

# TEST TEORIA

## TEMA 1

1. ¿En qué generación, dentro de la historia de los computadores digitales, se alcanzaron tiempos de conmutación del orden de nanosegundos?

- a) primera
- b) segunda
- >c) tercera
- d) cuarta

2. En una arquitectura RISC típica:

- > a) se usa un porcentaje elevado de las instrucciones del repertorio.
- b) no puede usarse segmentación.
- c) la programación resulta mucho más simple que en una arquitectura CISC.
- d) la UC es más compleja que en una arquitectura CISC.

3. ¿Cuál de las siguientes afirmaciones es incorrecta?

- a) El repertorio de instrucciones es el conjunto de operaciones que es capaz de interpretar la unidad de control.
- b) El modo de direccionamiento permite determinar un operando o la ubicación del operando.
- > c) Los operandos siempre están almacenados en memoria.
- d) El repertorio de instrucciones debe ser capaz de realizar una tarea en un tiempo finito.

4. Un modo de direccionamiento en el que se especifica un registro y una dirección de memoria cuyo contenido se suma al contenido del registro base para obtener la dirección efectiva, se conoce como:

- a) base con desplazamiento
- b) directo o absoluto
- c) indirecto a registro través de memoria
- > d) ninguno de los anteriores

5. El ancho de palabra de una memoria corresponde a:

- a) El número que identifica únicamente cada posición de la memoria.
- > b) La cantidad de bits que caben en una sola posición
- c) La longitud del registro de direcciones de la memoria.

d) El número de posiciones que la componen.

6. En la captación de la instrucción:

a) en MAR indicamos la dirección donde está la instrucción y en la ALU recogemos la instrucción.

b) en MBR indicamos la dirección donde está la instrucción y en la ALU recogemos la instrucción.

-> c) en MAR indicamos la dirección donde está la instrucción y en MBR recogemos la instrucción.

d) en MBR indicamos la dirección donde está la instrucción y en MAR recogemos la instrucción.

7. Un sistema con direcciones de 8bits utiliza una puerta NAND conectada a las líneas A7...A5 para atacar la entrada CS# (activa baja) de un módulo de memoria. En el mapa de memoria las siguientes posiciones corresponderán a dicho módulo

a) 0x00 a 0x1f

b) 0x00 a 0x0f y 0x80 a 0x8f

-> c) 0xe0 a 0xff

d) 0x70 a 0x7f y 0xf0 a 0xff

8. ¿En qué generación, dentro de la historia de los computadores digitales, aparece la memoria cache?

a) primera

b) segunda

-> c) tercera

d) cuarta

9. Una instrucción máquina del tipo "Add M,R" podría formar parte del repertorio de

a) una máquina con arquitectura R/R

sería ADD R1,R2

b) una máquina pila

sería ADD

-> c) una máquina con arquitectura M/M

podría ser también ADD M,M

d) una máquina de acumulador

sería ADD M

10. ¿Cuál de las siguientes no es una característica de los computadores RISC?

- a) Un computador RISC no debe emplear microprogramación.
- b) Para acelerar el computador RISC se emplean técnicas de pipelining.
- > c) Las funciones que realizan los computadores RISC deben ser lo más complejas y potentes que sea posible.
- d) La decodificación de las instrucciones debe ser simple: un computador RISC debería emplear un único formato de instrucción

11. ¿Qué tipo de direccionamiento se usa para el registro destino en la instrucción IA32 add

array(%ebx,4), %edx?

- a) Direccionamiento indexado
- > b) Direccionamiento a registro
- c) Direccionamiento relativo a registro base
- d) Direccionamiento inmediato

12. ¿Qué novedad se desarrolló en la tercera generación de computadores?

- a) Los microprocesadores CISC
- > b) Los circuitos integrados
- c) Los microprocesadores RISC
- d) Los primeros lenguajes de programación de alto nivel

13. En un procesador de la familia 80x86 las posiciones de memoria que representan una

variable long (entero 4B compl.2) contiene los bytes: F0 FF FF FF. ¿Cuánto vale dicha variable?

- > a) -16
- b) 16
- c) 4294967280
- d) 4043309055

14. En el direccionamiento inmediato, tras captarse completamente la instrucción:

- > a) se accede al operando, que es una constante contenida en la propia instrucción.
- b) el código de operación contiene el operando.
- c) se accede al operando, que está contenido en una posición de memoria principal.
- d) se accede al operando, que se encuentra almacenado en uno de los registros programables.

15. Si almacenamos según el criterio little-endian la palabra de 64 bits 0xFACEB00C a partir

de la dirección 0xCAFEBAE, el byte 0xCE quedará almacenado en la dirección:

- a) 0xCAFEBAE
- b) 0xCAFEBAE1
- c) 0xCAFEBAF
- > d) 0xCAFEBAC0

16. ¿Cuál de las siguientes afirmaciones es cierta?

- a) la unidad de control necesita como entrada el registro contador de programa para saber cuál es la instrucción que debe ejecutar a continuación  
realmente la UC copia PC en MAR, y lo que lee en MDR (captación) lo lleva a IR que sí es entrada a la UC (decodificación)
- b) la arquitectura von Neumann de los computadores tradicionales consiste en tener almacenados los datos separados de las instrucciones en memorias distintas  
vimos en [T4.2EjmSeg] tr.12 que tener memoria de instrucciones separada de la de datos es una variante llamada "arquitectura Harvard", llamándose "arquitectura Princeton" la original  
[https://en.wikipedia.org/wiki/Von\\_Neumann\\_architecture](https://en.wikipedia.org/wiki/Von_Neumann_architecture)  
[https://en.wikipedia.org/wiki/Modified\\_Harvard\\_architecture](https://en.wikipedia.org/wiki/Modified_Harvard_architecture)
- c) el registro de direcciones de memoria es un registro de propósito general que puede contener tanto direcciones como datos  
MAR no está disponible al programador, lo usa la UC para indicar la dirección de memoria a leer/escribir
- > d) el registro de estado (flags) es un registro de propósito específico cuyo contenido puede ser visto directa o indirectamente por el usuario mediante el uso de ciertas instrucciones específicas  
aunque no hemos visto pushf y popf, por eliminación ésta es la respuesta

17. Un bus se compone de:

- a) líneas de alimentación
- b) líneas de datos y líneas de dirección
- > c) líneas de control/estado, líneas de dirección y líneas de datos
- d) líneas de estado y líneas de control

18. ¿Cuál es el contenido de la pila al terminar de ejecutarse la siguiente secuencia de instrucciones de una arquitectura de pila?:

push #4

push #7

push #8

add

push #10

sub

mul

a) 4, 7, 48

b) 4

c) 4, 7, 8, 10

-> d) 20

19. La primera generación de computadores se caracteriza por el uso de:

a) Microprocesadores.

b) Transistores.

c) Fibra óptica.

-> d) Tubos de vacío.

20. ¿Cuál de las siguientes afirmaciones sobre el direccionamiento absoluto es falsa?

a) El objeto está en una posición de la memoria

b) La instrucción contiene la dirección de memoria en la que se encuentra el objeto

-> c) El tamaño del operando direccionado queda limitado por el nº de bits del campo de  
direcciónamiento

d) El rango de posiciones direccionables queda limitado por el tamaño del campo de  
direcciónamiento

No hemos explicado detalladamente que absoluto = directo con dirección completa. Evitar  
esta pregunta

21. Son funciones de la unidad de control:

a) la codificación de las instrucciones máquina

b) la lectura de memoria principal de la instrucción apuntada por el µPC

-> c) el secuenciamiento de las instrucciones máquina

d) todas las respuestas son ciertas

22. ¿De qué depende el tamaño del contador de programa?

- a) Del ancho del bus de datos.
- > b) Del número de direcciones de memoria.
- c) Del número de instrucciones diferentes y de los tipos de direccionamiento posibles.
- d) De la longitud del código de operación de las instrucciones.

23. Si queremos almacenar la palabra de 16 bits 0x8965 en una memoria de bytes según "little-endian", quedará almacenada a partir de la posición 0x8600 como:

- a) M[0x8600]=0x89 y M[0x8601]=0x65
- > b) M[0x8600]=0x65 y M[0x8601]=0x89
- c) M[0x8600]=0x69 y M[0x8601]=0x85
- d) M[0x8600]=0x85 y M[0x8601]=0x69

24. Una máquina superescalar es aquella que:

- a) basa su funcionamiento en la segmentación software como forma de incrementar el paralelismo.
- b) las instrucciones tienen un campo por cada unidad funcional al realizarse varias operaciones por instrucción.
- > c) emite simultáneamente múltiples instrucciones por ciclo de reloj, por ejemplo, una entera y otra de coma flotante.
- d) ninguna respuesta de las anteriores es correcta.

25. Si queremos almacenar la palabra de 64bits 0x0000001f ffffffe0 en una memoria de bytes según la convención little-endian a partir de la posición 0x0804913c, quedará

- a) 0x00 en 0x0804913c y 0xe0 en 0x08049143
- b) 0x1f en 0x0804913c y 0xe0 en 0x08049140
- > c) 0xe0 en 0x0804913c y 0x1f en 0x08049140
- d) Todas las respuestas anteriores son incorrectas

26. ¿En qué generación, dentro de la historia de los computadores digitales, aparecen los sistemas operativos multiusuario?

- a) primera
- b) segunda
- > c) tercera
- d) cuarta

27. Si queremos almacenar la palabra de 16 bits 0x9660 en una memoria de bytes según

"little-endian", quedará almacenada a partir de la posición 0x1000 como:

- a) M[0x1000]=0x96 y M[0x1001]=0x60
- b) M[0x1000]=0x69 y M[0x1001]=0x06
- > c) M[0x1000]=0x60 y M[0x1001]=0x96
- d) M[0x1000]=0x06 y M[0x1001]=0x69

28. Para obtener una única velocidad comparativa final, el benchmark SPEC CPU combina las velocidades de ejecución de una serie de tests, respecto a un ordenador de referencia, usando la media...

- a) armónica
- b) aritmética
- c) ponderada
- > d) geométrica

29. Un procesador con E/S mapeada a memoria tiene un bus de direcciones de 10 líneas y uno de datos de 8. El mapa de memoria tiene 512 posiciones para código (ROM), 256 para datos (RAM) y 256 para E/S, en ese orden. Los rangos de direcciones para esas tres zonas serán:

- a) 000 a 7FF, 800 a BFF y C00 a FFF
- > b) 000 a 1FF, 200 a 2FF y 300 a 3FF
- c) 000 a 9FF, A00 a CFF y D00 a FFF
- d) 000 a 5FF, 600 a 7FF y 800 a 9FF

30. ¿Cuál de las siguientes direcciones NO está alineada a double (8-byte)?

(Al no poder escribir el 2 como subíndice, aclaramos que ")2" indica binario)

- > a) 1110110101110100)2
- b) 1110110101101000)2
- c) 1110110101110000)2
- d) Todas están alineadas a doublé

31. En una arquitectura RISC típica:

- > a) suele usarse segmentación
- b) la programación resulta mucho más simple que en una arquitectura CISC
- c) se usan pocas instrucciones de las disponibles en el conjunto de instrucciones
- d) la UC es más compleja que en una arquitectura CISC

32. ¿Qué parámetro es más importante para comparar la velocidad de dos ordenadores diferentes?

- a) La frecuencia de reloj del procesador.
- b) La arquitectura del procesador.
- > c) El resultado de la ejecución de un conjunto de programas de prueba.
- d) El número medio de ciclos de reloj por instrucción.

33. ¿Por qué se impusieron las arquitecturas de registros de propósito general a las arquitecturas basadas en pila?

- a) Porque las basadas en registros permiten reducir el tamaño del programa
- > b) Porque las basadas en registros son capaces de lograr un mejor rendimiento cuando se asignan variables a registros
- c) Porque no se puede programar una arquitectura de pila en un lenguaje de alto nivel
- d) Porque la memoria es más cara que los registros

34. Para direccionar una memoria de bytes en la que quepan 2G palabras de 32 bits se necesitarán:

- a) 21 bits como máximo
- b) 32 bits exactamente
- > c) 33 bits como mínimo
- d) 31 bits como mínimo

35. ¿Cuál de los siguientes no es un tipo de bus?

- > a) Secuencial  
opuestos a buses paralelos son los buses serie  
opuestos a programas secuenciales son los programas paralelos
- b) Sistema
- c) E/S
- d) Paralelo

36. En una CPU de 32 bits con memoria de bytes, el problema es que...

- a) No tiene sentido, un registro no cabría en memoria
- b) Hay que usar 4 instrucciones de lectura (o escritura) para leer (o escribir) un registro completo
- > c) Hay que respetar el ordenamiento de bytes y reglas de alineamiento con que se diseñó la

CPU

d) No hay problema, cuando se salva un registro a memoria se escribe en la posición deseada

37. ¿Cuál es la característica tecnológica principal de la tercera generación de computadores?

a) Las válvulas

b) Los transistores

c) La gran integración de los circuitos (VLSI)

-> d) Los circuitos integrados

38. ¿Cuál es el valor mínimo (más negativo) que puede tomar un entero de 32bits en complemento a dos?

( $\wedge$  simboliza potenciación)

a)  $-2^{31} + 1$

b)  $-2^{32} + 1$

-> c)  $-2^{31}$

Present. tr.5,12, Intro. tr.8

d)  $-2^{32}$

39. En las instrucciones aritméticas con dos operandos de un procesador con arquitectura de pila, los dos operandos...

a) son dos registros del procesador.

b) pueden estar en cualquier posición de la pila.

c) se introducen en la pila tras realizar la operación.

-> d) son la cima de la pila y el elemento siguiente de la cima de la pila.

40. El direccionamiento directo a memoria utiliza...

a) dos desplazamientos contenidos en la propia instrucción.

b) un registro y un desplazamiento contenidos en la propia instrucción.

-> c) un desplazamiento.

d) un registro.

41. En una máquina little-endian con memoria de bytes y representación en complemento a dos que permite accesos a memoria de tamaño byte (1 B), media palabra (2 B) y palabra (4 B), se almacenan a partir de la posición 0xCAFEBA0 cuatro palabras con valores -1, -2, -3, -4.

¿Qué se obtendría al consultar la media palabra de la posición 0xCAFEBABE?

-> a) -1

los contenidos son

CAFEBAB0: FF FF FF FF

CAFEBAB4: FE FF FF FF

CAFEBAB8: FD FF FF FF

CAFEBABC: FC FF FF FF

las últimas dos posiciones, a partir de CAFEBABC, contienen FF FF, que es -1

b) -4

c) no se puede saber, faltan datos

d) ninguna de las anteriores

42. ¿Cuál de las siguientes afirmaciones sobre el benchmark SPEC CPU es falsa?

-> a) El resultado final es la media aritmética de las (12 ó 17) velocidades, bien sea de enteros ó de punto flotante (SPECint2006 ó SPECfp2006)  
es la media geométrica

b) Se cronometran unos 12 tests de enteros (CINT2006) y unos 17 tests de punto flotante  
(CFP2006)

c) La última versión es SPEC CPU2006 V1.2 de 2011

d) Se usa como referencia un computador UltraSPARC II 300MHz, y para cada test se calcula el cociente entre el tiempo de ejecución en el computador a testear y en el de referencia

43. Se pretende almacenar una palabra de 4 B en una memoria de bytes a partir de una dirección determinada. ¿Cuál de las siguientes es válida, si la palabra debe quedar alineada?

-> a) 0xFACEB00C

b) 0xCAFEBABE

c) 0xABADFOOD

d) 0xDEADBEEF

44. ¿En qué generación, dentro de la historia de los computadores digitales, aparece la memoria virtual?

a) primera

b) segunda

-> c) tercera

d) cuarta

45. ¿En qué pareja de registros están el dato/instrucción que se leerá o escribirá en memoria, y la dirección de memoria?

- > a) MBR y MAR
- b) MAR y ACUMULADOR
- c) MBR y PC
- d) IR y ACUMULADOR

46. La idea de desarrollar máquinas CISC surgió para:

- a) simplificar el diseño hardware de la UC.
- > b) tener instrucciones cercanas al lenguaje de alto nivel.
- c) conseguir un conjunto de instrucciones cortas y sencillas de decodificar.
- d) ninguna de las respuestas anteriores es cierta.

47. Un modo de vídeo de 512 x 256 píxeles y 16 colores por píxel ocupa una memoria de:

- a) 2 MB
- > b) 64 KB
- c) 128 KB
- d) 8 KB

48. ¿Cuál de las siguientes direcciones está alineada a double (8-byte)?

(Al no poder escribir el 2 como subíndice, aclaramos que ")2" indica binario)

- a) 1110110101110111)2
- b) 1110110101110100)2
- > c) 1110110101110000)2
- d) Ninguna de ellas

49. ¿Cuál es la característica tecnológica principal de la segunda generación de computadores?

- a) Los circuitos integrados
- b) Las válvulas
- c) La gran integración de los circuitos (VLSI)
- > d) Los transistores

50. ¿Cómo se almacenaría como palabra de 32 bits el número -128 en un sistema que utilice el criterio del extremo menor ("little endian")?

- a) posición 0: FF pos.1:FF pos.2: FF pos.3: 80

-> b) 0:80 1:FF 2:FF 3:FF

Present. tr.5,12, Intro. tr.8

c) 0:00 1:01 2: 00 3:80

d) Ninguna de las anteriores

51. En una estructura de computador de bus único (bus del sistema):

a) la UC concede el acceso al bus, por lo que éste funciona a la velocidad de la CPU

b) es la estructura más usada en los PC actuales

c) es necesario el arbitraje entre los maestros potenciales, no es suficiente la técnica de robo de ciclo ni otras similares.

-> d) sólo una unidad funcional puede tener el control del bus en cada momento

52. ¿Cuál de las siguientes afirmaciones es incorrecta?

a) En las arquitecturas CISC hay más instrucciones que en las RISC.

-> b) El tamaño de una instrucción en lenguaje máquina siempre ocupa dos bytes en los procesadores RISC.

c) Las arquitecturas RISC son del tipo registro-registro.

d) Las arquitecturas RISC simplifican la decodificación.

53. ¿Cuál de las siguientes expresiones representa un direccionamiento inmediato?

a) %eax

-> b) \$0x400

c) 8(%ebp)

d) (%eax)

54. Si N es el número de instrucciones máquina de un programa, F es la frecuencia de reloj, y

C el número promedio de ciclos por instrucción, el tiempo de ejecución del programa será:

a) N·F/C

b) N·F·C

c) N/(F·C)

-> d) N·C/F

55. El programador de lenguaje ensamblador necesita conocer:

a) la microarquitectura del procesador.

