

Preguntas que pueden caer en el examen de aap:

1- Tipos de riesgos , porque se producen y cómo se pueden eliminar.

-Riesgos de datos:

-Se producen cuando una instrucción depende de los resultados de una instrucción anterior que aún no ha completado su ejecución.

-Se pueden eliminar utilizando la anticipación de datos para predecir y resolver las dependencias de datos.

-Riesgos estructurales:

-Se producen cuando múltiples instrucciones compiten por el mismo recurso funcional, como una unidad de ejecución o una memoria. Esto se debe a que la realización de ciertas etapas está construida con la misma unidad funcional.

-Se pueden eliminar aumentando el número de unidades funcionales o utilizar técnicas de emisión y reordenamiento de instrucciones para evitar cuellos de botella en la ejecución.

2- Los MIPS pueden variar inversamente al rendimiento. Justificar esta afirmación poniendo un ejemplo.

Esto se debe a que el rendimiento real de un procesador está influenciado por varios factores, como las posibles diferencias en la arquitectura y la eficiencia de ejecución de las instrucciones, y simplemente medir la cantidad de MIPS puede ser erróneo. Poner ejemplo tipo ejercicio 3 de la lista, en el cual se muestra como muchas veces no es mejor comparar MIPS si no comparar el tiempo de ejecución.

3- Tipos de formato de instrucción, identificación de los campos que lo componen y significado de estos campos: *

add \$5, \$7, \$8

addi \$5, \$7, 40

srl \$5,\$7, 4

lw \$5,20(\$7)

j 30

beq \$5,\$7,40

Instrucción	Tipo de formato	Operación	Registro de destino (rd)	Primer operando/registro fuente (rs)	Segundo operando (rt)	Shamt (cantidad de desplazamiento)	Dirección de destino	Valor inmediato	Desplazamiento (offset)	Función (función específica que realiza)
add \$5, \$7, \$8	R	add	\$5	\$7	\$8	x	x	x	x	suma
addi \$5, \$7, 40	I	addi	\$5	\$7	x	x	x	40	x	x
srl \$5, \$7, 4	R	srl	\$5	\$7	x	4	x	x	x	desplazamiento a la derecha lógico
lw \$5, 20(\$7)	I	lw	\$5	\$7	x	x	x	x	20	x
j 30	J	j	x	x	x	x	30	x	x	x
beq \$5, \$7, 40	I	beq	\$5	\$7	x	x	x	x	40	x

**4- ¿Qué tipo de dependencia hay en el siguiente código?
¿Cómo gestiona la unidad de anticipación la siguiente
secuencia de código? ¿Cómo harías para resolver el problema
que se presenta para optimizar el número de ciclos? Razona tu
respuesta:**

```
lw $5, 10($2)
add $6, $5, $5
```

Hay una dependencia de datos de tipo EX, ya que afecta a la etapa de ejecución de la operación add.

En este caso, la unidad de anticipación detecta que la instrucción add depende del resultado de lw, y en lugar de esperar a que lw complete su escritura en el registro \$5, la unidad de anticipación adelanta el valor cargado directamente a la etapa de ejecución de add. Aunque en este caso en concreto, siendo este el único en el que pasa, aun con una unidad de anticipación activa, hay que poner una burbuja entre estas dos si está activo escritura y lectura simultánea y 3 si no está activo, ya que a la unidad de anticipación no le da tiempo. Si no queremos meter burbujas, y el programa lo permite, pondría instrucciones entre media de estas para que no haya ciclos muertos.

5- Diferencias entre arquitectura Harvard y arquitectura Von Neumann. ¿Cual de las dos arquitecturas es mejor para aplicar la mejora de rendimiento de la segmentación? Razona tu respuesta

6- ¿Cómo se establece el periodo de reloj T del procesador monociclo MIPS, y en qué unidad funcional actuamos con el reloj para sincronizar el procesador? Justifica tu respuesta.

En un procesador monociclo MIPS, el periodo de reloj T se establece de manera que sea suficiente para que todas las etapas del pipeline se completen antes de que comience la siguiente instrucción. Las etapas en un pipeline MIPS monociclo incluyen:

IF (Instrucción Fetch): Se lee la instrucción desde la memoria de instrucciones.

ID (Decodificación de Instrucción): Se decodifica la instrucción y se leen los registros si es necesario.

