Store FC(R1), R14	Nop	Addf R23, R42, R52	Mulf R54, R52, R5
9	Store F8(R1), R24	Nop	Nop	Nop
10	Store F4(R1), R34	Nop	Addf R3, R3, R13	Nop
11	Store F0(R1), R44	Nop	Nop	Nop
12	Store EC(R1), R54	Nop	Nop	Nop
13	Nop	Nop	Nop	Nop
14	Nop	Nop	Addf R3, R3, R23	Nop
15	Nop	Nop	Nop	Nop
16	Nop	Nop	Nop	Nop
17	Nop	Nop	Nop	Nop

Tiempo total de ejecución P2.b: Tiempo: 17 ciclos

c) Si cada unidad de operación requiere instrucciones de 32 bits. ¿Qué tamaño tiene el código de la respuesta a y b sin considerar ningún tipo empaquetado?

El código a: 10 instrucciones x 4 codop x 32 bits / (8 bits/Byte) = 10 * 16 = 160 bytes.
El código b: 17 instrucciones x 4 codop x 32 bits / (8 bits/Byte) = 17 * 16 = 272 bytes.