-> b) la arquitectura del ordenador.

Intro. tr.4

c) el diseño RTL del procesador.

d) todas las anteriores son ciertas.

56. El primer computador electrónico basaba su funcionamiento en:

-> a) tubos de vacío

1<sup>a</sup> generación

b) circuitos integrados LSI

3<sup>a</sup>-4<sup>a</sup> generación

c) núcleos de ferrita

2<sup>a</sup>-3<sup>a</sup> generación, tecnología RAM, no tecnología de conmutación

d) amplificadores operacionales

computadores analógicos

57. En una arquitectura de acumulador, la instrucción LOAD X:

a) transfiere el contenido del registro X a la memoria

no existe registro X en una máquina acumulador pura

el argumento de LOAD es una posición de memoria X

el 6502/6510 sí tenía, además de acumulador A, índices X e Y, y correspondientemente tenía instrucciones LDA, LDX, LDY

b) suma M(X) al acumulador

sería ADD X

-> c) transfiere el contenido de la posición de memoria X al acumulador

d) transfiere el contenido del acumulador a la posición de memoria X

sería STORE X

58. Un computador con 8 bits en el bus de direcciones puede direccionar como máximo:

a) 16384 palabras

-> b) 256 palabras

c) 8192 palabras

d) 1024 palabras

59. ¿En qué generación, dentro de la historia de los computadores digitales, aparecieron la microprogramación, la segmentación de cauce, la memoria cache, los S.O. multiusuario y la memoria virtual?

a) 2<sup>a</sup> generación (1955-65)

-> b) 3<sup>a</sup> generación (1965-75) //en la 3<sup>o</sup> generación se inventó casi todo.

c) 4<sup>a</sup> generación (1975-85)

d) esas innovaciones se repartieron a lo largo de varias generaciones, no sólo una

60. En una máquina little-endian con memoria de bytes y representación en complemento a dos que permite accesos a memoria de tamaño byte (1B), media palabra (2B) y palabra (4B), si se almacena en la posición 0xBABC una palabra de valor -2, ¿qué se obtendría al consultar la media palabra en la posición 0xBABE?

a) 0

b) 1

c) -2

-> d) -1

61. ¿En qué registro está contenido el último dato (o instrucción) leído de memoria, o el dato que se va a escribir en memoria?

a) PC.

-> b) MBR.

c) MAR.

d) Acumulador.

62. El bus del sistema es

a) en un sistema con bus único, todo el bus salvo la parte relacionada con E/S (SATA, GPU, USB, Ethernet, etc)

b) el que conecta las distintas partes del sistema: UC, ALU, E/S, M

-> c) el que conecta CPU-M, ya sea un sistema con bus único o con múltiples buses

d) en un sistema con buses separados, el que conecta el sistema E/S con el resto

63. En un sistema con dos buses separados, uno para el subsistema de memoria y otro para la E/S...

-> a) el bus que une la memoria y el procesador suele funcionar a la velocidad de la memoria

b) el bus de E/S funciona a la velocidad del periférico más rápido

c) ambos buses tienen que tener el mismo ancho de banda

d) Ninguna de las respuesta anteriores es cierta

64. El objetivo de un diseño CISC es...

a) disminuir la frecuencia de reloj.

- b) disminuir el número medio de ciclos por instrucción.
  - c) disminuir el tamaño medio de instrucción.
- > d) disminuir el número de instrucciones a ejecutar por un programa.

65. En un procesador de la familia 80x86 una variable de 32 bits, entera con signo, almacenada a partir de la dirección n contiene: 0xFF en la dirección n, 0xFF en la dirección n+1, 0xFF en la dirección n+2 y 0xF0 en la dirección n+3. ¿Cuánto vale dicha variable?

- a) 4294967280
  - b) 16
  - c) -16
- > d) -251658241

Como es little-endian, se trata de un número negativo de gran magnitud, y éste es el único con ese aspecto

66. ¿Cuál de las siguientes afirmaciones es incorrecta?

- a) El formato de una instrucción nos indica el significado de cada bit de la instrucción
  - b) Todas las instrucciones deben tener código de operación
  - c) No siempre es necesario indicar la dirección de la siguiente instrucción
- > d) Todas las instrucciones deben tener operando fuente y operando destino

67. ¿Cuál de las siguientes afirmaciones es incorrecta?

- a) El direccionamiento indexado es útil para manejo de vectores
- > b) El direccionamiento indirecto indica la dirección del operando
- c) En el direccionamiento inmediato el dato se encuentra en la propia instrucción
- d) En el direccionamiento implícito no se indica la ubicación del operando

No hemos explicado detalladamente el implícito, evitar esta pregunta

68. Sea un computador con 48 registros y 200 instrucciones máquina. ¿Cuántas direcciones de memoria permite el formato de la instrucción de 32 bits hipotética "beqz r1, r2, dir"?

(En este enunciado el símbolo  $\wedge$  representa potenciación)

- > a)  $2^{12}$  //En la antigua ECI se hacían estos cálculos, códigos Hamming, etc. En la nueva EC no se ha explicado con tanto detalle. Evitar esta pregunta.
- b)  $2^{14}$
- c)  $2^{18}$
- d)  $2^{16}$

69. En las arquitecturas RISC hay...

- > a) muchos registros y pocos modos de direccionamiento.
- b) pocos registros y muchos tipos de instrucciones.
- c) pocos modos de direccionamiento y muchos formatos de instrucción.
- d) pocas instrucciones muy rápidas con muchos modos de direccionamiento.

70. En el contexto del lenguaje máquina, el acrónimo ISA suele referirse a:

- a) Intel Standard Architecture
- b) Industry Standard Architecture
- > c) Instruction Set Architecture
- d) Information Security Architecture

71. El espacio direccionable de memoria de un computador depende del diseño del:

- > a) Bus de direcciones
- b) Bus de datos
- c) a) y b) son correctas
- d) Ninguna de las anteriores es correcta

72. Respecto a los dispositivos activos y pasivos en un bus podemos decir que:

- a) Los dispositivos pasivos sólo pueden convertirse en esclavos
- b) Sólo los dispositivos activos pueden convertirse en maestros
- > c) Las respuestas a y b son ciertas
- d) Las respuestas a y b son falsas

73. En una memoria de bytes que contuviera a partir de la posición 0 los valores

1,0,0,0,0xFE,0xFF,0xFF,0xFF, se puede decir que...

- a) Hay una palabra de 16bit big-endian con valor 1 en la posición 0
- b) Hay una palabra de 16bit little-endian con valor 254 en la posición 3
- c) Hay una palabra de 32bit little-endian con valor -1 en la posición 4
- > d) Todas las respuestas anteriores son incorrectas

74. En la ejecución de una instrucción...

- a) la UC activa las señales de control que envía por el bus de direcciones
- b) siempre se altera el registro de estado
- > c) la ALU realiza las operaciones aritméticas y lógicas
- d) el registro de instrucción se va incrementando para apuntar a la siguiente instrucción

75. ¿En qué generación, dentro de la historia de los computadores digitales, aparece la microprogramación?

- a) primera
- b) segunda
- > c) tercera
- d) cuarta

76. ¿En qué generación, dentro de la historia de los computadores digitales, aparece la segmentación de cauce?

- a) primera
- b) segunda
- > c) tercera
- d) cuarta

77. La ecuación básica de rendimiento calcula

- > a) cuánto tiempo tarda en ejecutarse un programa concreto conociendo su número de instrucciones y el número de etapas (promedio) y la frecuencia del procesador
- b) el promedio de las ganancias obtenidas con una serie de programas de punto entero
- c) la media geométrica de los cocientes entre los tiempos de ejecución de una serie de programas predeterminados
- d) cómo de mejor es un procesador frente a otro, conociendo las prestaciones de las respectivas UC, ALU, E/S y M

78. Si usamos una estructura de bus con DMA:

- > a) la CPU puede dejar las transferencias entre MP y periféricos en manos de este controlador (DMA) y seguir ejecutando otras instrucciones.
- b) al bus del sistema sólo se conecta la CPU y la MP, ya que el DMA se conecta directamente a MP para realizar las transferencias de datos.
- c) la velocidad de este controlador establece la velocidad del bus del sistema.
- d) podemos prescindir de controladores de E/S ya que el controlador de DMA se ocupa de controlar las transferencias hacia/desde los periféricos.

79. ¿Cuál es el contenido de una pila al terminar de ejecutarse la siguiente secuencia de operaciones push y pop?:

push #1

push #2

push #3 ; pop a

push #4

pop a

pop a

-> a) 1

b) 1 y 2

c) 1, 2, 3 y 4

d) 10

80. El registro MBR...

a) especifica la dirección en memoria de la palabra que va a ser escrita o leída

-> b) contiene el valor que va a ser almacenado en la memoria, o bien se usa para recibir un valor procedente de la memoria

c) contiene la dirección de la próxima instrucción que va a ser captada de memoria

d) contiene el código de operación de la instrucción que se está ejecutando

81. En una arquitectura de registros de propósito general (a nivel de lenguaje máquina):

a) operar usando registros es más rápido.

b) la generación de código resulta más simple que en arquitecturas de pila o acumulador.

c) se evita el cuello de botella (por ejemplo, pila, o acumulador) que otras arquitecturas presentan al evaluar expresiones aritméticas complejas

-> d) todas las respuestas anteriores son ciertas.

82. En la arquitectura Von Neumann...

a) los bloques principales son la unidad de control, la ALU y la CPU.

-> b) el programa se encuentra residente en memoria.

c) los registros se encuentran en la memoria principal.

d) Todas son ciertas.

83. Si un computador X ejecuta un programa de 450 millones de instrucciones en 26

segundos y un computador Y tarda 14 segundos en ejecutar ese mismo programa, ¿cuántas

veces es más rápido el computador Y que el X?

a) 1,538

-> b) 1,857

c) 0,538

d) 12

84. Una CPU con bus de direcciones de 16 bits y bus de datos de 8 bits tiene un registro de 8 bits conectado al bus de datos y a la unidad de control. Puede tratarse del registro

a) Contador de programa

b) De direcciones

c) Puntero de pila

-> d) De instrucción

85. No en todas las instrucciones máquina hay una fase de

a) decodificación

-> b) captura de operandos

c) captación

d) ejecución

86. El conjunto de todos los atributos de un sistema que son visibles para el programador y son necesarios para programar en lenguaje máquina se denomina:

a) repertorio de instrucciones máquina

-> b) arquitectura del computador

Intro. tr.4

c) conjunto de componentes físicos del computador

d) organización del computador

87. Si en un bus de direcciones de 32 bits se decodifica parcialmente la dirección de un dispositivo de 32 posiciones usando 22 bits, ¿cuántas veces aparecerá repetido en el mapa de memoria?

a) 10

b) 1024

-> c) 32

el dispositivo tiene 32 puertos, usa 5 bits de direccionamiento (probablemente los 5 LSB), pero en lugar de decodificarse con 27, se usan sólo 22 bits (probablemente los 22 MSB) de manera que quedan 5 bits (probablemente intermedios) sin usar, que pueden tomar 32 combinaciones posibles

d) 16

88. ¿Cuál de las siguientes afirmaciones es falsa?

- a) el bus de control transporta señales de estado
- b) el bus de datos es bidireccional
- c) el bus de direcciones es unidireccional
- > d) la anchura del bus de datos es de 16 bits

89. Si queremos almacenar la palabra de 16 bits 0x8965 en una memoria de bytes según

"big-endian", quedará almacenada a partir de la posición 0x1000 como:

- a) M[0x1000]=0xA6 y M[0x1001]=0x91
- > b) M[0x1000]=0x89 y M[0x1001]=0x65
- c) M[0x1000]=0x91 y M[0x1001]=0xA6
- d) M[0x1000]=0x65 y M[0x1001]=0x89

90. ¿Cuál de los siguientes registros se utiliza para guardar la dirección de memoria donde se localiza la instrucción siguiente?

- a) Memory Data Register
- > b) Program Counter
- c) Memory Address Register
- d) Instruction Register

91. Para direccionar una memoria de bytes en la que quepan 1G palabras de 32 bits se necesitarán:

- a) 31 bits
- b) 33 bits
- > c) 32 bits
- d) 21 bits

92. Según la clasificación m/n, las máquinas de acumulador son de tipo

- a) 0/0
- > b) 1/1
- c) 2/2 ó 2/3
- d) 1/2

93. En la captación de un operando que reside en memoria:

- a) en MBR indicamos la dirección donde está y en MAR lo recogemos
- b) en MBR indicamos la dirección donde está y en IR lo recogemos

- c) en MAR indicamos la dirección donde está y en IR lo recogemos
- > d) en MAR indicamos la dirección donde está y en MBR lo recogemos

94. En el arbitraje de un bus...

- a) los dispositivos pasivos pueden requerir el uso del bus para iniciar una transferencia
- > b) si hay un único dispositivo pasivo, siempre funciona como esclavo
- c) si hay varios dispositivos activos, siempre funcionan como maestros
- d) todas las respuestas anteriores son ciertas

95. ¿Cuál es el contenido de la pila al terminar de ejecutarse la siguiente secuencia de instrucciones de una arquitectura de pila?:

push #4

push #7

add

push #10

sub

-> a) 1

b) 11, 1

c) 4, 11, 1

d) 4, 7, 10

96. Se dice que las máquinas con arquitectura Von Neumann siguen un modelo de programa...

a) Cableado.

b) Micropogramado.

c) Externo.

-> d) Almacenado.

97. Un computador con 13 líneas de direcciones utiliza E/S mapeada a memoria. Si se supone que cada uno de los periféricos ocupa 4 direcciones y que el número de periféricos que se planea conectar es de  $2^{10}$ , ¿qué tamaño de memoria admite el computador?

a)  $2^{10}$  palabras

-> b)  $2^{12}$  palabras

c)  $2^{13}$  palabras

d) Ninguna de las anteriores

98. ¿De qué tipo son los procesadores Intel que usamos en los laboratorios?

- a) puede ajustarse mediante un bit de control en el registro CR0 que funcionen como little- o big-endian
- b) el concepto de endian no es aplicable a estas máquinas, ya que un registro del procesador no cabe en una posición de memoria
- c) big-endian
- > d) little-endian

99. ¿De qué depende el tamaño del contador de programa?

- a) de la longitud del código de operación
- b) del ancho del bus de datos
- c) el tamaño no importa
- > d) ninguna de las anteriores es cierta

100. El instrumento GIADA de la sonda espacial ROSETTA (diseñado en Granada) está basado en un microprocesador 8086 y el siguiente mapa de memoria:

RAM volátil: 00000 - 0FFFF

RAM no volátil: 10000 - 1FFFF

ROM: F0000 - FFFFF

¿Cuál es el tamaño total de la memoria?

- a) 3MB
- b) 48KB
- c) 2MB
- > d) 192KB

101. Un computador que utilice el sistema big-endian, almacena el número 0x2143 a partir de la dirección 0 como:

- a) M[0]=0x43 y M[1]=0x21
- > b) M[0]=0x21 y M[1]=0x43
- c) M[0]=0x12 y M[1]=0x34
- d) M[0]=0x34 y M[1]=0x12

102. El direccionamiento relativo a registro base utiliza...

- a) dos desplazamientos contenidos en la propia instrucción.
- b) un registro y un factor de escala

c) dos registros.

-> d) un registro y un desplazamiento

103. El computador EDVAC, propuesto por John Von Neumann, presentaba dos importantes diferencias respecto al ENIAC:

- > a) Empleaba aritmética binaria y permitía trabajar con un programa almacenado.
- b) Se programaba enchufando centenares de clavijas y empleaba aritmética octal.
- c) Era electromecánico y de propósito específico.
- d) Utilizaba transistores y memoria de semiconductor.

104. ¿Cuál de las siguientes características es posterior a la segunda generación de computadores?

- a) Lenguaje ensamblador.
- b) Transistor.
- > c) RISC.
- d) Memoria de núcleos de ferrita.

105. ¿Qué arquitectura se caracteriza por presentar una gran variación en la longitud de las instrucciones?

- a) registro-registro
- b) registro-memoria
- > c) memoria-memoria
- d) ninguna de las anteriores es cierta

106. En las últimas generaciones de computadores la mejora de prestaciones viene dada por:

- a) avances en las tecnologías de fabricación.
- > b) avances en tecnología y avances en la estructura y arquitectura del computador.
- c) avances en los sistemas operativos y aplicaciones.
- d) avances en la estructura y arquitectura del computador.

107. Una dirección de memoria se refiere siempre a:

- a) una palabra
- b) 16 bits
- c) un byte
- > d) ninguna de las anteriores

108. Según la clasificación m/n, las máquinas con arquitectura R/R son de tipo

a)  $x/x$  con  $x=2,3$

b)  $x/0$  con  $x=2,3$

-> c)  $0/x$  con  $x=2,3$

d)  $0/0$

109. En un sistema con un único bus...

a) sólo un dispositivo puede escribir en un instante dado en el bus

b) se utilizan las mismas líneas de control para conectar todos los dispositivos

c) el procesador y los periféricos pueden funcionar a diferentes velocidades si el funcionamiento del bus es asíncrono

-> d) Todas las respuestas anteriores son ciertas

110. ¿Qué tipo de instrucciones se emplean más en una arquitectura de acumulador?

a) de desplazamiento y rotación

b) de transferencia de datos entre registros

-> c) de transferencia de datos con memoria

d) aritmético-lógicas

111. Un computador con 13 líneas de direcciones utiliza E/S mapeada a memoria. Si se supone que cada uno de los periféricos ocupa 4 direcciones y que el número de periféricos que se planea conectar es de  $2^{10}$ , ¿qué tamaño de memoria admite el computador?

a)  $2^{10}$  palabras

-> b)  $2^{12}$  palabras

c)  $2^{13}$  palabras

d) Ninguna de las anteriores

112. ¿Cómo se almacenaría como palabra de 32 bits el número -128 en un sistema que utilice el criterio del extremo menor ("little endian")?

a) posición 0: FF pos.1:FF pos.2: FF pos.3: 00

b) 0:00 1:FF 2:FF 3:FF

c) 0:00 1:01 2: 00 3:80

-> d) Ninguna de las anteriores

113. Una memoria que está estructurada en palabras de 8 bits tiene una capacidad de 64 Kbits. ¿Cuántas líneas de dirección tiene dicha memoria?

a) 24

b) 12

-> c) 13

d) 8

114. El ancho de palabra de una memoria corresponde a:

a) La longitud del registro de direcciones de la memoria.

b) El número de posiciones que la componen.

c) El número que identifica únicamente cada posición de la memoria.