EX (Ejecución): Se realiza la operación principal de la instrucción (puede incluir la ALU).

MEM (Acceso a Memoria): Se accede a la memoria si es necesario (por ejemplo, para cargar o almacenar datos).

WB (Write Back): Se escriben los resultados de la instrucción en los registros.

La sincronización del procesador se realiza principalmente a través de la unidad de control, que actúa en conjunción con el reloj para coordinar y secuenciar las operaciones en cada etapa del pipeline.

¿Cómo se establece el periodo de reloj T del procesador segmentado MIPS? Justifica tu respuesta.

En un procesador segmentado MIPS, el periodo de reloj debe ser lo suficientemente largo para permitir que todas las etapas de todos los segmentos se completen antes de pasar a la siguiente instrucción. La segmentación implica dividir el camino de datos en etapas más pequeñas y realizar varias instrucciones en paralelo en estas etapas. Cada segmento puede tener su propio conjunto de etapas, como IF (Instrucción Fetch), ID (Decodificación de Instrucción), EX (Ejecución), MEM (Acceso a Memoria), WB (Write Back), etc.

El periodo de reloj T se establece teniendo en cuenta las siguientes consideraciones:

Duración de la etapa más larga: El T debe ser mayor o igual al tiempo necesario para completar la etapa más larga en cualquier segmento. Esto garantiza que todos los segmentos tengan suficiente tiempo para ejecutar sus operaciones.

Sincronización entre segmentos: Dado que los segmentos operan en paralelo, es necesario coordinar y sincronizar sus operaciones.

Evitar hazards entre segmentos: Se deben tener en cuenta los posibles riesgos (hazards) entre los diferentes segmentos. El periodo de reloj debe ser lo suficientemente largo para permitir la ejecución sin problemas y para gestionar cualquier tipo de hazard que pueda surgir debido a la segmentación.

La unidad de control juega un papel crucial en la coordinación y sincronización de las operaciones de los diferentes segmentos. Es responsable de generar las señales de control adecuadas para dirigir el flujo de datos y garantizar que las operaciones se realicen en el orden correcto.

7- Cuáles de las siguiente afirmaciones son falsas:

- a. Menor MIPS menor rapidez de ejecución. Verdadera
- b. Mayor MIPS menos rapidez de ejecución. Falsa
- c. Menor MIPS mayor rapidez de ejecución. Falsa

8- Modificaciones en el camino de datos y en la estructura del procesador MIPS teniendo en cuenta los siguiente formatos de datos en instrucciones:

Formato de datos: ancho de palabra de 32 bits.

Formato de instrucciones: ancho de palabra de 16 bits con los mismos campos y tamaños de cada uno de los campos que en el procesador de 32 bits

Opc code: 6 bits

Registro Rs: 5 bits

Registro Rd: 5 bits

Registro Rt: 5 bits

Shamt: 5 bits

Func.: 6 bits

Reducción de Tamaño de Palabra: Cambiamos el ancho de palabra de 32 bits a 16 bits para las instrucciones.

Reorganización de Campos: Acomodamos los campos de la instrucción según la nueva palabra de 16 bits, manteniendo los tamaños específicos para cada campo.

Opc Code, Registros y Funciones: Ajustamos la longitud de los campos Opc Code, Registro Rs, Registro Rd, Registro Rt, Shamt y Func. según los nuevos tamaños de palabra.

Camino de Datos: Modificamos el camino de datos para manejar instrucciones de 16 bits en lugar de 32 bits.

Instrucciones de Salto: Ajustamos las direcciones de destino de salto para que se ajusten a la nueva longitud de palabra.

