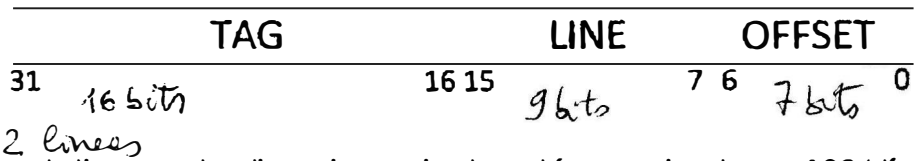
Pregunta	si	no
1. La Figura muestra el direccionamiento de una memoria cache de mapeo directo en un procesador de 32 bits.	☐	☒



16 bits 9 bits 7 bits

$2^9 = 512$ líneas

La cache relativa a este direccionamiento está organizada en 1024 líneas cada una de las cuales almacena bloques de 32 palabras. ¿Es eso cierto?

2. Considere un procesador segmentado y asuma que el 20% del tiempo haya dependencias de datos que introducen 1 ciclo adicional y el 10% del tiempo de ejecución haya otro tipo de dependencia que introducen 2 ciclos adicionales. ¿Es cierto que el CPI medio de este procesador es 1.7?	☐	☒

$CPI = 0,7 \cdot 1 + 0,2 \cdot 2 + 0,1 \cdot 3 = 0,7 + 0,4 + 0,3 = 1,4$

3. Un procesador tiene una frecuencia de reloj de 2GHz y un CPI medio de 1.5. Si el procesador ejecuta un programa en 15 segundos el programa en cuestión estará formado por 2.5×10⁹ instrucciones. ¿Es eso cierto?	☐	☒

$2\,GHz \rightarrow t_{ciclo} = 0,5\,ns \rightarrow t_{exec\,instr} = 1,5 \cdot 0,5 = 0,75\,ns \rightarrow t_{exec\,programa} = 2,5 \cdot 10^9 \cdot 0,75 \cdot 10^{-9} = 1,875\,s$

4. A=**10010011** y B=**00111110** representan dos números binarios en magnitud y signo. ¿Es cierto que el |A|>|B|?	☐	☒

5. El número **3E4CF000** es un número en coma flotante de precisión sencilla en formato IEEE 754. ¿Es cierto que este número representa **0.1575** en base 10?	☐	☒

PREGUNTA 2.1. Considere el fragmento de código MIPS que se muestra a continuación:

```
add  $t0, $s0, $s1
xor  $t1, $t0, $s2
lw   $s0, -12($a0)
sub  $s5, $s0, $s1
```

(a) Identifique el número y el tipo de dependencias de dato en el fragmento anterior.
(b) ¿Es posible resolver alguna de las dependencias halladas en el punto anterior, reordenando las instrucciones de manera que no sea necesario introducir caminos de adelantamiento (bypass)? En caso afirmativo muestre cómo, en caso negativo explique el porqué.
(c) Finalmente, haga un ejemplo sencillo de dependencia de dato que no pueda ser resuelta utilizando caminos de adelantamiento.

PREGUNTA 2.2. Escriba el código ensamblador MIPS que implementa una función **`atoi(s)`** que devuelve un entero **`d`** que es la representación numérica de la cadena de caracteres **`s`** terminada por el carácter '\n' (retorno de carro). Es decir, si por ejemplo s = "105\n" es un vector de 4 caracteres ('1', '0', '5' y '\n'), la función **`atoi("105\n")`** deberá devolver el número entero 105. Antes de diseñar el algoritmo de conversión tenga en cuenta lo siguiente:

1. En la representación ASCII, los caracteres que representan los dígitos de 0 a 9 se hallan en posiciones consecutivas, siendo el carácter '0' el primero y el '9' el último. Por tanto, el número representado por el carácter **`c`** es simplemente **`c`**- '0'.
2. De forma análoga a los lenguajes de programación de alto nivel, en el lenguaje ensamblador del MIPS, las comillas '**`c`**' devuelven el código ASCII del carácter **`c`**.

Asuma que **`$t0`** contenga la dirección de la cadena de caracteres a convertir, utilice **`$t1`** para las operaciones intermedias y guarde el resultado de la conversión en **`$t2`**.

PREGUNTA 2.3. Considere el fragmento de código siguiente:

```
lw  $s2, 0($s1)
lw  $s1, 40($s3)
sub $s3, $s1, $s2
add $s3, $s2, $s2
or  $s4, $s3, $zero
sw  $s3, 50($s1)
```

Asuma el cauce de 5 etapas del MIPS (IF-ID-EX-MEM-WB) sin caminos de adelantamiento de datos y con un circuito que atasca el cauce en presencia de dependencias de datos. ¿Cuántas veces se producirán atascos? ¿Cuántos ciclos tendrá cada atasco? Dibuje el diagrama de ejecución del fragmento anterior y calcule cuántos ciclos tarda

el fragmento anterior en ejecutarse. Finalmente coloque una **nop** en el código anterior para eliminar las dependencias, dibuje el nuevo diagrama de ejecución y calcule el nuevo número de ciclos de ejecución.