-> d) La longitud del registro de datos de la memoria.

115. ¿Cuál de las siguientes afirmaciones acerca de la arquitectura Von Neumann es cierta?

-> a) La separación entre almacenamiento y unidad de procesamiento es una de las ideas contempladas en la arquitectura Von Neumann.

b) La arquitectura Von Neumann es un diseño que sitúa el programa en un almacenamiento distinto al de los datos.

c) Existe un consenso general en considerar justo el término "arquitectura von Neumann", ya que las ideas de esta arquitectura fueron completamente originales de John Von Neumann y no influenciadas por sus colaboradores o predecesores.

d) Para ejecutar un programa en una máquina Von Neumann, es necesario volver a cablear o incluso rediseñar la máquina

116. En la memoria de un procesador de la familia Intel 64 (x86-64) se almacena a partir de la dirección N los siguientes contenidos: 0xff, 0xff, 0xff, 0xff, 0x00, 0x00, 0x00, 0x01, 0xff, 0xff, 0xff, 0x00, 0x00, 0x00, 0x00, 0x02. Posteriormente se lee (carga) %rax desde (a partir de) la dirección N. ¿Cómo es %rax entonces, considerándolo con signo?

a) menor que -2000 millones

b) entre -2000 millones y -1

c) entre 0 y 2000 millones

->d) mayor que 2000 millones

almacenado en little-endian es 0x01000000 ffffffff, positivo muchísimo mayor que 2G (2G-1 cabría en 32 bits)

117. ¿Cuál de las siguientes afirmaciones sobre el direccionamiento directo es falsa?

a) El objeto está en una posición de la memoria

sí, a partir de la indicada (si no cabe en sólo una)

b) El rango de posiciones direccionables queda limitado por el tamaño del campo de direccionamiento

sí, particularmente relevante en SysV-AMD64 con "32bit RIP-relative"

->c) El tamaño del operando direccionado queda limitado por el nº de bits del campo de direccionamiento

no, el operando puede ocupar (parte de) la posición indicada y sucesivas posiciones

d) La instrucción contiene la dirección de memoria en la que se encuentra el objeto  
sí, ésa es la definición de direccionamiento directo

118. El número -12 se almacenará en complemento a 2 en el registro %eax como:

->a) 0xFFFFFFFF4

b) 0xFFFFFFF0C

c) 0xFFF4

d) 0xFF0C

119. ¿En qué generación, dentro de la historia de los computadores digitales, aparece la memoria virtual?

a) primera

b) segunda

->c) tercera

d) cuarta

120. ¿Cuál de los siguientes elementos no forma parte de la Arquitectura del Repertorio de Instrucciones (ISA)?

a) Descripción de los tipos de datos sobre los que opera el lenguaje máquina.

b) Descripción de los registros de datos, registros de estado y control.

c) Descripción del espacio de direccionamiento de la memoria y de la E/S.

->d) Descripción de los campos de bits en los que están organizadas conceptualmente las microinstrucciones.

no, eso no hace falta para redactar un programa correcto, de hecho existen procesadores cuya UC no está micropogramada

121. ¿Qué tipo de direccionamiento se usa para el registro destino en la instrucción x86-64 add array(%rbx,4), %edx?

a) Direccionamiento inmediato

b) Direccionamiento relativo a registro base

c) Direccionamiento indexado

->d) Direccionamiento a registro

122. Justo antes de que una instrucción máquina escriba un resultado en memoria:

- a) en MAR está el resultado y en MBR la dirección donde se almacenará
- b) en IR está el resultado y en MBR la dirección donde se almacenará
- >c) en MBR está el resultado y en MAR la dirección donde se almacenará
- d) en IR está el resultado y en MAR la dirección donde se almacenará

123. ¿En qué registro está contenido el último dato (o instrucción) leído de memoria, o el dato que se va a escribir en memoria?

->a)

mbr/mdr

- b) registro de propósito general
- c) mar
- d) pc

124. Escoger de entre las 4 operaciones la de mayor valor que pueda calcularse con enteros de 4B con signo sin problemas

a)  $3.000.000.000 + 3.000.000.000$

seis mil millones no cabe. Ni siquiera se puede representar correctamente tres mil millones

b)  $300.000.000 + 300.000.000$

seiscientos millones... sin problema

c)  $100.000.000 + 100.000.000$

doscientos millones... sin problema

->d)  $1.000.000.000 + 1.000.000.000$

dos mil millones cabe (y un poco más también)

## TEMA 2

1. Una instrucción de "salto si menor", para números sin signo, tiene que comprobar el valor de:

- a) el bit de cero
- b) los bits de signo y desbordamiento
- > c) el bit de acarreo
- d) el bit de signo

2. ¿Qué valor contendrá el registro edx tras ejecutar las dos instrucciones siguientes?

```
movl $-1, %edx
```

```
movb $1, %dl
```

UsuarioProfesores

a) 00000000 00000000 00000000 00000001

b) 11111111 11111111 11111111 11111111

c) 00000001 00000000 00000000 00000000

-> d) 11111111 11111111 11111111 00000001

3. En los modos de direccionamiento del tipo Desplazamiento(Base,Indice,Factor Escala),

puede usarse como

- a) base, cualquiera de los 8 registros enteros salvo %esp
- b) factor de escala, cualquier constante de 1, 2, 4 u 8 bytes
- > c) desplazamiento, cualquier constante de 1, 2 o 4 bytes (incluso el nombre de una variable, por su dirección)
- d) índice, también cualquiera salvo %ebp

4. Respecto a requisitos de alineamiento de structs en gcc/IA32 x86 y x86-64, alguna de las siguientes afirmaciones es falsa

- a) en x86-64 Linux alinea double a 8x (Windows también)
- b) en x86 Linux alinea double a 4x (Windows no)
- > c) en x86-64 Linux alinea float a 8x (Windows también)

Transparencias corregidas, ver Tema 2.5 tr.6

- d) en x86 Linux alinea long double a 4x (Windows también)

5. La primera instrucción ensamblador de una subrutina compilada con gcc en Linux/x86  
cdecl suele ser:

- a) push %ebx
  - b) mov %esp, %ebp
  - c) pop %ebx
- > d) push %ebp

6. La instrucción JGE / JNL provoca un salto si...

- a) CF = 1  
sería jc/jb

-> b) OF = SF

basta recordar que "Less" no era un flag solo (es OF^SF)

recordar también que "Below" comprueba CF

- c) SF = 1  
sería js
- d) SF = 0  
sería jns

7. El lenguaje máquina es...

- > a) difícil de codificar manualmente.
- b) portable entre arquitecturas.
- c) una alternativa razonable al uso del lenguaje ensamblador.
- d) fácilmente legible por el programador.

8. La arquitectura x86-64 tiene:

- a) 8 registros de propósito general (RPG) de 64 bits (%rax, %rbx, ... %rsp, %rbp)
- b) 64 registros RPG de 64 bits
- > c) 16 registros RPG de 64 bits
- d) 32 registros RPG de 64 bits

9. El marco de pila en x86-64 Linux...

- a) se crea para funciones en las que GCC no puede evitar que RBP baje más, como por ejemplo: que haya demasiadas variables locales (y no quepan en registros), o que haya que salvar algún registro salva-invocado
- b) no existe, porque RBP no es registro especial en x86-64
- > c) se crea para funciones en las que GCC no puede evitar que RSP baje más, como por ejemplo: que haya que calcular la dirección de una variable local, o pasar más de 6 argumentos

a otra función

d) sólo se crea para funciones que invocan anidadamente a otra función (procedimientos padre, no hojas)

10. La instrucción leave equivale a:

- > a) movl %ebp, %esp; popl %ebp
- b) pushl %ebp; movl %esp, %ebp
- c) movl %esp, %ebp; popl %esp
- d) pushl %esp; movl %ebp, %esp

11. El comienzo de un procedimiento que siga la convención cdecl es:

- a) mov %esp,%ebp; push %ebp
- b) push %ebp; mov %ebp,%esp
- c) mov %ebp,%esp; push %ebp
- > d) push %ebp; mov %esp,%ebp

12. Las instrucciones que asignan la dirección 0x78e00 a un puntero situado en la posición 0xff00 son:

- a) mov 0x78e00,%eax  
lea %eax,0xff00
- b) mov \$0x78e00,%eax  
lea %eax,0xff00
- c) mov 0x78e00,%eax  
mov %eax,0xff00
- > d) mov \$0x78e00,%eax  
mov %eax,0xff00

13. Al traducir la sentencia C r->i = val; gcc genera el código ASM movl %edx, 12(%eax). Se deduce que

- > a) el desplazamiento de i en \*r es 12  
y en concreto r está en %eax
- b) val es un entero que vale 12  
val está en %edx
- c) r es un puntero que apunta a la posición de memoria 12  
r es puntero a struct y seguramente valdrá 0x0804XXXX en ejecutables x86

d) i es un entero que vale 12

después de la asignación, i vale lo que había en %edx (que es val)

14. Despues de ejecutar el siguiente código, ¿qué variables serán igual a 0? (Suponer ints de 32bits y longs de 64bits)

```
unsigned int a = 0xffffffff;
```

```
unsigned int b = 1;
```

```
unsigned int c = a + b;
```

```
unsigned long d = a + b;
```

```
unsigned long e = (unsigned long)a + b;
```

a) Ninguna

b) c

-> c) c y d

En el problema 3.4 sólo se explica que una extensión de tamaño se hace según el fuente sea signed (extensión sgn) o unsigned (ext. con ceros), pero no se explica la sección 2.2.6 (y 2.2.5) en donde se aclara que las operaciones que impliquen a algún unsigned se hacen en unsigned.

Evitar esta pregunta en el futuro.

d) c, d, y e

15. ¿Cuál de las siguientes expresiones toma el valor 0x01 si x es múltiplo de 32 y 0x0 en caso contrario? Asumir que x es unsigned int.

a)  $(x \& 0x1f)$

-> b)  $!(x \& 0x1f)$

c)  $(x | 0x3f)$

d)  $!(x \& 0x3f)$

16. ¿Cuál de las siguientes afirmaciones es correcta?

a) El lenguaje máquina es igual para todos los computadores.

-> b) Las instrucciones en lenguaje máquina se almacenan y tratan como cadenas de unos y ceros.

c) El lenguaje ensamblador es igual para todos los computadores.

d) Las instrucciones en lenguaje ensamblador se almacenan y tratan como cadenas de unos y ceros.

17. Al ejecutar el fragmento de código:

```
leal -48(%eax), %edx
```

```
cmpl $9, %edx
```

```
ja .L2
```

se salta a .L2 si el contenido del registro %eax:

- a) es mayor o igual que 57
- b) está dentro del intervalo [48,57]
- > c) está fuera del intervalo [48,57]
- d) es mayor o igual que 48

18. De entre las siguientes construcciones de flujo de control en lenguaje C, la que se traduce

más directamente a lenguaje ensamblador es...

- > a) El bucle do-while
- b) El bucle for
- c) La selección switch-case
- d) El bucle while

19. Respecto a registros de propósito general (RPG), el 80386 tiene:

- a) 8 registros de 8 bits
- > b) 8 registros de 32 bits

Tema1 tr.26

- c) 16 registros de 64 bits
- d) 16 registros de 16 bits

20. La instrucción seta %al (seta significa "set if above"):

- a) Pone AL a 1 si CF=1 y ZF=0
- > b) Pone AL a 1 si CF=0 y ZF=0
- c) Pone AL a 1 si CF=1 o ZF=1
- d) Pone AL a 1 si CF=0 o ZF=0

21. Se ha declarado en un programa C la variable int val[5]={1,5,2,1,3} . ¿Cuál de las tres primeras opciones (a, b, o c) es FALSA?

- > a) val[1] == 1
- b) &val[3] == val+3
- c) sizeof(val) == 20
- d) No se puede marcar ninguna de ellas, todas (a, b y c) son ciertas

22. Respecto a registros base e índice en IA32, la excepción es que

- a) EBP no puede ser registro índice
  - b) ESP no puede ser registro base
  - c) EBP no puede ser registro base
- > d) ESP no puede ser registro índice

23. ¿Cuál de las siguientes instrucciones máquina copia en EAX la dirección efectiva resultante de la operación EDX\*4 + EBX?

- a) movl 4(%edx, %edx), %eax
  - b) leal 4(%edx, %edx), %eax
  - c) movl (%ebx, %edx, 4), %eax
- > d) leal (%ebx, %edx, 4), %eax

24. Al traducir de lenguaje C a ensamblador, gcc en máquinas Linux/IA32 almacena (reserva espacio para) una variable “var” local a una función “fun” en una dirección de memoria referenciable (en lenguaje ensamblador) como:

- > a) -k(%ebp), siendo k un número constante positivo relativamente pequeño
- b) k(%esp), siendo k un número constante positivo relativamente pequeño
- c) var (el nombre de la variable representa su posición de memoria)
- d) k(%ebp), siendo k un número constante positivo relativamente pequeño

25. Por x86-64 se entiende la misma arquitectura de repertorio (ISA) que

- a) IA32
- > b) AMD64
- c) x86
- d) IA64

26. Cuál de las instrucciones máquina siguientes es incorrecta en x86-64:

- a) movl (%rdi,%rcx,4), %edx
  - b) testl %edx, %edx
  - c) addq \$1, %rcx
- > d) movl %r8, %eax

27. Usando el repertorio IA32, para intercambiar el valor de 2 variables (por ejemplo A: .int 1 y B: .int 2) se pueden usar...

- a) una instrucción mov y una instrucción lea

-> b) 4 mov, no menos (debido a la arquitectura R/M) //ver la función swap()

c) dos instrucciones mov

d) 3 mov, no menos (se le llama "intercambio circular")

si fuera arquitectura M/M se podría hacer intercambio circular:

mov A,R1

mov B,A; // esta instrucción no es de repertorio R/M

mov R1,B

29. Si EBX=0x00002a00 y EDI=0x0000000a, ¿cuál es la dirección efectiva en la instrucción add

0x64(%ebx,%edi,2),%eax?

a) 0x00005478

b) 0x000054dc

c) 0x00002a70

-> d) 0x00002a78

30. En Linux IA32, si gcc usa la instrucción leave se puede asegurar que en ese punto del programa

-> a) ya no hay registros salva-invocado que recuperar

tal vez porque no había ninguno, para empezar

b) ya no se hacen llamadas anidadas y por tanto no hay parámetros que ocupen espacio en pila

puede usarse leave habiendo espacio reservado en pila para argumentos de llamadas anidadas, siempre que no haya que recuperar registros salva-invocado

c) correspondería emitir la secuencia de salida pop/ret, pero leave hace lo mismo y ocupa menos espacio

leave no equivale a ret, y sólo hace pop %ebp

d) ya no hay variables locales que destruir

puede usarse leave para destruir rápidamente todas las variables locales, siempre que no haya que recuperar registros salva-invocado

31. ¿Cuál es el efecto de la instrucción siguiente?

mov 8(%ebp),%ecx

-> a) Suma 8 al contenido de ebp, trata la suma como una dirección de memoria y almacena el contenido de esa dirección en ecx

- b) Suma los contenidos de ebp y de la dirección de memoria 8 y almacena la suma en ecx
- c) Suma 8 al contenido de ebp y almacena la suma en ecx
- d) Suma 8 al contenido de la posición de memoria cuya dirección está almacenada en ebp, y almacena la suma en ecx

32. En el direccionamiento indirecto a través de registro, la dirección efectiva...

- a) se encuentra en el registro de instrucción.
- b) se calcula como la suma del contenido de dos registros.
- c) se encuentra en una dirección de memoria.
- > d) se encuentra en un registro general del procesador.

33. Respecto a registros salva-invocante y salva-invocado en GCC/Linux IA32, ¿cuál de éstos es de distinto tipo que el resto?

- > a) EBX
- b) EAX
- c) EDX
- d) ECX

34. Las instrucciones JB y JNAE del Pentium provocan un salto si...

- a) SF == 1
- b) ZF != SF
- > c) CF == 1

below es la condición más fácil: sin signo 3-4 = debo una

- d) ZF == 0

35. ¿En qué registro se pasa el primer argumento a una función en Linux gcc x86-64?

- a) edx
- > b) edi
- c) ecx
- d) esi

36. La convención de llamada Linux/GCC x86-32 considera, respecto a convenios de uso de registros:

- a) 8 registros salva-invocante, 6 registros salva-invocado, y 2 especiales
- b) Algunos registros para pasar argumentos, otros salva-invocante, otros salva-invocado, dos especiales

-> c) 3 registros salva-invocante, 3 registros salva-invocado, y 2 especiales

d) Algunos registros salva-invocante, otros salva-invocado, uno especial

37. Con el repertorio IA32, para sumar %eax y %ebx dejando el resultado en %ecx se podría hacer lo siguiente:

-> a) lea (%eax, %ebx, 1), %ecx

b) lea %ecx, %ebx, %eax

c) lea %eax, %ebx, %ecx

d) lea %ecx, [%eax, %ebx]

38. Suponiendo que todos los registros inicialmente contienen el valor 1, ¿cuál es el valor de ECX tras la ejecución de la siguiente secuencia de instrucciones?

mov \$4, %eax

mov \$3, %ebx

lea (%eax, %eax), %ecx

sub %ebx, %ecx

imul %ecx, %eax

a) 6

b) 36

-> c) 5

d) 20

39. Diferencias gcc Linux IA32/x86-64: marcar la respuesta falsa

-> a) el tipo double pasa de 4 B a 8 B

siempre 8 B

b) los enteros largos (long) pasan de 32 a 64 bits

c) long double pasa de 10/12 B a 16 B

d) los punteros (void\*) pasan de 32 a 64 bits

40. Si AX = 0xFA50 y ejecutamos XOR \$0xFF, AX

a) El registro AL se pone a 0.

b) El registro AH se pone a 0.

-> c) Se realiza el complemento a 1 de AL.

d) Se realiza el complemento a 1 de AH.