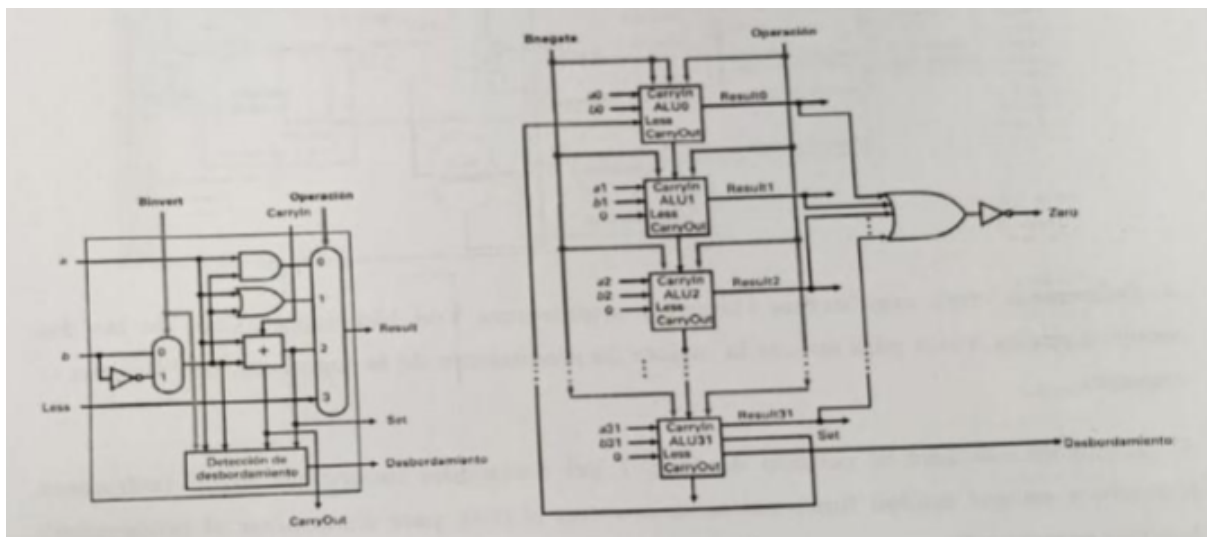
Extensión de Signo: Añadimos extensiones de signo a los campos que lo requieran para mantener la semántica de las operaciones

Actualización del Decodificador: Modificamos el decodificador de instrucciones para interpretar las nuevas instrucciones de 16 bits y generar las señales de control correspondientes.

Control de Riesgos: Revisamos y ajustamos el manejo de riesgos para adaptarse a las nuevas instrucciones de 16 bits.

Pruebas y Depuración: Realizamos pruebas exhaustivas para asegurarnos de que el procesador modificado funcione correctamente con las nuevas instrucciones.

9. Teniendo en cuenta estas dos figuras, explica como funciona la operación inicializar sobre menor que (set-on-less-than)

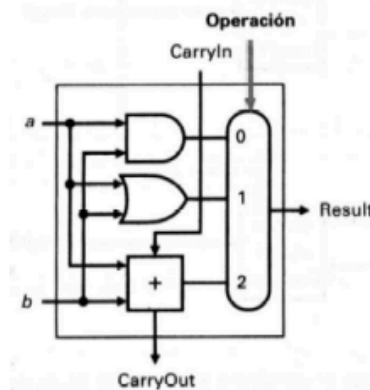
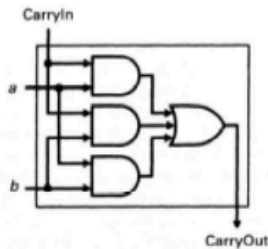


Lo mas comun es el acarreo en serie, se pone de primeras a 0, el Cout se pasa a la siguiente etapa al Cin

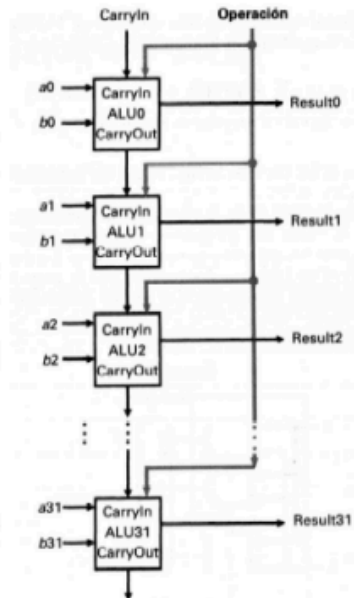
ARITMÉTICA PARA COMPUTADORES

CONSTRUCCIÓN DE UNA UNIDAD ARITMÉTICO LÓGICA

- Bloque sumador
- ALU completa de 1 bit (operaciones and, or y suma)