PROBLEMA 3.1. Considere una memoria cache de mapeo directo con un direccionamiento de 32 bits: 17 bits de campo de etiqueta (TAG), 10 bits de campo de índice (INDEX) y 5 bits de campo de OFFSET. Se pide:

(a) Calcular el número de filas y el tamaño de bloque de la memoria

(b) El tamaño total (en bytes) de la cache, considerando que cada fila tiene también un bit de validez. ¿Cuál es la relación entre número de bits totales y bits de datos efectivamente almacenados en la cache?

(c) A partir del encendido del procesador, las siguientes direcciones de bytes referencian la cache: 0, 4, 16, 132, 232, 160. 1024, 30, 140, 2100, 180, 2180. Rellene la tabla que se muestra a continuación y diga cuántos bloques son sustituidos con este patrón de acceso a memoria.

Dirección	0	4	16	132	232	160	1024	30	140	3100	180	2180
#fila	0	0	0	4	7	5	32	0	4	96	5	68
Hit/Miss	M	H	H	M	M	M	M	H	H	M	H	M
Sustitución	N	N	N	N	N	N	N	N	N	N	N	N

Cada bloque de la cache tiene 32 bytes por tanto la dirección de memoria expresada en bloques es: $\left\lfloor \frac{\text{dirección}}{32} \right\rfloor$

Addr	# block
0	0
4	0
16	0
132	4
232	7
160	5
1024	32
30	0
140	4
3100	96
180	5
2180	68

El mapeo será:

#fila = # bloque de memoria mod tamaño cache en bloques

Tamaño cache = 2^{10} filas · $\frac{1 \text{ bloque}}{\text{fila}}$ = 1024 bloques

Ningún bloque es sustituido

Parte 2

PREGUNTA 2.1

Punto a: dos dependencias RAW:

1) add → xor 2) lw → sub

Punto b: hay dos posibles reordenamientos:

add, lw, sub, xor } elimino (1)

lw, add, xor, sub — elimino (2)

Punto c:

```
lw $t0, 0($t1)
add $t2, $t0, $t1
```

Las dependencias RAW en las que el "writer" es una instrucción de tipo R pueden manejarse ~~con~~ sin cortocircuitos

PREGUNTA 2.2

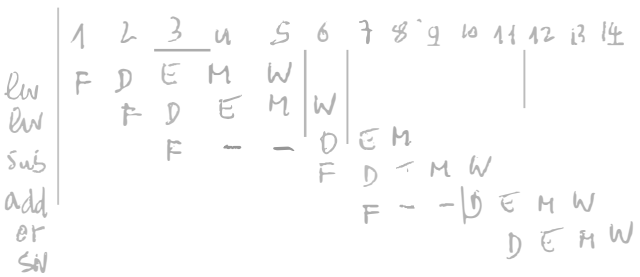
Código ensamblador:

```
    li $t2, 0
loop: lb $t1, ($t0)
    addu $t0, $t0, 1
    beq $t1, '\n', end_loop
    blt $t1, '0', end_loop
    bgt $t1, '9', end_loop
    mul $t2, $t2, 10
    sub $t1, $t1, '0'
    add $t2, $t2, $t1
    j loop
end_loop: ____
```

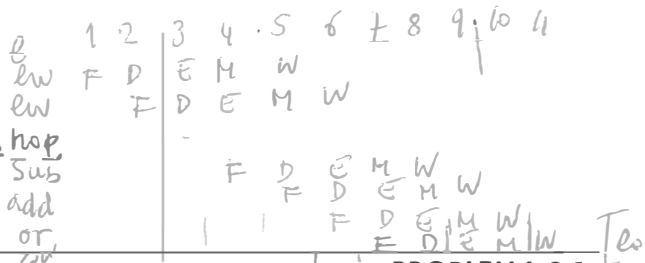
```
blt $t1, '0', end_loop
        ||
slt $t0, $t1, '0'
bne $t0, $zero, end_loop
```

```
bgt $t1, '9', end_loop
        ||
slt $t0, '9', $t1
bne $t0, $zero, end_loop
```

PREGUNTA 2.3



Se producen 2 atascos y cada uno dura dos ciclos.



nop después del segundo lw de manera que el valor de $s1 puede ser encaminado hacia la etapa MEM de la sub

[La secuencia tarda 11 ciclos]

PROBLEMA 3.1

Parte 3

Punto a:

$2^{10} = 1024$ filas $2^5 = 32$ bytes/bloque

Punto b:

tamaño fila = 1 + 17 + 32 - 8 = 274 bits

tamaño = $1024 \cdot \frac{274}{8} = 35.072$ bytes

Punto c:

ningún bloque es sustituido

$\frac{32 - 8 \cdot 1024}{274 \cdot 1024} =$

$\frac{256}{274} = 0,934$

$= 93,4\%$