41. La instrucción movzbl %al, %eax

-> a) Copia en %eax el valor sin signo almacenado en %al, rellenando con ceros

Tema 2.1 tr. 28, Hallaron, p.205-206

b) Copia en %eax el valor de %al si el indicador de cero está activado

c) Copia en %eax el valor del indicador de cero

d) Pone a 0 el registro %eax

42. En IA32, ¿cuál de los siguientes fragmentos de programa tiene un efecto sobre los flags

distinto al resto?

-> a) mov \$-1, %edi

mov no afecta a los flags

b) mov \$0, %edi

sub \$1, %edi

sub afecta

c) sub %edi, %edi

adc \$0xFFFFFFFF, %edi

adc afecta

d) mov \$-1, %edi

add \$0, %edi

add afecta

43. Si la estructura struct a ocupa un espacio de 28 bytes en memoria, ¿cuántos bytes ocupa

la siguiente estructura struct b cuando se compila en 64 bits?

```
struct b {
```

```
    struct a a1;
```

```
    int i;
```

```
    struct a a2;
```

```
};
```

-> a) 60 bytes //28 + 4 + 28 = 60

b) 84 bytes

c) 24 byte // imposible: una sola struct a ya ocupa más

d) 64 bytes

44. ¿Cuál de las siguientes afirmaciones es falsa?

a) Las subrutinas necesitan ser capaces de reservar espacio en memoria para las variables

locales sin sobrescribir ningún dato usado por el programa que hace la llamada

b) Las subrutinas necesitan algún modo de saber desde dónde han sido llamadas para poder volver al programa que realizó la llamada cuando se completa la subrutina

c) Los programas necesitan una forma de pasar parámetros a las subrutinas y de recibir las salidas de vuelta

-> d) Las subrutinas necesitan recibir parámetros desde el programa que hace la llamada que indiquen qué registros pueden alterar y cuáles no

45. ¿Cuál de las siguientes afirmaciones es cierta?

a) En el direccionamiento inmediato, la dirección efectiva se encuentra en un registro de uso general del procesador.

-> b) El modo de direccionamiento indirecto a través de registro es el modo más sencillo de direccionamiento a memoria.

c) Los modos de direccionamiento de una instrucción se especifican en ensamblador mediante las directivas.

d) En el direccionamiento a registro, el objeto direccionado es una constante contenida en la propia instrucción.

46. En una resta de dos números en complemento a dos, se produce desbordamiento cuando...

a) los dos operandos son negativos y el resultado es positivo.

b) los dos operandos son positivos y el resultado es negativo.

c) Las dos respuestas a y b son correctas.

-> d) Ninguna de las anteriores es correcta.

47. El primer microprocesador de 32 bits de la familia x86 fue el:

a) 80486

b) 80286

c) Pentium

-> d) 80386

48. ¿Cuál es la diferencia entre los desplazamientos a la derecha lógico y aritmético?

a) El aritmético inserta en el bit más a la derecha una copia del bit de signo

b) Ninguna, la diferencia es entre los desplazamientos a la izquierda

c) El lógico inserta siempre ceros en el bit más a la derecha

-> d) Insertan de forma distinta el bit más a la izquierda

49. ¿Cuál de las siguientes listas menciona registros x86-64 del mismo tipo respecto a convenio de uso? (salva-invocante, invocado, etc)

a) RAX, RBX, RCX, RDX //RBX salva-invocante

-> b) CL, DX, R8d, R9 //todos "amarillos" (args. salva-invocante) en tr.2.4.5

c) RBX, RSI, RDI //RBX salva-invocante

d) RSP, RBP

RSP especial (el único especial)

50. Una sentencia en C del tipo "while (test) body;" puede transformarse en código "goto"

como:

a) loop:

body;

if (test) goto loop;

-> b) if (!test) goto done;

loop:

body;

if (test) goto loop;

done:

c) loop:

if (test) goto done;

body;

goto loop;

done:

d) if (test) goto true;

goto done;

true:

body;

done:

51. Indicar cuál es la dirección de la instrucción mov en el siguiente desensamblado, donde se ha borrado parte de la dirección

0804xxxx: 74 12 je 08048391

0804xxxx: b8 00 00 00 00 mov \$0, %eax

a) 08048391 + 12 = 08048403

-> b) 0804837f

Tema 2.3, tr.7-8

c) 08048391 – 12 = 08048379

sí, es esa resta, pero está mal hecha, como si fueran números en decimal

d) 0804837d

52. El resultado de desplazar 0xFFOF a la derecha aritméticamente 4 veces es:

-> a) 0xFFFF0

b) 0xOFF0

c) 0xF0FF

d) 0xFOFO

53. ¿Cuál es el resultado de evaluar la expresión 0b1110 ^ 0b1010 en lenguaje C?

(0b indica que el número está expresado en binario)

a) 0b1010

-> b) 0b0100

c) 0b1111

d) 0b0110

54. Para crear espacio en la pila para variables locales sin inicializar suele realizarse la

siguiente operación:

-> a) Restar una cantidad positiva a ESP

b) Restar una cantidad positiva a EBP

c) Sumar una cantidad positiva a ESP

d) Sumar una cantidad positiva a EBP

55. Respecto a direccionamiento a memoria en ensamblador IA32 (sintaxis AT&T), de la

forma D(Rb, Ri, S), sólo una de las siguientes afirmaciones es FALSA. ¿Cuál?

a) ESP no se puede usar como registro índice

b) El desplazamiento D puede ser una constante literal (1, 2 ó 4 bytes)

c) El factor de escala S puede ser 1, 2, 4, 8

-> d) EBP no se puede usar como registro base

56. ¿Cuál de las siguientes afirmaciones sobre compilación C->ASM es falsa?

- > a) Puede que el compilador altere por optimización el orden de los parámetros pasados en una llamada a función, si el marco de pila resultante es más eficiente
- b) Puede que el compilador elimine por optimización construcciones C enteras (como un bucle for completo), si se conoce el resultado en tiempo de compilación
- c) Puede suceder que varias sentencias C se traduzcan por una única instrucción ASM
- d) Puede que el compilador altere el orden del código C, apareciendo antes la traducción de sentencias C posteriores

57. El ajuste de marco de pila que gcc (Linux/IA32) prepara para todas las funciones consiste en las instrucciones

-> a) pushl %ebp

movl %esp, %ebp

b) pushl %esp

movl %ebp, %esp

c) movl %esp, %ebp

pushl %esp

d) movl %ebp, %esp

pushl %ebp

58. ¿Cuál es la diferencia entre las instrucciones mov y lea?

a) mov puede usarse para copiar un registro a otro, mientras que lea no

lea (%ebx), %eax

b) lea puede usarse para copiar un registro a otro, mientras que mov no

mov %ebx, %eax

c) lea accede a la posición indicada, mientras que mov no lo hace

-> d) mov accede a la posición indicada, mientras que lea no lo hace

59. Si la variable val está almacenada en ebx y la variable x está almacenada en eax, la

sentencia val ^= x; se puede traducir a ensamblador como:

-> a) xorl %eax,%ebx

b) xorl %ebx,%eax

c) andl %ebx,%eax

d) testl %eax,%ebx

60. Después de ejecutar una instrucción de suma sobre dos números con signo de la que

sabemos que no provocará overflow (los dos números son pequeños en valor absoluto), queremos comprobar si el resultado de la suma es menor que 0. ¿Qué flag necesita comprobar la instrucción de salto condicional equivalente a... ?

if (resultado<0) then goto label

- a) OF
- b) CF
- > c) SF
- d) ZF

61. Considerar las siguientes declaraciones de estructuras en una máquina Linux de 64-bit.

```
struct RECORD {  
    long value2;  
    int ref_count;  
    char tag[4];  
};  
  
struct NODE {  
    double value;  
    struct RECORD record;  
    char string[8];  
};
```

También se declara una variable global "my\_node" como sigue:

```
struct NODE my_node;
```

Si la dirección de my\_node es 0x601040, ¿cuál es el valor de &my\_node.record.tag[1]?

- a) 0x601050
- b) 0x601054
- > c) 0x601055
- d) Ninguna de las anteriores

62. La instrucción movl %esp,%ebp

- a) Intercambia los contenidos de los registros ESP y EBP.
- b) Introduce en la pila el contenido del registro EBP.
- > c) Copia el contenido del registro ESP en el registro EBP.
- d) Mueve el contenido del registro ESP al registro EBP, poniendo a 0 el registro ESP.

63. ¿Cuál de las siguientes listas está correctamente ordenada temporalmente?

- > a) 8086, 486, Pentium MMX, Pentium III, Pentium 4, Core 2
- b) 8086, 486, Pentium III, Pentium MMX, Core 2, Pentium 4
- c) 486, 8086, Pentium MMX, Core 2, Pentium III, Pentium 4
- d) 486, 8086, Core 2, Pentium III, Pentium 4, Pentium MMX

64. En IA32 el puntero de pila es:

- a) una dirección de memoria de 32 bits almacenada en el contador de programa
- b) un conjunto de posiciones de memoria usadas para almacenar información temporal durante la ejecución del programa
- > c) un registro de 32 bits en el microprocesador
- d) un registro de 16 bits en el microprocesador

65. En un sistema x86-64, si %rsp tiene el valor 0x7fffff0000 inmediatamente antes de ejecutar una instrucción retq, ¿cuál es su valor inmediatamente después?

- a) 0x7fffff0000
- +0 no: no libera nada
- b) 0x7ffffefff8
- 8 no: pila no libera hacia abajo
- c) 0x7fffff0004
- +4 no: retq retira %rip de 64bits
- > d) 0x7fffff0008
- +8 ok

66. ¿Cuál de los siguientes fragmentos de código deja en %eax un resultado distinto a los otros tres fragmentos?

- a) not %eax

add \$1, %eax

//compl.2, definición: neg = not + 1

- b) xor %edx, %edx

sub %eax, %edx

mov %edx, %eax

//0-eax = -eax = compl.2

- > c) mov \$-1, %edx

sub %eax, %edx

mov %edx, %eax

d) neg %eax

//compl.2

67. Si AX = 0xFA50 y ejecutamos AND \$0xFF, %AX

a) El registro AL se pone a 0

b) El registro AL se pone a FF

c) El registro AH se pone a FF

-> d) El registro AH se pone a 0

68. La diferencia entre el flag de acarreo y de overflow es que...

-> a) ambos se recalculan tras cada operación aritmético-lógica con ints, correspondiendo al programador consultar uno u otro según piense que sus datos son con o sin signo

b) el de acarreo indica que el resultado es demasiado grande (para p.f.) o positivo (si se trata de ints) para poder almacenarse, el de overflow indica que es demasiado pequeño (p.f.) o negativo (ints)

c) el flag de acarreo indica que ha habido acarreo en una operación con números enteros (ints), el de overflow indica que ha habido desbordamiento en una operación con números en punto flotante (p.f.)

d) uno se activa cuando se opera con números con signo y otro cuando son sin signo

69. Siendo EDX=0xf000 y ECX=0x0100, ¿cuál de las siguientes instrucciones tiene como dirección efectiva 0xf400?

-> a) xorl (%edx, %ecx, 4), %eax

b) leal 0x80(%edx, 2), %eax

c) movl 0x8(%edx), %eax

d) addl (%edx, %ecx), %eax

70. Es responsabilidad del procedimiento llamado salvaguardar los registros:

a) %eax, %ebx, %ecx, %edx

b) %esi, %edi

c) %eax, %edx, %ecx

-> d) %ebx, %esi, %edi

71. ¿Cuál afirmación es FALSA al comparar las arquitecturas x86 y x86-64?

- a) El tamaño de un double es el mismo
- > b) El tamaño de un puntero es el mismo
- c) El tamaño de un entero (int) es el mismo
- d) El tamaño de las posiciones de memoria es el mismo

72. Respecto a direccionamiento a memoria en ensamblador IA32 (sintaxis AT&T), de la

forma D(Rb, Ri, S), sólo una de las siguientes afirmaciones es FALSA. ¿Cuál?

- a) El desplazamiento D puede ser una constante literal (1, 2 ó 4 bytes)
- b) El desplazamiento D también puede ser el nombre de una variable (que se traduce por su dirección, de 4bytes)
- > c) Los registros base e índice (Rb y Ri) pueden ser cualesquiera de los 8 registros enteros (EAX...ESP)
- d) El factor de escala S puede ser 1, 2, 4, 8

73. Sean un int\*a y un int n. Si el valor de %ecx es a y el valor de %edx es n, ¿cuál de los siguientes fragmentos de ensamblador se corresponde mejor con la sentencia C return a[n]?

-> a) mov (%ecx,%edx,4),%eax

ret

b) mov (%ecx,%edx,1),%eax

ret

c) leal (%ecx,%edx,4),%eax

ret

d) ret (%ecx,%edx,4)

74. ¿Cuál de las siguientes instrucciones es errónea? (sale mensaje de error al intentar ensamblar):

a) movw %dx, (%eax)

-> b) pushb \$0xFF

Tema 2.1, tr.28, Hallaron, p.205-208, Probl. 3.2.5

[CS:APProb3.2]: Para cada una de las siguientes líneas de lenguaje ensamblador, determinar el sufijo

de instrucción apropiado basándose en los operandos:

...

5) push \$0xFF

...

c) movswl (%eax), %edx

d) movzbl %dl, %eax

75. El cuerpo del siguiente código C:

```
unsigned copy(unsigned u) {return u;}
```

puede traducirse a ensamblador como:

a) movl 8(%ebp), (%eax)

b) movl 8(%esp), %ebp

c) movl %ebp, 8(%eax)

-> d) movl 8(%ebp), %eax

76. La instrucción leave hace exactamente lo mismo que

a) pop %eip

-> b) mov %ebp, %esp; pop %ebp

c) mov %esp, %ebp; pop %esp

d) ret

77. El sufijo l de la instrucción movl significa:

a) Que la instrucción usa ordenación de bytes little-endian en lugar de big-endian.

b) Que la instrucción afecta a la parte de 16 bits más a la izquierda de los operandos (left word).

-> c) Que la instrucción trabaja con operandos de 32 bits (long word).

d) Que la instrucción afecta a los 16 bits menos significativos de los operandos (low word).

78. Comparando las convenciones de llamada de gcc Linux IA32 con x86-64 respecto a registros

a) En IA32 %esi es salva-invocado, y en x86-64 %rsi es salva-invocado también  
%rsi es 2º argumento - salva-invocante (amarillo en tr.2.4.5)

b) En IA32 %ebp es especial (marco de pila), y en x86-64 %rbp también  
%rbp es salva-invocado (verde en tr.2.4.5)

-> c) En IA32 %ecx es salva-invocante, y en x86-64 %rcx es salva-invocante también

d) En IA32 %ebx es salva-invocante, pero en x86-64 %rbx es salva-invocado  
%ebx es salva-invocado (verde en tr.2.3.36)

79. ¿Cuál de los siguientes lenguajes no permite el paso de parámetros por referencia?

- a) C++
- b) Pascal
- c) FORTRAN
- > d) C

80. Dada la siguiente declaración en lenguaje C, una estructura de este tipo podría ocupar, bien sea en un sistema Linux IA32 o bien en uno x86-64, un total de...

```
struct a{  
    int i;  
    double d;  
    char c;  
    short s; };
```

- a) 18 B

no, sería 16 B en IA32, sólo 1 B relleno entre c y s

- b) 20 B

- c) 22 B

- > d) 24 B

sí, 4 B relleno entre i y d, 1B entre c y s, 4B relleno tras s

81. ¿Qué salida produce el siguiente código? Asumir representación de datos de arquitectura IA32.

```
unsigned int x = 0xDEADBEEF;  
unsigned short y = 0xFFFF;  
signed int z = -1;  
if (x > (signed short) y)  
    printf("Hello");  
if (x > z)  
    printf("World");
```

(Aviso: en comparaciones con un dato unsigned se pasa el otro dato a unsigned)

- a) Imprime "Hello"
- b) Imprime "HelloWorld"
- > c) No imprime nada

En el problema 3.4 sólo se explica que una extensión de tamaño se hace según el fuente sea

signed (extensión sgn) o unsigned (ext. con ceros), pero no se explica la sección 2.2.6 (y 2.2.5) en donde se aclara que las comparaciones con unsigned se hacen unsigned. Evitar esta pregunta en el futuro.

d) Imprime "World"

82. ¿Cuál de los siguientes microprocesadores no es de 64 bits?

a) Core 2

-> b) Pentium III

c) Core i7

d) Itanium

83. Si el registro %eax contiene el siguiente valor binario:

11111111 10101010 01010101 11110000

¿Cuál será el valor de %eax tras ejecutar la instrucción xorb %al, %al?

a) 00000000 10101010 01010101 11110000

-> b) 11111111 10101010 01010101 00000000

c) 11111111 10101010 01010101 11110000

d) 00000000 00000000 00000000 00000000

84. Respecto a los registros enteros en arquitectura IA32

-> a) Hay 8 de cada tamaño (32, 16, 8 bits), aunque no todos los registros tienen versión en 8 y 16 bits

Tema1 tr.26

b) Son de 32bits en las aplicaciones de 32bit, y de 64bits en las aplicaciones de 64bit

c) No hay distintos tamaños, son sólo registros de 32 bits, como corresponde a dicha arquitectura

d) Hay 8, y en cada uno puede accederse a todos los 32 bits (p.ej. EAX), a los 16 bits menos significativos (p.ej. AX) ó a los 8 LSBs (p.ej. AL)

85. Para traducir una construcción if-then-else de lenguaje C a lenguaje ensamblador, gcc utiliza generalmente

-> a) un salto condicional, según la condición opuesta a la del código C, y otro salto incondicional

b) dos saltos condicionales y dos saltos incondicionales

c) dos saltos condicionales (uno para la parte if y otro para la parte else)

d) un salto condicional, según la condición expresada en el código C

86. ¿Cuál afirmación es FALSA en arquitecturas x86-64?

a) El tamaño de un puntero es 64 bits

b) El tamaño de los registros es 64 bits

-> c) El tamaño de las posiciones de memoria es 64 bits

d) El tamaño de un double es 64 bits

87. Si A y B son dos enteros almacenados respectivamente en %eax y %ebx, ¿cuál de las siguientes implementaciones de if (!A && !B) {...then part...} es incorrecta?