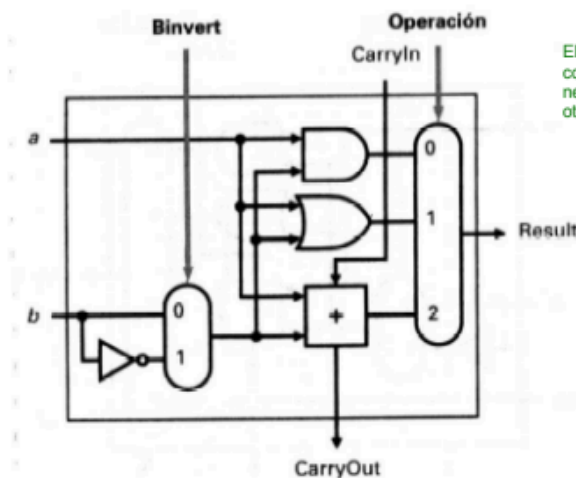
- ALU de 32 bits



ARITMÉTICA PARA COMPUTADORES

CONSTRUCCIÓN DE UNA UNIDAD ARITMÉTICO LÓGICA

- ALU de 1 bit con las operaciones:
A AND b ; a OR b ; a + b ; a + b' Sumar el complemento a 1



El multiplexor, se compone de una OR y a esta conectada dos AND, una patilla de una AND esta negada y puentada con otra de la otra AND, y las otras dos puentadas y a la variable