-> a) test %ebx, %eax

jne not\_true

...then part...

not\_true:

...

test hace AND, no OR

equivale a if (A&&B) {... then\_part ...}

b) or %ebx, %eax

jne not\_true

...then part...

not\_true:

...

de Morgan

equivale a if ( !(A|B) ) {... then\_part ...}

c) test %eax, %eax

jne not\_true

test %ebx, %ebx

jne not\_true

...then part...

not\_true:

...

short-circuit

gcc usa test %eax,%eax para comprobar cmp \$0,%eax

d) cmp \$0, %eax

jne not\_true

cmp \$0, %ebx

jne not\_true

...then part...

not\_true:

88. Si EAX=2, EBX=5 y M[3]=3, ¿qué valores quedan tras ejecutarse la instrucción XOR %BL, 1(%EAX)?

a) EAX=6 , EBX=5 , M[3]=2

b) EAX=5 , EBX=6 , M[3]=2

c) EAX=6 , EBX=2 , M[3]=5

-> d) EAX=2 , EBX=5 , M[3]=6

89. En IA32 la pila es:

a) una dirección de memoria de 32 bits almacenada en el contador de programa

b) un registro de 32 bits en el microprocesador

-> c) un conjunto de posiciones de memoria usadas para almacenar información temporal durante la ejecución del programa

d) un registro de 16 bits en el microprocesador

90. ¿Cuál fue el primer procesador de Intel de 64-bits en la familia x86(-64)?

a) 386

b) Core i7

c) 8086

-> d) Pentium 4F

91. Respecto a registros salva-invocante y salva-invocado en GCC/Linux IA32, ¿cuál de éstos es de distinto tipo que el resto?

a) EBX

b) EDI

-> c) EAX

d) ESI

92. Para traducir una construcción do-while de lenguaje C a lenguaje ensamblador, gcc utiliza generalmente

-> a) un salto condicional hacia atrás, según la misma condición que en lenguaje C

Tema 2.2, tr.40

b) un salto condicional hacia atrás, según la condición opuesta a la de lenguaje C

c) un salto condicional hacia adelante, según la condición opuesta a la de lenguaje C

d) un salto condicional hacia adelante, según la misma condición que en lenguaje C

93. ¿Cuál de las siguientes instrucciones no modifica necesariamente la secuencia de ejecución del programa?

a) CALL dir

b) JMP dir

c) RET

-> d) JNE dir

94. Estudiando el listado de una función C presuntamente compilada con gcc en modo 64bit

(x86-64), nos dicen que la instrucción "movl (%rdi), %eax" carga en el registro EAX el valor adonde apunta el primer argumento.

a) Está mal, porque el primer argumento de una función C no se pasa en RDI

b) Está mal, porque EAX no se puede usar en modo 64bit, debería ser RAX

-> c) Está bien, y pone a cero los 32 bits más significativos de RAX

d) Está mal, porque EAX no se carga con ningún valor

95. En el direccionamiento inmediato el operando reside en:

-> a) en la instrucción tras el código de operación

b) en la pila

c) en un registro del procesador

d) en memoria, en la dirección indicada

96. Sobre el direccionamiento relativo a contador de programa:

a) Favorece la implementación de código reubicable.

b) Su uso en los saltos y llamadas a subrutinas reduce el tamaño de la instrucción.

c) Es adecuado para alcanzar instrucciones próximas a la que se está ejecutando.

-> d) Todas las respuestas son ciertas.

97. Las siguientes afirmaciones sugieren que el tamaño de varios tipos de datos en C (usando el compilador gcc) son iguales tanto en IA32 como en x86-64. Sólo una de ellas es FALSA.

¿Cuál?

- > a) El tamaño de un long es 4 bytes
- b) El tamaño de un int es 4 bytes
- c) El tamaño de un unsigned es 4 bytes
- d) El tamaño de un short es 2 bytes

98. Si EAX=2, EBX=5 y M[3]=3, ¿qué valores quedan tras ejecutarse la instrucción XOR %BL, 1(%EAX)?

- a) EAX=6 , EBX=5 , M[3]=2
- b) EAX=5 , EBX=6 , M[3]=2
- > c) EAX=2 , EBX=5 , M[3]=6
- d) EAX=6 , EBX=2 , M[3]=5

99. ¿Cuál de las siguientes afirmaciones sobre la instrucción leave es cierta?

- a) Equivale a mov %esp,%ebp seguida de pop %ebp //mov al revés
- b) Se ejecuta justo después de retornar de un procedimiento
- lo típico después de retornar (de una función C) es pop/add \$n,%esp y/o mov %eax,...
- c) Equivale a pop %ebp seguida de mov %ebp,%esp

en realidad es mov %ebp,%esp; pop %ebp

-> d) No es obligatorio usarla. En su lugar puede realizarse una secuencia explícita de operaciones mov y pop

100. En el fragmento de código

804854e:e8 3d 06 00 00 call 8048b90 <main>

8048553:50 pushl %eax

la instrucción call suma al contador de programa la cantidad:

- a) 0x0804854e
- > b) 0x0000063d
- c) 0x08048553
- d) 0x50

101. Alguna de las siguientes afirmaciones sobre sistemas Linux x86-64 no es cierta

- > a) Todos los argumentos de función se pasan a través de la pila
- b) %rax se usa para devolver los valores de retorno de funciones
- c) %eax y %ebx pueden usarse como en un sistema IA32
- d) %rbp se puede usar como cualquier otro registro (no hay puntero base)

102. ¿Cuál de los siguientes no es un modo de direccionamiento IA32?

- > a) Cache
- b) Registro
- c) Memoria
- d) Inmediato

103. Considerar las siguientes declaraciones de estructuras en una máquina Linux de 64-bit.

```
struct RECORD {  
    long value2;  
    int ref_count;  
    char tag[4];  
};  
  
struct NODE {  
    double value;  
    struct RECORD record;  
    char string[8];  
};
```

También se declara una variable global "my\_node" como sigue:

```
struct NODE my_node;
```

¿Cuál es el tamaño de my\_node en bytes?

- a) 28
- > b) 32
- struct record 8+4+4=16B 8x, struct node 8+16+8=32B 8x
- c) 40
- d) Ninguno de los anteriores

104. El registro SP / ESP / RSP...

- a) es un registro transparente al usuario y contiene la dirección de memoria a la que se está accediendo  
sería MAR o el equivalente en IA32
- b) es un registro transparente al usuario y contiene la instrucción que se está ejecutando  
ni siquiera IP apunta a la instrucción en ejecución, sino a la siguiente
- > c) es un registro de propósito específico y contiene la dirección de la cima de la pila

específico en el sentido de que no se puede usar como índice y se usa implícitamente (sin nombrarlo) en push, pop, call, ret...

d) es un registro de propósito específico y contiene la dirección de la siguiente instrucción a ejecutar

105. Si %rsp vale 0xdeadbeefdeadd0d0, ¿cuál será su nuevo valor después de que se ejecute pushq %rbx?

a) 0xdeadbeefdeadd0d4

+4 no: pila no crece hacia arriba

-> b) 0xdeadbeefdeadd0c8

$0xd0d0 - 0x8 = 0xd0c8$

c) 0xdeadbeefdeadd0cc

-4 no: pushq ocupa quad word

d) 0xdeadbeefdeadd0d8

+8 no

106. ¿Cuál de las siguientes secuencias de instrucciones calcula  $a=b-a$ , suponiendo que %eax es a y %ebx es b?

a) subl %eax, %ebx

sería  $b=b-a$

b) subl %ebx, %eax

sería  $a=a-b$

c) notl %eax

addl %ebx, %eax

sería  $a=-a-1+b=b-a-1$

-> d) Varias o ninguna de las respuestas anteriores son correctas, no se puede marcar una y sólo una

107. En un sistema de 32bits, ¿cuál de las siguientes expresiones C es equivalente a la expresión  $(x[2] + 4)[3]$ ?

(Asumir que x se ha declarado como int \*\*x. Recordar que C usa aritmética de punteros.

Notar que muchos de los paréntesis no son necesarios, sólo se han añadido para evitar confusiones por precedencia de operadores)

-> a)  $*((x + 2)) + 7$

ok,  $(X[2])[7]$

b)  $*((\ast x) + 2) + 7$

sería  $X[0][2+7]$

c)  $(\ast\ast(x + 2) + 7)$

sería  $X[2][0]+7$ , la última suma sería suma entera, no aritmética punteros

d)  $*(\ast(x + 8)) + 28$

108. ¿Cuál de las siguientes instrucciones x86 se puede usar para sumar dos registros y guardar el resultado sin sobrescribir ninguno de los registros originales?

a) mov

-> b) lea

c) add

d) Ninguna de ellas

109. La instrucción test es...

a) Lo mismo que and

b) Lo mismo que sub, pero no guarda el resultado, sólo ajusta los flags

c) Lo mismo que sub

-> d) Lo mismo que and, pero no guarda el resultado, sólo ajusta los flags

110. La primera letra (l) de la instrucción lea:

a) indica que la instrucción afecta a los 16 bits menos significativos del operando destino (low word)

b) indica que la instrucción usa ordenación de bytes little-endian

c) indica que la instrucción trabaja con un operando destino de 32 bits (long word)

-> d) forma parte del nemotécnico de la instrucción

111. ¿Cuál de las siguientes instrucciones es incorrecta en ensamblador GNU/as IA32?

a) mov (%esp),%ebp

-> b) pop %eip

No existe pop %eip, usar ret para conseguir ese objetivo.

El problema 3.30 avisa de que call/pop es la única forma de copiar %eip, sugiriendo que no se puede con mov/lea ni con push/pop.

c) lea 0x10(%esp),%ebp

d) pop %ebp

112. Un programa de ordenador que convierte un programa fuente de alto nivel completo en lenguaje máquina se llama un:

- a) simulador
- b) ensamblador
- > c) compilador
- d) intérprete

113. En la secuencia de programa siguiente:

804854e:e8 3d 06 00 00 call 8048b90 <main>

8048553:50 pushl %eax

¿cuál es el valor que introduce en la pila la instrucción call?

- a) 0x8048b90
- b) 0x804854e
- c) 0x804854f
- > d) 0x8048553

114. La instrucción necesaria para cargar 0x07 en %eax es:

- a) movl \$0x07,%ah
- > b) movl \$0x07,%eax
- c) movb \$0x07,%al
- d) movb \$0x07,%eax

115. La diferencia entre las instrucciones test y cmp consiste en que

- a) test realiza una operación and lógico, mientras que cmp realiza una resta
- b) test modifica sólo los flags lógicos (ZF,SF) mientras que cmp modifica los aritmético-lógicos (ZF,SF,CF,OF)
- > c) ambas respuestas son correctas
- d) ambas respuestas son incorrectas

116. La dirección efectiva del primer parámetro de llamada a una función suele calcularse

desde el código de la función como:

- a) EBP-8
- b) EBP-4
- > c) EBP+8
- d) EBP+4

117. ¿Cuál es el popcount (peso Hamming, nº de bits activados) del número 29?

- a) 2
- b) 3
- > c) 4

$29 = 0x1D = 16+8+4+1 = 0b0001\ 1101 \rightarrow \text{popcount } 4$

- d) 5

118. Un procesador cuya instrucción CALL guardara la dirección de retorno en un registro RL (llamado "de enlace"):

- a) Permitiría llamadas anidadas y recursivas.
- b) Permitiría llamadas recursivas, pero no anidadas.
- c) Permitiría llamadas anidadas, pero no recursivas.
- > d) No permitiría llamadas anidadas ni recursivas.

119. Si %edx contiene 0xf000 y %ecx contiene 0x0100, el direccionamiento

$0x80(%ecx,%edx,2)$  se refiere a la posición

- a) 0xf182
- b) 0xf280
- > c) 0x1e180
- d) Ninguna de las respuestas anteriores es correcta

120. Si la variable val está almacenada en ebx y la variable x está almacenada en eax, la sentencia  $\text{val} \wedge= x;$  se puede traducir a ensamblador como:

- a) testl %eax,%ebx
- > b) xorl %eax,%ebx
- c) andl %ebx,%eax
- d) xorl %ebx,%eax

121. Un archivo .o que contiene código objeto:

- a) Contiene las direcciones definitivas de las variables globales.
- b) Incluye el código de las funciones de biblioteca a las que llame.
- > c) Contiene instrucciones máquina binarias.
- d) Puede ejecutarse directamente.

122. Alguna de estas opciones contiene algún elemento que no corresponde a los contenidos de un marco de pila GCC/Linux IA32.

a) Argumentos de llamada a la función y Dirección de retorno

-> b) Variables globales y Valor de retorno de la función

c) Variables locales y Registros salva-invocado

d) Antiguo marco de pila y Registros salva-invocante

123. El resultado de desplazar aritméticamente dos posiciones hacia la derecha el número de

8 bits en complemento a dos -32 es:

a) 56

b) -128

-> c) -8

d) Ninguno de los resultados anteriores es correcto

124. La zona roja en x86-64 Linux es...

a) una zona de pila en donde no deben escribir las funciones invocadas (porque si se produjera una interrupción, el manejador de interrupción sobreescribiría los valores escritos en pila)

-> b) una zona bajo (RSP) (adonde apunta RSP) que una función puede usar sin reservarla, pero sólo si no llama a ninguna otra función mientras la usa

c) una zona de pila en donde pueden escribir las funciones invocadas, pero teniendo en cuenta que los valores escritos pueden verse alterados si se produce una interrupción

d) una zona bajo (RBP) (adonde apunta RBP) que una función puede usar sin reservarla, pero sólo si no llama a ninguna otra función mientras la usa

125. Un overflow nunca puede ocurrir cuando:

a) se suman dos números positivos

0x7fffffff + 0x1

-> b) se suma un número positivo a un número negativo

resultado queda entre ambos => se puede representar

c) se suman dos números negativos

0x80000000 + 0xffffffff

d) se resta un número positivo de un número negativo

0x80000000 - 0x00000001

126. ¿En qué registro se pasa el primer argumento a una función según el estándar \_cdecl en una arquitectura IA32?

a) edi

b) esi

c) eax

-> d) Las anteriores respuestas son erróneas

se pasa en registros en x86-64

127. GCC/Linux IA32 resuelve el ajuste de marco de pila mediante las instrucciones:

-> a) pushl %ebp; movl %esp, %ebp

b) movl %ebp, %esp; popl %ebp

c) pushl %esp; movl %ebp, %esp

d) movl %esp, %ebp; popl %esp

128. Si ECX vale 0, la instrucción adc \$-1,%ecx

a) Pone CF=1

b) Cambia CF

c) Pone CF=0

-> d) No cambia CF

129. ¿Cuál de las siguientes parejas de mnemotécnicos de ensamblador IA32 corresponden a

la misma instrucción máquina?

a) CMP (comparar), SUB (restar)

-> b) JZ (saltar si cero), JE (saltar si igual)

c) JC (saltar si acarreo), JL (saltar si menor, para números con signo)

d) SAR (desplazamiento aritmético a la derecha) / SHR (desplazamiento lógico a la derecha)

130. ¿Cómo se devuelve en ensamblador x86-64 Linux gcc el valor de retorno de una función

long int al terminar ésta?

a) La instrucción RET lo almacena en un registro especial de retorno.

b) Por convención se guarda en %eax.

c) Se almacena en pila justo encima de los argumentos de la función.

-> d) Ninguna de esas formas es la correcta

131. Considere una función C declarada así:

void fun4arg(int a, int b, int c, int d);

Suponiendo que fun4arg se ha compilado para una máquina x86 IA32 con enteros de 4

bytes, ¿cuál sería la dirección del argumento b relativa a %ebp, en el marco de pila de

fun4arg?

a) %ebp + 8

b) %ebp + 16

-> c) %ebp + 12

d) %ebp + 20

132. Para desplazar %eax a la derecha un número variable de posiciones  $\leq 32$ , indicado en %ebx, se puede hacer

a) sar %ebx, %eax

no existe

-> b) mov %ebx, %ecx

sar %cl, %eax

ver problema Hallaron 3.8 y sección 3.5.3

c) sar %bl, %eax

no existe

d) No se puede, sar sólo admite un número fijo de posiciones (debe ser un valor inmediato)

133. Respecto a la convención de llamada usada en Linux/gcc:

a) Una subrutina que modifique algún registro debe restaurar su valor anterior antes de retornar

b) Todos los registros pueden ser modificados libremente por todas las subrutinas

-> c) Hay registros que pueden ser modificados libremente por las subrutinas, y otros que, si se modifican, se deben restaurar posteriormente. Y también hay registros especiales

d) Hay registros modificables, otros que deben ser restaurados, y las subrutinas anidadas deben respetar los registros modificables que están en uso por otras subrutinas

134. Si la dirección del primer elemento de un vector de enteros z está almacenada en el registro %edx y la variable entera i está almacenada en el registro %eax, la instrucción máquina que realiza la operación  $z[i]++$  es:

a) addl \$4, (%eax,%edx)

no,  $z[i]++$  suma 1, no 4

-> b) addl \$1, (%edx,%eax,4)

c) addl \$1, (%edx,%eax)

no, falta multiplicar ix4 para indexar en array enteros

d) addl \$1, (%eax,%edx,4)

no, (dirección de z)x4 no

135. ¿Cuál de las siguientes afirmaciones es incorrecta?

- a) En lenguajes de alto nivel no se tiene acceso a detalles del procesador como las tablas de páginas, el paso de modo usuario a modo supervisor, el bit de paridad, etc.
- b) En lenguaje ensamblador cada instrucción se corresponde con una instrucción máquina.
- > c) En lenguaje ensamblador las instrucciones se escriben en binario.
- d) El lenguaje de alto nivel es más portable que el lenguaje máquina.

136. Para comprobar si el entero almacenado en EAX es cero (y posiblemente saltar a continuación usando JZ/JNZ), gcc genera el código

a) test %eax

test tiene 2 argumentos

b) cmp %eax, \$0

en todo caso sería cmp \$0,%eax

no le gusta a gcc eso

-> c) test %eax, %eax

d) cmp %eax

cmp tiene 2 argumentos

137. Cuando se ejecuta la instrucción ret al final de una subrutina:

- > a) la dirección almacenada en la cima de la pila se transfiere al contador de programa
- b) la dirección almacenada en la cima de la pila se transfiere al puntero de pila
- c) la dirección de comienzo de la pila se transfiere al puntero de pila
- d) la dirección de memoria de la instrucción ret se transfiere al contador de programa

138. Considerar la declaración C

int array[10] = {0, 1, 2, 3, 4, 5, 6, 7, 8, 9};

Suponer que el compilador tiene la dirección de array en el registro %ecx. ¿Cómo se movería el valor array[3] al registro %eax? Asumir que %ebx es 3.

- > a) movl (%ecx,%ebx,4),%eax
- b) movl 8(%ecx,%ebx,2),%eax
- c) leal 12(%ecx),%eax
- d) leal (%ecx,%ebx,4),%eax

139. La instrucción xor \$3, %eax tiene como resultado:

- a) Poner a 0 los últimos 3 bits del registro EAX
- > b) Cambiar 0<->1 (complemento a 1 de) los últimos 2 bits del registro EAX
- c) Poner a 1 el último bit del registro EAX
- d) Ninguno de los anteriores resultados

140. El Intel 8086:

- a) Contaba con 29 millones de transistores.
- b) Podía direccionar 1 GB.
- > c) Incluía instrucciones de multiplicación.
- d) Fue el primer microprocesador.

141. Al llamar a una función de 2 argumentos foo(arg1, arg2), ¿cuál es el orden correcto de las operaciones? (suponiendo convención de llamada x86 cdecl, y que foo requiere ajustar marco de pila, esto es, salvar %ebp)

- > a) push arg2, push arg1, call foo, push %ebp
- b) push arg2, push arg1, push %ebp, call foo
- c) push arg1, push arg2, call foo, push %ebp
- d) push arg1, push arg2, push %ebp, call foo

142. Dado el código C siguiente:

```
struct data {  
    char str[16];  
};  
  
char *f(struct data *ptr) {  
    return &(ptr->str[2]);  
}
```

la función se traducirá a ensamblador de x86-64 como:

- a) movq (%rdi,2), %rax; ret
- b) leaq (%rdi,2), %rax  
ret
- > c) leaq 2(%rdi), %rax  
ret
- d) movq 2(%rdi), %rax  
ret

143. En IA32, el registro contador de programa se denomina:

- > a) EIP
- b) PCR
- c) RIP
- d) PC

144. Respecto a tamaños de tipos integrales en x86 y x86-64, la excepción es que

- a) int pasa de 4 B (x86) a 8 B (x86-64)
- > b) long int pasa de 4 B a 8 B
- c) long long int pasa de 4 B a 8 B
- d) ninguna de las anteriores

145. Si tenemos un número n, de 64 bits, almacenado en la pareja de registros EDX:EAX (EDX contiene los 32 bits más significativos y EAX los 32 bits menos significativos) y queremos realizar la división  $n / (2^{32})$  entonces:

- a) Podemos usar las instrucciones siguientes, pero sólo en el caso de que n sea un número sin signo:

```
mov $0x100000000,%ecx  
div %ecx
```

- b) Podemos quedarnos con EDX, pero sólo en el caso de que n sea un número sin signo.

- c) Podemos usar las instrucciones siguientes tanto si n es un número con signo como sin signo:

```
mov $0x100000000,%ecx  
div %ecx
```

- > d) Podemos quedarnos con EDX tanto si n es un número con signo como sin signo.

Cocientes positivos o sin signo redondeados hacia cero, cocientes negativos redondeados hacia más negativo: es decir, redondeado siempre hacia inferior.

No es la división entera estándar de C (que redondea hacia cero) pero el enunciado tampoco lo exige.

146. La extensión de signo a m bits de un número original N de n bits, con  $m > n$ , consiste en:

- a) Realizar la operación  $2^m - N - 1$
- b) Incrementar la cantidad de bits a m rellenando con unos por la izquierda.
- c) Realizar la operación  $2^m - N$
- > d) Incrementar la cantidad de bits a m preservando el signo y el valor del número.

147. En un sistema Linux x86-64, ¿cuál de las siguientes variables ocupa más bytes en memoria?

a) char a[7]

-> b) int \*c

c) short b[3]

d) float d

148. ¿Cuál de las siguientes secuencias de instrucciones multiplica el (contenido del) registro EAX por 18?

a) sarl \$1, %eax

imul \$9, %eax

sería shl

b) imull \$0x18, %eax

sería \$18 sin 0x hex

c) shll \$18, %eax

equivale a  $x2^{18}$

-> d) leal (%eax,%eax,8), %eax

leal (%eax,%eax), %eax

149. En los casos concretos indicados para las siguientes instrucciones IA32, ¿cuál no funciona como instrucción de transferencia?

a) pushl %eax

-> b) cmpl 0x08048040, %eax

c) movl \$0x15, %eax

d) leal variable, %eax

150. En un sistema IA32 Linux, ¿cuál es el tamaño de un long?

a) 8 bytes

-> b) 4 bytes

c) 6 bytes

d) 2 bytes

151. ¿Cuál de las siguientes secuencias de instrucciones multiplica %eax por 10?

a) leal(%eax,%eax,4), %eax

sall \$2, %eax

sería x5x4=x20

b) imull \$0x10, %eax //sería x16

c) addl %eax, %eax

shll \$5, %eax

sería x2x32=x64

-> d) Varias o ninguna de las respuestas anteriores son correctas, no se puede marcar una y sólo una

152. Estudiando el listado de una función C presuntamente compilada con gcc en modo 64bit (x86-64), nos dicen que la primera instrucción, movl %eax, (%rdi), carga en EAX el valor adonde apunta el primer argumento.

a) Está mal, porque EAX no se puede usar en modo 64bit, debería ser RAX

b) Está bien, y pone a cero los 32 bits más significativos de RAX

-> c) Está mal, porque EAX no se carga con ningún valor

d) Está mal, porque el primer argumento de una función C no se pasa en RDI

153. La instrucción cmovb %edx, %eax

a) copia en %eax el contenido de %edx si %eax es menor que %edx

haría falta cmp %edx, %eax previo para ajustar CF

b) copia en %eax el byte de memoria apuntado por la dirección contenida en %edx

inventado, sería (%edx) y además %eax es de tamaño l (long)

-> c) copia en %eax el contenido de %edx si el indicador de acarreo es 1

"below" equivale a CF

d) copia el byte bajo de %edx en el byte bajo de %eax

154. Una función C llamada get\_el() genera el siguiente código ensamblador. Se puede

adivinar que:

movl 8(%ebp), %eax

leal (%eax,%eax,4), %eax

addl 12(%ebp), %eax

movl var(%eax,4), %eax

-> a) var es un array bidimensional de enteros, con cinco columnas

b) var es un array bidimensional de enteros, no se pueden adivinar dimensiones

c) var es un array multi-nivel (punteros a enteros) de cuatro filas

d) var es un array multi-nivel pero no se pueden adivinar las dimensiones

155. ¿Cuál de las siguientes afirmaciones es falsa respecto al lenguaje C?

- a) En lenguaje C, al llamar a una subrutina o función se introducen los parámetros en la pila y después se realiza una llamada a la subrutina
- b) Los parámetros se introducen en la pila en el orden inverso a como aparecen en la llamada de C, es decir, empezando por el último y acabando por el primero
- c) Pasar a una función un puntero a una variable se traduce en introducir en la pila el valor de la dirección de memoria donde está almacenada la variable

-> d) Antes de volver de la rutina llamada, el programa en C se encarga de quitar de la pila los parámetros de llamada realizando varios pop

156. En un microprocesador de 4 bits, una operación en la que el bit 0 de un registro se copia en el acarreo, después el bit 1 se copia en el bit 0, después el bit 2 se copia en el bit 1, y por último el bit 3 se copia en el bit 2, es:

- a) Un desplazamiento lógico a la derecha.
- b) Una rotación a la derecha a través de acarreo.
- > c) Un desplazamiento aritmético a la derecha.
- d) Una rotación a la derecha.

157. Se definen la unión, variables, y función C, siguientes:

```
typedef union {  
    float f;  
    unsigned u;  
} bit_float_t;  
  
float f1;  
  
unsigned u1=0x80000000;  
  
float f2;  
  
float bit2float(unsigned u) {  
    bit_float_t arg;  
    arg.u = u;  
    return arg.f;  
}
```

¿Cuál afirmación es verdadera?

- a) Si asignamos `f1=bit2float(u1); f2=float(u1);` entonces `f1==f2`, ambos distintos de cero  
`float(u1)` es error de compilación, y además `f1` no es distinto de cero
- b) Si asignamos `f1=bit2float(u1); f2=(float)u1;` entonces `f1== 0.0` pero `f2==4294967296.00`  
0x8000 0000 son 2G, no 4G, `f2` está mal
- c) Si asignamos `f1=bit2float(u1); f2=float(u1);` entonces `f1==1073741824.00` y `f2== -1073741824.00`  
`float(u1)` es error de compilación, y además +-1G no es 0x8000 0000
- > d) Si asignamos `f1=bit2float(u1); f2=(float)u1;` entonces `f1== -0.0` pero `f2==2147483648.00` correcto, 0x8000 0000 es el bit de signo, y también es 2G
158. Si el contenido de EBP es 0x13000, ¿a qué dirección se hace referencia con la instrucción `INC -0x80(%EBP)?`
- a) 0x13080
  - > b) 0x12F80
  - c) 0x13000
  - d) 0x80

159. Para poner a 1 el bit 5 del registro %edx sin cambiar el resto de bits podemos usar la instrucción máquina:

- > a) or \$0x20, %edx
- b) and \$0x5, %edx
- c) and \$32, %edx
- d) or \$0b101, %edx

160. En la nomenclatura del ensamblador de IA32, una cantidad de 16 bits es designada como:

- a) half word
- > b) word
- c) double word
- d) quad Word

161. Si declaramos `int val[5]={1,5,2,1,3};` entonces

- a) `val[5]` es de tipo int y vale 3  
`val[4]==3, val[5]` apunta fuera del array
- b) `val+1` es de tipo int y vale 2

val+1==&val[1] es int\*

-> c) &val[2] es de tipo int\* y vale lo mismo que (void\*)val+8

En Sep15 faltaba (void\*) y entonces sería falsa por aritmética de punteros

d) val+4 es de tipo int\* y se cumple que \*(val+4)==5

Por aritmética de punteros, \*(val+4)==val[4]==3

162. ¿Cuál sería el "equivalente x86-64" del "pseudo-código C" rcx = ((int\*)rax)[rcx]?

a) lea (%rax,4,%rcx),%rcx

b) lea (%rax,%rcx,4),%rcx

c) mov (%rax,4,%rcx),%rcx

-> d) mov (%rax,%rcx,4),%rcx

eso sería movq, debería ser movl destino %ecx

si se indexa en int[] se debería obtener int, no long, si se hubiera dicho (long\*)rax tendríamos

que indexar 8x, no 4x, evitar esta pregunta

163. Una instrucción de "salto si igual" tiene que comprobar el valor de:

a) los bits de signo y desbordamiento

b) el bit de signo

c) el bit de acarreo

-> d) el bit de cero

164. La instrucción IA32 test sirve para...

-> Realizar la operación and lógico bit-a-bit (a&b) pero no guardar el resultado, sino

simplemente ajustar los flags

b) Realizar la operación resta (a-b) pero no guardar el resultado, sino simplemente ajustar los flags

c) Testear el código de condición indicado, y poner un byte a 1 si se cumple

d) Mover el operando fuente al destino, pero sólo si se cumple la condición indicada

165. Dentro de una función declarada como void swap(int \*xp, int \*yp), que intercambia los valores de los dos enteros cuyas direcciones de memoria (punteros) son pasadas como parámetros a la función, la instrucción movl 12(%ebp),%ecx copia en %ecx...

a) el valor del entero apuntado por el puntero pasado como primer parámetro

-> b) el valor del puntero pasado como segundo parámetro

c) el valor del puntero pasado como primer parámetro

d) el valor del entero apuntado por el puntero pasado como segundo parámetro

166. ¿Qué combinación de flags aritmético-lógicos corresponde al código de condición b (below)?

- > a) CF
- b) OF xor SF
- c) CF xor OF
- d) OF

167. Considerar las siguientes declaraciones de estructuras en una máquina Linux de 64-bit.

```
struct RECORD {  
    int value2;  
    short ref_count;  
    char tag[10];  
};  
  
struct NODE {  
    long value;  
    struct RECORD record;  
    char string[8];  
};
```

También se declara una variable global "my\_node" como sigue:

```
struct NODE my_node;
```

Si la dirección de my\_node es 0x600940, ¿cuál es el valor de &my\_node.record.tag[1]?

- a) 0x60094a
- b) 0x60094e
- > c) 0x60094f
- d) Ninguna de las anteriores

168. Si RCX vale -1, tras ejecutar las instrucciones

```
rol $1, %cl
```

```
rcr $2, %rcx
```

el nuevo valor de RCX y del flag CF es

- a. hay algún fallo de sintaxis o gramática en esas instrucciones
- b. RCX≠-1, CF mantiene su valor

-> c. RCX=-1, CF=1 el primer rol pone CF=1, el segundo recircula pero todos los bits son 1

d. no se puede marcar ninguna de las opciones anteriores

169. Para comprobar si el entero almacenado en RAX es cero (y tal vez saltar a continuación usando JZ/JNZ), gcc genera el código

a. cmp %eax, \$0 en todo caso sería cmp \$0,%eax - no le gusta a gcc eso

b. test %eax test tiene 2 argumentos

c. cmp %eax cmp tiene 2 argumentos

-> d. test %eax, %eax

170. Si RCX vale -1, tras ejecutar la instrucción sal \$3, %ecx el nuevo valor de RCX es

a. 0xffff ffff ffff ffff

b. 0xffff ffff ffff fff8

c. 0x1fff ffff ffff ffff

-> d. 0xffff fff8

171. Uno de los puntos clave de la traducción que gcc hace de una construcción switch-case de lenguaje C a lenguaje ensamblador es...

a. El salto condicional hacia atrás

b. El salto relativo a contador de programa

c. El salto directo

-> d. El salto indirecto

172. Sabiendo que las instrucciones de salto condicional codifican la dirección de salto con direccionamiento relativo a contador de programa (de 8 o 32 bits con signo), indicar cuál es la dirección de salto de la instrucción je en el siguiente desensamblado, donde se ha tachado precisamente dicha dirección.

40042f: 74 f4 je xxxxxx

400431: 5d pop %rbp

a. 400431 eso sería si el desplazamiento relativo fuera 0

b. 400525 eso es 0x400431 + 0xf4, se debe extender signo

-> c. 400425 Ej.3.15b p.245/367 del libro, y Tema2.3, tr.15-16

d. 40043d

173. Para traducir una asignación condicional (a=b?c:d;) de lenguaje C a lenguaje ensamblador, gcc puede que utilice...

a. Un salto incondicional, según la condición expresada en el código C, y otro salto

incondicional

b. Un salto condicional, según la condición opuesta a la del código C, y otro salto condicional incond.,

-> c. Una instrucción de movimiento condicional, pero sólo si el procesador es Pentium Pro/II o superior

d. Una instrucción de movimiento incondicional, pero sólo si el S.O. es de 64bits

174. Sabiendo que las instrucciones de salto condicional codifican la dirección de salto con direccionamiento relativo a contador de programa (de 8 o 32 bits con signo), indicar cuál es la dirección de la instrucción pop en el siguiente desensamblado, donde se ha tachado la parte de las direcciones.

xxxxxx: 77 02 ja 400547

xxxxxx: 5d pop %rbp

a. 400547

b. 400549

-> c. 400545

d. 400543

175. En la convención de llamada SystemV AMD64 seguida por gcc Linux/x86-64...

a. RAX es un registro salva-invocante, por eso en cualquier función hay que salvarlo antes de usarlo

-> b. R10 es un registro salva-invocante, por eso si es necesario hay que salvarlo antes de llamar a función

c. R11 es un registro salva-invocado, por eso en cualquier función hay que salvarlo antes de usarlo

d. RBP es un registro salva-invocado, por eso si es necesario hay que salvarlo antes de llamar a función

176. Una función C llamada get\_el() genera el siguiente código ensamblador. Se puede adivinar que:

leaq (%rdi,%rdi,4), %rax

addq %rax, %rsi

movl arr(%rsi,4), %eax

ret

- a. arr es un array multi-nivel (punteros a enteros) de cuatro filas
  - b. arr es un array multi-nivel pero no se pueden adivinar las dimensiones
  - c. arr es un array bidimensional de enteros, no se pueden adivinar dimensiones
- > d. arr es un array bidimensional de enteros, con cinco columnas

177. En un sistema Linux x86-64, ¿cuál de las siguientes expresiones es equivalente a la expresión C  $(x[2] + 4)[3]$ ? Suponer que previamente se ha declarado int \*\*x.

- a)  $*((x + 16)) + 28$

cierto si se hubiese hecho typecast (void\*)x para evitar aritmética punteros

- b)  $(*(*x + 2) + 4) + 3$

sería  $x[2][4]+3$ , la última suma sería suma entera, no aritmética punteros

- > c)  $*((x + 2) + 4) + 3$

sería  $x[2][4+3]$ ,  $+4+3 == +7$  sí sería aritmética punteros

- d)  $*(((*x) + 2) + 7)$

178. Si el registro rax contiene un long (64 bits CON signo) x, la secuencia de instrucciones siguiente:

cmpq \$10, %rax

ja dest

saltará a la etiqueta dest si:

- a)  $x > 9$

- b)  $x < 10$

- >c)  $x \leq -1 \mid x \geq 11$

- d)  $x \geq 0 \ \&& \ x \leq 10$

179. Usando el repertorio x86-64, para intercambiar el valor de 2 variables (por ejemplo A: .int 1 y B: .int 2) se pueden usar...

- >a) 4 mov, no menos (debido a la arquitectura R/M)

ver la función swap()

- b) dos instrucciones mov

- c) una instrucción mov y una instrucción lea

- d) 3 mov, no menos (se le llama "intercambio circular")

si fuera arquitectura M/M se podría hacer intercambio circular:

mov A,R1

mov B,A; // esta instrucción no es de repertorio R/M

mov R1,B

180. En x86-64 el puntero de pila es:

- a) una dirección de memoria de 64 bits almacenada en el contador de programa
- >b) un registro de 64 bits en el microprocesador  
%rsp, para ser más precisos
- c) un conjunto de posiciones de memoria usadas para almacenar información temporal durante la ejecución del programa
- d) un registro de 32 bits en el microprocesador

181. Los arrays bidimensionales en lenguaje C se almacenan en orden...

- a) "de fila a columna" (file-to-column)
- b) "de mayor a menor" (major-to-minor)
- >c) "mayor-de-fila" (row-major)
- d) "mayor-de-columna" (column-major)

182. Para crear espacio en la pila para variables locales sin inicializar suele realizarse la siguiente operación:

- a) Restar una cantidad positiva a RBP

no, los punteros a marco se salvan/recuperan, no se opera con ellos

- >b) Restar una cantidad positiva a RSP

sí, eso es reservar espacio en pila

- c) Sumar una cantidad positiva a RSP

no, eso es liberar espacio en pila

- d) Sumar una cantidad positiva a RBP

no, los punteros a marco se salvan/recuperan, no se opera con ellos

183. ¿Cuál de los siguientes registros x86-64 es distinto del resto en convenio de uso? (salvador/invocante/invocado)

- >a) RBX

sí, el único salva-invocado

- b) RCX

- c) RSI

- d) R8

184. ¿Cuál de las siguientes instrucciones es incorrecta?