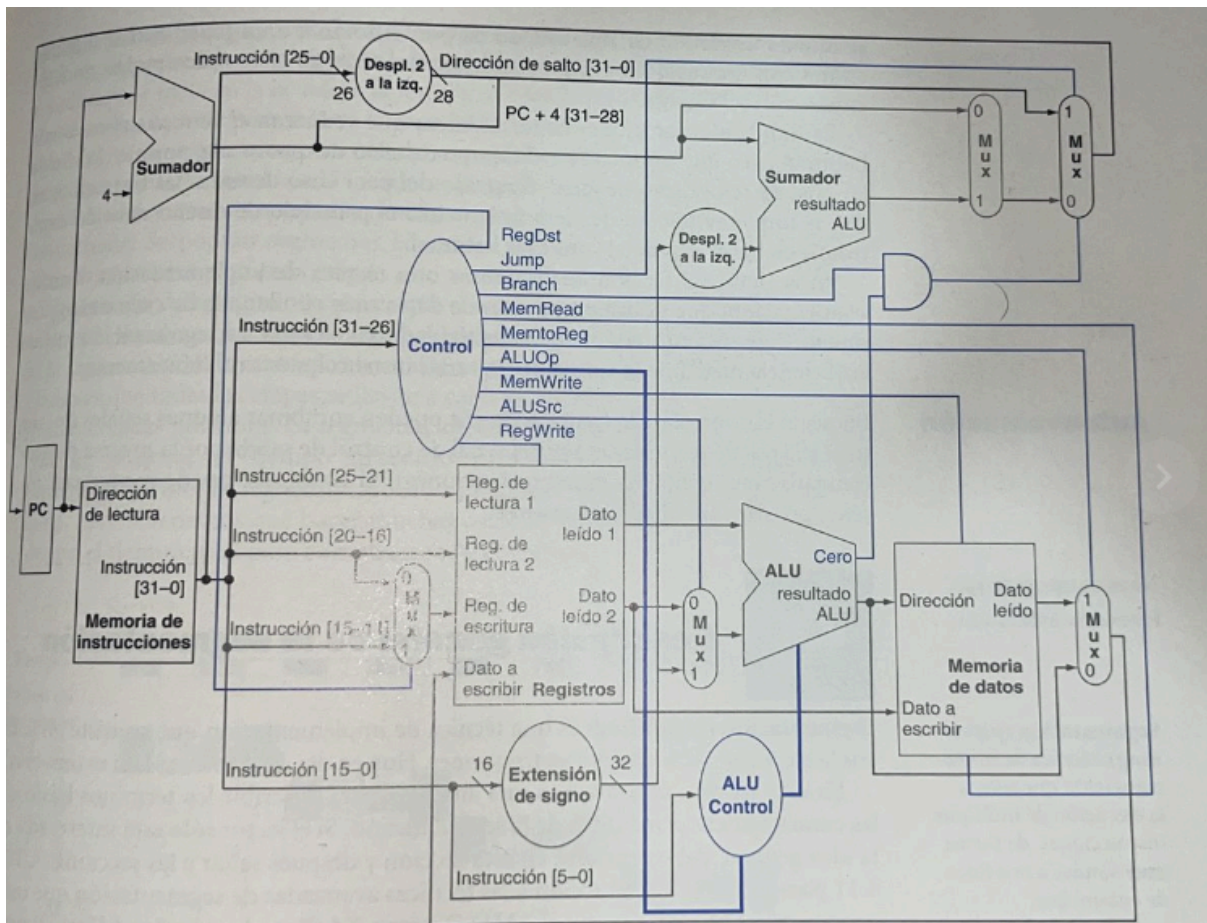
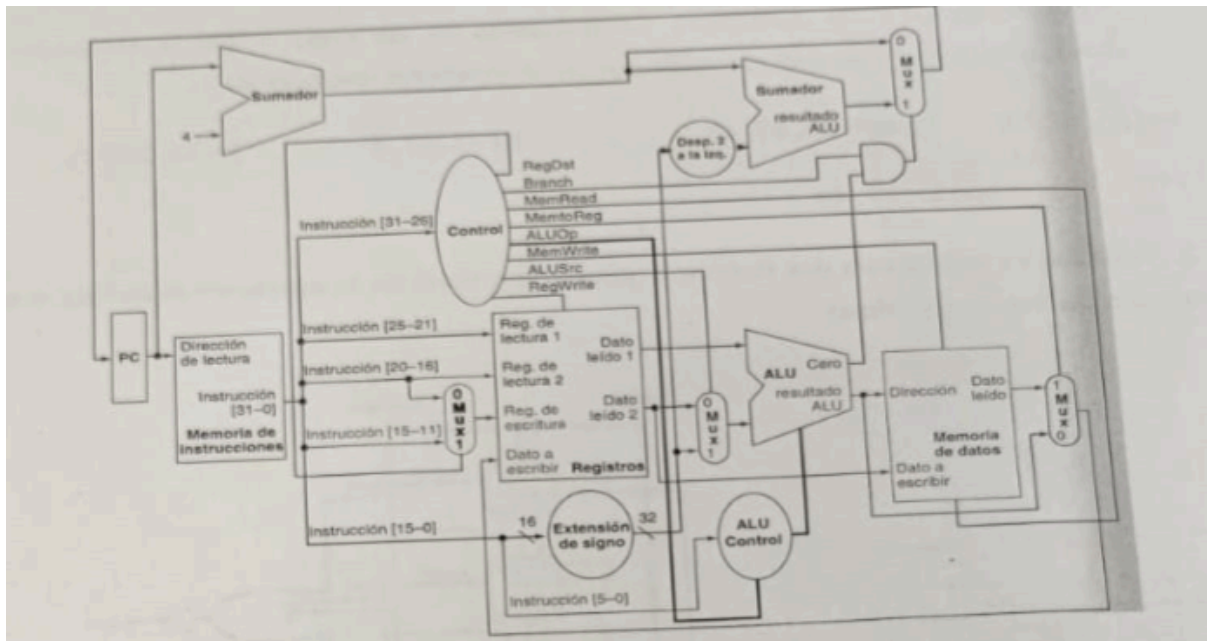
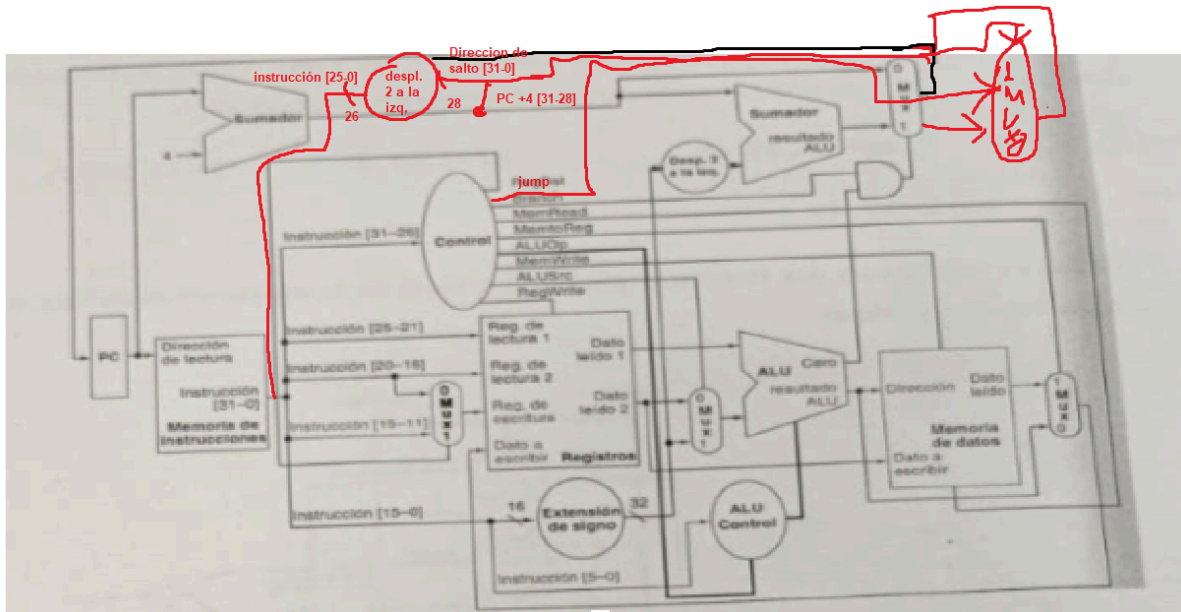