- a) shr \$1,%rdx

- b) shr %rdx
- c) shr %cl,%rdx
- >d) shr %rcx,%rdx

185.¿Cuál es el popcount (peso Hamming, nº de bits activados) del número 19?

- a) 2
- >b) 3

$$19 = 0x13 = 16+3 = 0b0001\ 0011 \rightarrow \text{popcount } 3$$

- c) 4
- d) 5

186.Una función C declarada como int get\_var\_digit(size\_t index, size\_t digit) genera como código ensamblador

```
movq var(%rdi,8), %rax  
movl (%rax,%rsi,4), %eax  
ret
```

Se puede adivinar que:

- a) var es un array multi-nivel (punteros a enteros) de cuatro filas
- >b) var es un array multi-nivel pero no se pueden adivinar las dimensiones
- c) var es un array bidimensional de enteros, con ocho columnas
- d) var es un array bidimensional de enteros, con cinco columnas

187.La instrucción x86-64 test sirve para...

- a) Realizar la operación resta (a-b) pero no guardar el resultado, sino simplemente ajustar los flags

eso sería cmp b,a

- b) Mover el operando fuente al destino, pero sólo si se cumple la condición indicada

eso sería cmovCC src,dst

- >c) Realizar la operación and lógico bit-a-bit (a&b) pero no guardar el resultado, sino simplemente ajustar los flags

eso es

- d) Testear el código de condición indicado, y poner un byte a 1 si se cumple

eso sería setCC %al

188.¿Cuál de los siguientes grupos de instrucciones IA32 sólo modifican los indicadores de estado sin almacenar el resultado de la operación?

- a) ADC, SBB
- >b) CMP, TEST
- c) IMUL, IDIV
- d) AND, OR, XOR

189. En el fragmento de código

400544: e8 07 00 00 00 callq 400550 <mult2>

400549: 48 89 03        mov %rax,(%rbx)

la instrucción call suma al contador de programa la cantidad:

- a) 0x48
  - no, eso es el codop de la siguiente instrucción MOV
- b) 0x400544
  - no, esa es la dirección de la propia instrucción CALL
- >c) 0x00000007
  - sí, codop e8 lleva offset 07 00 00 00 en little-endian
- d) 0x00400549
  - no, esa es la dirección de retorno

## TEMA 3

1. ¿Cómo actúa el indicador de signo?

- a) Se pone a 1 cuando el resultado es positivo
- > b) Se pone a 1 cuando el resultado es negativo
- c) Se pone a 1 cuando el resultado es distinto
- d) Se pone a 0 cuando el resultado es negativo

2. La microprogramación vertical se caracteriza por tener:

- a) escaso o ningún solapamiento entre campos
- b) capacidad para expresar un alto grado de paralelismo en las microoperaciones a ejecutar
- c) microinstrucciones largas
- > d) mucha codificación

3. Un procesador con una unidad de control microprogramada tiene una memoria de control de 256 palabras de 16 bits, de las que 128 son diferentes. ¿Qué ahorro en número de bits obtendríamos si usáramos nanoprogramación?

- > a) 256 bits
- b) No se produce ahorro
- c) 4096 bits
- d) 3840 bits

4. La conexión de las salidas de tres registros hacia un bus común en el camino de datos puede realizarse usando...

- > a) dos multiplexores de 2 a 1
- b) tres conexiones directas al bus común
- c) tres demultiplexores
- d) dos buffers triestado

5. Sea un formato de microinstrucción que incluye dos campos independientes de 9 bits cada uno. Si se rediseña de modo que se solapen los dos campos, ¿cuántos bits se ahorran en cada microinstrucción?

- a) 1
- b) 4
- > c) 8

d) 9

6. ¿Cómo actúa el indicador N del registro de indicadores de estado?

- > a) Se pone a 1 cuando el resultado es negativo.
- b) Se pone a 1 cuando el resultado es positivo.
- c) Se pone a 1 cuando el resultado de una operación es 0.
- d) Se pone a 0 cuando el resultado es negativo.

7. Los procesadores comerciales con unidad de control microprogramada suelen almacenar los microprogramas...

- a) en una RAM.
- b) en un banco de registros.
- > c) en una ROM.
- d) en una PLA.

8. Respecto a MBR y MAR

- a) Ambos permiten guardar información sobre el marco de pila //sin relación directa
- b) Ambos son accesibles por el programador //ninguno
- c) MAR contiene el dato/instrucción que se leerá o escribirá en memoria //ese es MBR
- > d) MAR requiere menos señales de control que MBR //en Tema 3 tr.10, sólo una (LoadMAR), //frente a 4 de MBR (Load/Enable Mem/Bus)

9. Un computador usa el formato vertical de codificación de instrucciones para parte de las señales de control y el formato horizontal para k señales de control. El formato vertical posee n campos codificados de m bits cada uno. ¿Cuál es el máximo número de señales de control que pueden usarse en este computador?

- a)  $k + n \cdot 2^m$
- b)  $k + n^m$
- > c)  $k + n \cdot (2^m - 1)$
- d) Ninguno de los anteriores

10. Un computador tiene una memoria de control de 640 palabras de 70 bits, de las que 280 son diferentes. ¿Qué ahorro en número de bits obtendríamos si usáramos nanoprogramación en lugar de microprogramación?

- >a) 19440
- b) 42280

c) 9840

d) ninguno de los anteriores resultados es exacto.

11. Un computador tiene una memoria de control de 16000 palabras de 250 bits, de las que 447 son diferentes. ¿Cuántos bits ahorramos usando nanoprogramación en lugar de micropogramación?

-> a) 3744250

b) 259206

c) 287935

d) ninguno de los resultados anteriores es exacto

12. En un camino de datos con un solo bus, para realizar la operación de copia de un registro r1 en un registro r2, es decir  $r2 \leftarrow r1$ , es necesario:

a) Activar la carga del registro r1 y habilitar la salida triestado del registro r2

b) Habilitar la salida triestado del registro r2 y activar la carga de los registros r1 y r2

c) Habilitar las salidas triestado de los registros r1 y r2 y activar la carga del registro r2

-> d) Habilitar la salida triestado del registro r1 y activar la carga del registro r2

13. Respecto a las unidades de control nanoprogramadas:

-> a) La anchura de la memoria de nanoprograma es la misma que la de memoria de micropograma en un diseño de la misma unidad de control que no usara nanoprogramación.

b) La realización nanoprogramada de una unidad de control es más rápida que la micropogramada.

c) El diseño de las unidades de control nanoprogramadas debe ser vertical.

d) Suponiendo una memoria de micropograma con n microinstrucciones de w bits cada una, de las cuales  $2^m$  son distintas, el ahorro en bits si se utiliza nanoprogramación es  $(n \cdot m + 2^m \cdot w) - n \cdot w$ .

14. En el pseudocódigo usado para representar las microinstrucciones, la expresión “goto f(IR)”:

a) Permite saltar a la dirección de memoria de control del principio de un microbucle.

b) Se utiliza para realizar un microsalto condicional en función del registro de estado.

c) Realiza una llamada a una microsubrutina.

-> d) Salta a una dirección de memoria de control que depende de la instrucción máquina actual.

15. Para conectar las salidas de dos registros hacia un bus común en el datapath...

- > a) se pueden usar dos buffers triestado.
- b) no se puede usar un multiplexor.
- c) se puede realizar una conexión directa.
- d) se puede usar un demultiplexor.

16. Un procesador con una unidad de control microprogramada tiene una memoria de control de 300 palabras de 100 bits, de las que 200 son diferentes. Si se rediseñara como unidad de control nanoprogramada, ¿qué tamaño ocuparía la nanomemoria que contiene las microinstrucciones completas sin repeticiones?

- a) 30000 bits
- > b) 20000 bits
- 200 uinstr. x 100 bits
- c) 21600 bits
- d) 22400 bits

17. ¿Cuál de las siguientes características es típica de la microprogramación horizontal?

- a) Muchos campos solapados.
- b) Escasa capacidad para expresar paralelismo entre microoperaciones.
- c) Microinstrucciones cortas.
- > d) Ninguna o escasa codificación.

18. ¿Cuál de las siguientes funciones no corresponde a la unidad de control de un procesador?

- > a) Calculo de operaciones de coma flotante
- b) Decodificación de las instrucciones
- c) Secuenciamiento de las instrucciones
- d) Generación de las señales de control que provocan la ejecución de cada instrucción

19. Para el procesador con unidad de control microprogramada estudiado en clase, Tanenbaum define 15 codops de 4bits para instrucciones con un operando dirección de 12bits (LODD M, STOD M, ADDD M, ...), y en lugar de gastar el último codop 1111 en otra instrucción con una dirección, realiza una extensión de codop de 3bits para definir ocho instrucciones más. En general esas instrucciones no tienen ningún operando, salvo INSP Y (sp+=Y) y DESP Y (sp-=Y), que tienen un operando inmediato.

a) Y es un operando de 8 bits y no sobra espacio de codificación (huecos) para modificaciones sin aprovechar bit8

-> b) Y es un valor de 8 bits pero podría definirse que fuera de 9 bits sin ninguna dificultad

c) Y es una dirección corta de 9 bits y podría definirse que fuera de 10 u 11 bits cambiando la codificación por extensión

// 11bits imposible, sólo cabrían INSP/DESP, no habría sitio para PUSH/POP etc

d) Y ocupa 10 bits, pero podría ser de 11 bits cambiando la codificación por extensión y uniendo ambas instrucciones en una sola ADDSP Y (sp+=Y) en donde Y se interpretara como número con signo en complemento a dos

// Y no ocupa 10bits.

//Sí que sería posible ADDSP codop 1111 1yyy yyyy yyyy con 11bit para Y, y entonces codop //1111 0ccc 0000 0000 permite de sobra los 6 codops restantes. [T3.4Tnbaum] tr.79

20. Una unidad de control microprogramada con secuenciamiento explícito con dos direcciones por microinstrucción, tiene una memoria de control con 35 bits de longitud de palabra. Si las microinstrucciones emplean 15 bits en total para los campos de control y de tipo y condición de salto, el número máximo de palabras de la memoria de control de esta unidad de control microprogramada es de:

a) 10

b)  $2^{20}$

-> c)  $2^{10}$

d) 20

21. ¿Qué circuito suele utilizarse para traducir el código de operación de una instrucción máquina a dirección de comienzo en la memoria de control del microprograma correspondiente?

-> a) Una memoria.

b) Un registro.

c) Un multiplexor.

d) Un contador.

22. En una arquitectura RISC típica:

a) no puede usarse segmentación.

b) la programación resulta mucho más simple que en una arquitectura CISC.

-> c) se usa un porcentaje elevado de las instrucciones del repertorio.

d) la UC es más compleja que en una arquitectura CISC.

//Evitar esta pregunta (de la antigua titulación), las transparencias que hablan de RISC están //ahora repartidas por los distintos temas

23. Sea un formato de microinstrucción que incluye dos campos independientes de 10 bits cada uno. Si se rediseña de modo que se solapen los dos campos, ¿cuántos bits se ahorran en cada microinstrucción?

a) 14

b) 13

c) 10

->d) 9

24. La salida de un campo del registro de microinstrucción que solapa dirección de salto y algunas señales de control han de conectarse a:

a) una ROM o PLA

->b) un demultiplexor controlado por el tipo de salto

c) el registro de instrucción

d) la memoria de control

25. Una unidad de control microprogramada se denomina "con secuenciamiento de microinstrucciones explícito" según tenga o no tenga

a) ROM/PLA para traducir el codop en dirección de inicio de microprograma (goto f(IR))

//todos los diseños implícitos en las transparencias usan ROM traducción, pero también se

//podría añadir ROM al explícito

b) microcódigo de decodificación que analice el codop bit a bit de izquierda a derecha

//con ROM traductora (goto f(IR)) no haría falta ese análisis

-> c) micro-contador de programa atacando a las líneas de dirección de la memoria de control

// si y sólo si tiene micro-PC, sería implícito

d) un multiplexor para seleccionar la fuente de la dirección de la memoria de control

//todos los diseños en las transparencias tienen un MUX en la dirección de la memoria de

//control (o en el micro-PC)

26. ¿Cuál de las siguientes afirmaciones es verdadera?

-> a) La unidad de control necesita como entrada el registro de estado para poder controlar la

ejecución de las instrucciones de salto condicional.

- b) El registro puntero de pila es un registro de propósito general que suele contener tanto direcciones como datos.
- c) Las únicas instrucciones en las que algunas de sus fases de ejecución conllevan un acceso a memoria son las instrucciones load y store.
- d) El registro de instrucción es un registro de propósito específico que contiene la dirección de la siguiente instrucción a ejecutar.

27. En una unidad de control microprogramada con formato de microinstrucciones vertical, un subcampo que deba especificar 16 señales de control codificadas de tal forma que pueda activarse sólo una o ninguna habrá de tener una anchura mínima de

- a) 17 bits
- b) 16 bits
- > c) 5 bits
- d) 4 bits

28. Para realizar la microoperación MAR <- PC, habrá que activar:

- a) EnPC y EnMAR
- b) LdPC y EnMAR
- > c) EnPC y LdMAR
- d) LdPC y LdMAR

29. Una posible codificación en microinstrucciones de la instrucción CALL X es:

- a) SP=SP-1 ; m[SP]=PC ; PC=PC+1
- > b) SP=SP-1 ; m[SP]=PC ; PC=X
- c) PC=X ; SP=SP-1 ; m[SP]=PC
- d) SP=PC-1 ; m[SP]=PC ; PC=X

30. Dado un camino de datos concreto, un posible formato de microprogramación se caracteriza como horizontal o vertical según tenga más o menos (señalar la respuesta falsa)

- a) longitud relativa de microinstrucción
- > b) microbifurcaciones
- c) solapamiento
- d) codificación

31. Alguna de las siguientes \*no\* es una operación básica de la unidad de control:

a) realizar operación ALU y guardar resultado en registro

b) transferir un registro a otro

-> c) (guardar o recuperar) un registro (en / de) la pila

d) (leer o escribir) un registro (de / a) memoria

32. Un computador tiene una memoria de control de 16 Kpalabras de 250 bits, de las que

447 son diferentes. ¿Cuántos bits ahorraremos usando nanoprogramación en lugar de

micropogramación?

a) 3928652

b) 259206

c) 287935

-> d) ninguno de los resultados anteriores es exacto

33. Un procesador con una unidad de control micropogramada tiene una memoria de

control de 340 palabras de 16 bits, de las que 180 son diferentes. ¿Qué ahorro en número de

bits obtendríamos si usáramos nanoprogramación?

a) 19440 bits

b) 2560 bits

c) 5260 bits

-> d) No se produce ahorro

34. Una instrucción máquina puede desglosarse en las siguientes operaciones elementales:

$sp := sp - 1$ ;  $m[sp] := pc$ ;  $pc := x$

Probablemente se trate de una instrucción de:

a) almacenamiento local

-> b) llamada a subrutina

c) apilamiento

d) carga local

35. ¿Cuál de los siguientes grupos de señales son entradas a la unidad de control?

a) Las señales de carga/INCREMENTO/desplazamiento de registros

b) Las señales de selección de entradas de multiplexores del datapath

-> c) Los bits del registro de indicadores (flags)

d) Los bits de las opciones b y c

36. Respecto a los términos microinstrucción y microcódigo:

- a) Son equivalentes, llamamos microcódigo o microinstrucción a una palabra de la memoria de control
  - b) Una microinstrucción está programada en microcódigo, que es un lenguaje para programar señales de control
  - c) Un microcódigo controla una serie de señales de control relacionadas (por ejemplo, el código 000 para que la ALU realice la suma), y varios juntos forman una microinstrucción
- > d) Microcódigo es el contenido de la memoria de control, y una microinstrucción es una palabra de dicha memoria

37. El control residual se utiliza para:

- a) reducir el tiempo de ejecución de las instrucciones máquina
- b) eliminar los bits residuales de la ejecución de los microinstrucciones
- > c) reducir el tamaño de la memoria de control
- d) ninguna de las anteriores es cierta

38. Son funciones de la unidad de control:

- a) la codificación de las instrucciones máquina
- b) la lectura de memoria principal de la instrucción apuntada por el  $\mu$ PC
- > c) el secuenciamiento de las instrucciones máquina
- d) todas las respuestas son ciertas

39. En cuanto al control microprogramado:

- a) se guardan en una ROM las instrucciones máquina del conjunto de instrucciones.
- b) la UC se construye con puertas lógicas, biestables, etc.
- > c) se usa en CISC para facilitar el diseño de la UC tan compleja.
- d) la lentitud en la ejecución de las instrucciones máquina la impone directamente la tecnología hardware usada.

40. Alguna de las siguientes señales no sirve de entrada a la unidad de control. ¿Cuál?

- a) señal de reloj (CLK)
- b) instrucción actual (bits del registro IR)
- c) estado de la unidad de proceso (flags Z, S, C, O...)
- > d) contador de programa (bits del registro PC)

41. [T3.1] ¿Cuál de las siguientes funciones es una tarea propia de la unidad de control en la CPU?

- > a) decodificar las instrucciones del programa
- b) almacenar instrucciones del programa
- c) realizar operaciones lógicas
- d) almacenar datos del programa

42. En una arquitectura RISC típica:

- a) se usan pocas instrucciones de las disponibles en el conjunto de instrucciones
- > b) suele usarse segmentación
- c) la programación resulta mucho más simple que en una arquitectura CISC
- d) la UC es más compleja que en una arquitectura CISC

43. ¿Cómo actúa el indicador Z del registro de indicadores de estado?