FIGURA 4.24 Un camino de datos sencillo y su control extendido para ejecutar instrucciones de salto incondicional. Se utiliza un multiplexor adicional (en la esquina superior derecha) para elegir entre el destino de la instrucción de salto incondicional y la dirección de la siguiente instrucción en orden secuencial. Este multiplexor se controla mediante la señal de salto incondicional. La dirección de destino



11- Si el tiempo para una operación ALU puede reducirse un 25%¿afectará a la mejora obtenida por la segmentación?¿Y si el tiempo de dicha operación aumenta un 25%?

- Si mejora, siempre que tomemos este nuevo tiempo de la ALU como el nuevo periodo de reloj. Solo mejora si la ALU era el segmento que marcaba el periodo de reloj.
- Si, si tomamos el nuevo tiempo de la ALU como nuevo periodo de reloj.

12-

6.- Suponiendo un camino de datos segmentado con unidad de detección de riesgos y unidad de anticipación, determinar qué registros habrán sido leídos y cuáles escritos en el quinto ciclo de reloj de la ejecución de:

- | | | | |
|----|-----------------|----|-----------------|
| a) | add \$2,\$3,\$1 | b) | add \$2,\$3,\$1 |
| | sub \$4,\$3,\$5 | | sub \$4,\$3,\$5 |
| | add \$5,\$3,\$7 | | add \$8,\$2,\$6 |
| | add \$7,\$6,\$1 | | add \$7,\$6,\$1 |
| | add \$8,\$2,\$6 | | add \$5,\$3,\$7 |

- No existen posibles riesgos.
- Posibles riesgos de datos:
 \$s2→ Cuando uno esta en acceso a memoria otro esta en lectura.
 \$s7→ Cuando uno está en ejecución, el otro está en lectura.

Solo son posibles riesgos porque gracias a la unidad de anticipación de datos, son solucionados.

Único caso de Riesgo que no es capaz de ser anticipado:

```
lw $s7,0($s1)
add $s2,$s7,$s6
```

Para solucionar este riesgo, la solución es usar burbujas (nop) u otra instrucción.

Las instrucciones R son:

Suma y resta, los que empiezan por s (menos sw, que es I) y los que acaban en r

Las instrucciones I son:

beq, benq, los que acaban en i y los que empiezan por L.

Las instrucciones J son:

j y jal. Todos los saltos menos jr

5.- Utilizando la figura siguiente y las instrucciones que se están ejecutando, determinar el valor de cada campo en los cuatro registros de segmentación en el ciclo que se muestra.

lw \$10,9(\$1)
sub \$11,\$2,\$3
and \$12,\$4,\$5
or \$13,\$6,\$7
add \$14,\$8,\$9

Suponer que el estado inicial de la máquina era:

- PC tenía el valor 500_{diez}, dirección de la instrucción lw.
- Cada registro tenía el valor inicial 10_{diez} más el número de registro (el registro \$8 tenía el valor 18_{diez}).
- Cada palabra de memoria accedida como dato tenía el valor inicial 1000_{diez} más la dirección del byte de la palabra (Memoria[8] tenía el valor inicial 1008_{diez}).