- a) Se pone a 1 cuando el resultado es negativo
- b) Se pone a 0 cuando el resultado es negativo
- > c) Se pone a 1 cuando el resultado de una operación es 0
- d) Se pone a 1 cuando el resultado es positivo

44. Alguna de las siguientes señales no es salida de la unidad de control. ¿Cuál?

- a) señales de lectura y escritura en memoria (RD, WR)
- > b) dirección de la siguiente microinstrucción (bits del campo DIR o Memoria B de Wilkes)
- c) códigos de selección en multiplexores, decodificadores, ALU, etc (00, 01, 10, 11...)
- d) señales de carga, habilitación y/o desplazamiento de registros (Load, Enable, ShiftL, ShiftR)

45. Sea un formato de microinstrucción que incluye dos campos independientes de 8 bits cada uno. Si se rediseña de modo que se solapen los dos campos, ¿cuántos bits se ahorran en cada microinstrucción?

- a) 1
- b) 8
- c) 9
- > d) 7

46. Sea un formato de microinstrucción que incluye dos campos independientes de 8 bits cada uno. Si se rediseña de modo que se solapen los dos campos, ¿cuántos bits se ahorran en cada microinstrucción?

- a) 1
- >b) 7
- c) 8

d) 9

47. ¿Cuál de las siguientes características sobre RISC es \*FALSA\*?

- a) Para acelerar un procesador RISC se deberían emplear técnicas de segmentación.
- >b) Las instrucciones máquina en un procesador RISC deberían ser complejas y potentes.
- c) La decodificación de las instrucciones debe ser simple: un procesador RISC debería emplear pocos formatos de instrucción.
- d) La unidad de control de un procesador RISC debería ser cableada, no microprogramada.

48. En el contexto de microprogramación, el control residual...

- a) intenta disminuir la cantidad de "bits residuales", usando las técnicas de codificación y/o solapamiento de campos, como opuestas a la microprogramación directa o "inmediata".
- b) se refiere a que cuanto más codificación y/o solapamiento se use, menos capacidad para expresar paralelismo se tiene, siendo ese menor control un "residuo" o consecuencia no deseada de dichas técnicas no "inmediatas"
- c) clasifica las microinstrucciones del microcódigo según formen parte de microprogramas ("microinstr. inmediatas") o no ("microinstr. residuales")
- >d) consiste en almacenar señales de control en un "registro de control residual" para usarlas en ciclos posteriores, a diferencia del "control inmediato", en donde los bits se utilizan inmediatamente

49. En una unidad de control microprogramada se tiene un campo de 14 señales de control de las cuales se activarían dos o menos en un ciclo de reloj, nunca tres o más en el mismo ciclo de reloj, y cuando se activan dos una es del grupo 1-7 y otra es del grupo 8-14, nunca las dos del mismo grupo. Sería entonces posible...

- a) codificarlas con 4 bits, y sobraría un código que quedaría sin uso
- >b) codificarlas con 6 bits (2 códigos de 3 bits), y no sobraría ningún código sin uso
- tr.47 dibujo inferior
- c) solaparlas en un solo campo de 5 bits, ahorrando por tanto 9 bits
- d) solaparlas en un solo campo de 7 bits, ahorrando por tanto 6 bits

50. Para el procesador con unidad de control microprogramada estudiado en clase, Tanenbaum propone codificar los N registros y añadir una señal "PERC" para habilitar la carga desde el bus C (recordar que era un diseño típico con 3 buses) y así no perder expresividad/paralelismo. Si fuera N=8, el ahorro de bits en cada microinstrucción debido a esta técnica es de

- a) 35 bits
- >b) 14 bits
- c) 34 bits
- d) 25 bits

51. ¿Cuál de las siguientes sentencias sobre la unidad de control es FALSA?

- a) Cuanto más horizontal es la microprogramación, más largas son las microinstrucciones
- b) Debido al pequeño número de operaciones simples, la sección de control de un procesador RISC puede ser cableada en lugar de microprogramada
- >c) El programador de lenguaje ensamblador necesita conocer la microarquitectura del ordenador
- d) Es posible realizar el diseño físico de una unidad de control cableada de manera semiautomática

52. ¿A qué instrucción de lenguaje máquina podría corresponder la siguiente secuencia de microinstrucciones del camino de datos con un bus estudiado en clase?

Enable R1, Load Y

Enable R2, Select Y, Add, Load Z

Enable Z, Load R2

a) load (R1,R2), R2

->b) add R1, R2

c) store R1+R2, (R2)

d) move Y(R1,R2), Z(R2)

53. Las microoperaciones de la fase de captación de una instrucción:

->a) Son comunes para todas las instrucciones

b) Dependen del código de operación de la instrucción que se encuentra en el registro de instrucción

c) Dependen de los indicadores de estado y del código de operación de la instrucción que se encuentra en el registro de instrucción

d) Dependen del valor del contador de programa

54. En una unidad de control microprogramada se tienen dos campos de 5 señales de control mutuamente exclusivos, de manera que nunca se activarían señales de ambos campos en el mismo ciclo de reloj. Con esas 10 señales sería entonces posible...

a) codificarlas con 4 bits, y sobraría un código que quedaría sin uso

b) codificarlas con 5 bits, y no sobraría ningún código sin uso

->c) solaparlas en un solo campo de 5 bits, ahorrando por tanto 4 bits

tr.48

d) solaparlas en un solo campo de 6 bits, ahorrando por tanto 5 bits

55. Un computador usa el formato vertical de codificación de instrucciones para parte de las señales de control y el formato horizontal para k señales de control. El formato vertical posee n campos codificados de m bits cada uno. ¿Cuál es el máximo número de señales de control que pueden usarse en este computador?

- a)  $k + n \cdot 2^m$
- b)  $k + n^m$
- >c)  $k + n \cdot (2^m - 1)$
- d) Ninguno de los anteriores

56. En una unidad de control microprogramada, bits del registro IR direccionan una ROM cuya salida puede cargarse en el registro  $\mu$ PC (micro-PC). Esa unidad de control...

- a) puede realizar la micro-operación  $IR = \mu$ PC
- >b) dispone de la funcionalidad “goto  $f(IR)$ ”  
sí, puede cargar  $\mu$ PC según ROM
- c) puede realizar la  $\mu$ -op.  $IR = ROM[MAR]$
- d) es una UC con secuenciamiento explícito

## TEMA 4

1. Cuando dos o más instrucciones necesitan un recurso hardware en el mismo ciclo, se trata de un riesgo:

- > a) estructural
- b) de control
- c) por dependencia de datos
- d) de salto

2. La ganancia en velocidad ideal de un cauce de K etapas de igual duración T ejecutando un programa de N

instrucciones es

- > a)  $S = KN/(K+N-1)$
- b)  $S = KN/(K-N+1)$
- c)  $S = NT/(N+K-1)T$
- d)  $S = NKT/(N-K+1)T$

3. Un procesador de 1GHz tarda 4ns en realizar 4 instrucciones sin realizar segmentación de cauce. ¿Cuánto

tardaría en realizar 9 instrucciones una versión de dicho procesador con segmentación de cauce de 4 etapas si no

existiera ningún retraso en ninguna de las instrucciones?

- a) 2 ns
- b) 9 ns
- > c) 3 ns

1ns para que acabe la 1<sup>a</sup> instrucción, y las restantes 8 van saliendo a cada 1ns/4etapas, o sea que duran otros 2ns

- d) 4.5 ns

4. Tipos de riesgos que hemos estudiado en cauces segmentados (señalar la opción incorrecta)

- a) riesgos de control
- b) riesgos estructurales
- > c) riesgos de transferencia
- d) riesgos de (dependencia de) datos

5. Respecto a la predicción de saltos, alguna de las siguientes afirmaciones es falsa

- a) si se toma la misma decisión para cada tipo de instrucción, se trata de "predicción estática"

b) si la predicción cambia según la historia de ejecución del programa, se trata de "predicción dinámica"

-> c) para predicción estática, es conveniente decidir que los saltos hacia adelante siempre se cumplen, y hacia atrás

no

en los bucles se fallaría siempre (salvo la última iteración)

d) para predicción dinámica, existen, entre otros, algoritmos de dos o cuatro estados, que requieren 1 o 2 bits por

instrucción

6. Un sistema no segmentado tarda 20 ns en procesar una instrucción. Las instrucciones pueden ser procesadas en

un cauce (pipeline) de 4 segmentos con un ciclo de reloj de 5 ns. Cuando se procesan muchas instrucciones, la

ganancia máxima de velocidad que se obtiene se acerca a:

a) 80

b) 5

c) 2,5

-> d) 4

7. Una cola de precaptación sirve para:

a) Disminuir el periodo de ciclo del cauce

b) Resolver ciertos problemas de dependencia de datos

c) Aumentar el número de etapas del cauce

-> d) Reducir el efecto de los fallos de cache

8. Si representamos la fase Decode con una D, Execute con una E, Fetch con una F y Writeback con una W, el

orden correcto de las distintas fases de una instrucción máquina es:

a) F W D E

b) D E F W

c) D F E W

-> d) F D E W

9. En la secuencia de instrucciones siguiente, siendo el primer registro el destino, ¿cuántos riesgos se dan?

sub r2,r1,r3

or r8,r6,r2

-> a) Un riesgo por dependencia de datos

la resta calculada en r2 debe estar disponible para el or con r6

b) Dos riesgos por dependencia de datos y uno de control

c) Un riesgo estructural

d) Un riesgo estructural y dos por dependencia de datos

10. Si un procesador no segmentado necesita 5 ns para leer una instrucción de memoria, 2 ns para decodificar la

instrucción, 3 ns para leer del banco de registros, 3 ns para realizar el cálculo requerido por la instrucción, y 2 ns

para escribir el resultado en el banco de registros, ¿cuál es la frecuencia de reloj máxima del procesador?

-> a) 66,67 MHz

b) 500 MHz

c) 200 MHz

d) 40 MHz

11. La técnica de "adelanto de registros" (register forwarding) en un cauce segmentado se usa para limitar el

impacto de los riesgos...

a) de control

b) organizativos

-> c) (por dependencias) de datos

d) estructurales

12. Respecto a la segmentación:

a) Cuando el número de instrucciones ejecutadas tiende al número de etapas de un procesador segmentado, la

ganancia máxima que se puede obtener tiende a infinito

b) Cuando el número de instrucciones ejecutadas en un procesador segmentado crece, la ganancia máxima que se

puede obtener tiende a 1

c) Cuanto más parecidos sean el tiempo de ejecución de una instrucción sin segmentar y el tiempo de una etapa en

el procesador segmentado, mayor será la ganancia máxima que se puede obtener

-> d) Cuanto mayor sea la relación entre el tiempo de ejecución de una instrucción sin segmentar y el tiempo de una

etapa en el procesador segmentado, mayor será la ganancia máxima que se puede obtener

13. Respecto a la segmentación, ¿cuál de las siguientes afirmaciones es falsa?

-> a) Retrasar la fase de decisión saltar/no saltar de las instrucciones de salto condicional contribuye a mejorar el

rendimiento del procesador

b) La técnica de register forwarding habilita una serie de caminos (buses) que se añaden al cauce para permitir que

los resultados de una etapa pasen como entradas a la etapa donde son necesarias

c) Cuantas más etapas tenga un cauce, más instrucciones se estarán ejecutando en distintas fases y más

posibilidades se presentan de que existan riesgos entre ellas

d) La reorganización del código y la introducción de instrucciones nop permite evitar dependencias de datos

15. Un sistema no segmentado tarda 20 ns en procesar una instrucción. Las instrucciones pueden ser procesadas

en un cauce (pipeline) de 4 segmentos con un ciclo de reloj de 5 ns. Cuando se procesan muchas instrucciones, la

ganancia máxima de velocidad que se obtiene se aproxima a:

a) 0,25

b) 5

c) 20

-> d) 4

16. Un sistema no segmentado tarda 200 ns en procesar una instrucción. Las instrucciones pueden ser procesadas

en un cauce segmentado de 20 etapas con un ciclo de reloj de 12 ns. Cuando se procesan muchas instrucciones, la

máxima ganancia de velocidad que podría obtenerse se acerca a:

-> a) 16,67

b) 1,2

c) 20

d) 1,667

17. Los riesgos de datos consisten en que...

a) dos instrucciones acceden a la vez al mismo dato

-> b) una instrucción necesita un dato calculado por otra anterior

- c) dos instrucciones necesitan leer el mismo dato
- d) todas las respuestas anteriores son correctas

18. Las instrucciones de salto...

- a) son las causantes de los riesgos de tipo RAW y WAW.
- > b) complican el diseño eficiente de los procesadores segmentados.
- c) siempre utilizan direccionamiento absoluto.
- d) Todas las afirmaciones anteriores son ciertas.

19. Un sistema no segmentado tarda 20 ns en procesar una instrucción. Las instrucciones pueden ser procesadas

en un cauce (pipeline) de 4 segmentos con un ciclo de reloj de 6 ns. Cuando se procesan muchas instrucciones, la

ganancia máxima de velocidad que se obtiene se aproxima a:

- a) 4
- b) 5
- > c) 3,33
- d) 0,3

20. La precaptación (cola de instrucciones) está relacionada con...

- a) Los riesgos de control (intenta determinar de antemano el flujo de control)
- > b) Los riesgos estructurales (intenta evitar el efecto de un fallo de cache)
- c) Los riesgos de transferencia (intenta agrupar las posibles transferencias de un conjunto de instrucciones)
- d) Los riesgos de (dependencia de) datos (intenta que el dato esté disponible anticipadamente)

21. Alguno de los siguientes NO es un motivo de que no se alcance la ganancia ideal en un cauce segmentado

-> a) La emisión múltiple (y posiblemente desordenada) de instrucciones

T4 tr.19

- b) El propio coste de la segmentación (carga de los registros de acople, etc...)
- c) Los riesgos (hazards)
- d) La duración del ciclo de reloj impuesta por la etapa más lenta

22. Respecto a los conceptos de procesamiento segmentado y superescalar, una de las siguientes afirmaciones es

falsa

-> a) en cualquier procesador resulta ventajoso usar una cola de instrucciones, pero es más importante para uno

segmentado (fundamental) que para uno superescalar (conveniente)

Tema 4 tr.46

b) implícitamente, se presupone que un procesador superescalar emitirá más de una instrucción por ciclo

c) idealmente, con el segmentado se intenta ejecutar una instrucción por ciclo, y con el superescalar más de una por

ciclo (al combinarlo con segmentado)

d) por definición, un procesador superescalar debe tener varias unidades funcionales (más de una)

23. ¿Cuál de las siguientes afirmaciones es cierta?

a) Al realizar la segmentación de cauce aumenta en general el tiempo necesario para la ejecución de un programa

b) Debido a que pueden existir dependencia de datos, los resultados de un programa pueden ser diferentes a si el

programa se ejecutara sin segmentación

c) La segmentación de cauce disminuye el número de instrucciones necesarias para la ejecución de un programa

-> d) Ninguna de las afirmaciones anteriores

24. En la técnica de salto retardado:

a) El salto se realiza varios ciclos antes de la instrucción de salto

-> b) El compilador puede reorganizar el código para llenar los huecos de retardo con instrucciones útiles

c) Las instrucciones en los huecos de retardo se ejecutan unas veces y otras no

d) El compilador no puede insertar operaciones NOP en los huecos de retardo

25. La predicción de saltos está relacionada con...

a) Los riesgos de (dependencia de) datos (intenta que el dato esté disponible anticipadamente)

b) Los riesgos de transferencia (intenta agrupar las posibles transferencias de un conjunto de instrucciones)

c) Los riesgos estructurales (intenta evitar el efecto de un fallo de cache)

-> d) Los riesgos de control (intenta determinar de antemano el flujo de control)

26. Un procesador está segmentado en las etapas F, D, E, M y W. Cada una de ellas consume un tiempo t. La

aceleración ideal (si no hay riesgos) al ejecutar n instrucciones respecto a un procesador no segmentado será:

- > a)  $5n / (4+n)$
- kn/(k+n-1), con k=5
- b)  $(5+n) / 4t$
- c)  $(4+n) / 5t$
- d)  $4n / (5+n)$

27. Un sistema no segmentado tarda 10 ns en procesar una instrucción. Las instrucciones pueden ser procesadas

en un cauce (pipeline) de 5 segmentos con un ciclo de reloj de 4 ns. Cuando se procesan muchas instrucciones, la

ganancia máxima de velocidad que se obtiene se acerca a:

- a) 50
- b) 5
- > c) 2,5
- d) 4

28. Motivos que impiden que la ganancia (aceleración) de un cauce segmentado sea ideal (señalar la respuesta

- falsa)
- a) fragmentación desigual (duración desigual de etapas)
  - > b) cola de instrucciones (precaptación)
  - T4 tr.19
  - c) riesgos (hazards)
  - d) registros de acople (coste de la segmentación)

29. La segmentación de cauce...

- a) permite ejecutar varias instrucciones concurrentemente
- b) acelera la ejecución de un programa
- c) provoca riesgos debido a datos
- > d) todas las respuestas son ciertas

30. Un sistema no segmentado tarda 20 ns en procesar una instrucción. Las instrucciones pueden ser procesadas

en un cauce (pipeline) de 4 segmentos con un ciclo de reloj de 5 ns. Cuando se procesan muchas instrucciones, la

ganancia máxima de velocidad que se obtiene se acerca a:

- a) 0,25
- b) 5
- c) 20
- > d) 4

31. ¿Cuál de las siguientes afirmaciones sobre la segmentación de cauce es cierta?

- a) El CPI de un cauce superescalar es siempre 1 o menor que 1
- > b) En general, una operación segmentada ("pipelined") requiere el mismo tiempo o más, desde el principio hasta el fin, que la misma operación en una implementación no segmentada
- c) La predicción de saltos es una técnica para minimizar los riesgos de datos
- d) Un cauce ("pipeline") de instrucciones inicialmente vacío y con 3 etapas tardará siempre 5 ciclos de reloj en ejecutar 3 instrucciones si cada una de ellas utiliza las 3 etapas

32. ¿Cuál de los siguientes modos de direccionamiento es menos preferible para un procesador de 32 bits y con

- tamaño de instrucción de 32 bits?
- a) indexado
  - b) registro
  - c) indirecto a través de registro
  - > d) directo (o absoluto)

Los otros 3 están indicados como preferibles en Tema 4 tr.42. Si una instrucción ocupa 32bits, no hay sitio ni para el

codop si la propia dirección (del argumento con direccionamiento directo) ocupa de por sí los 32bits.

33. Si el proceso de empaquetado de un producto (50 segundos de duración) puede segmentarse en 5 etapas,

cada una de 10 segundos, de modo que 5 operarios puedan trabajar cada uno en una etapa, ¿cuál de las siguientes

afirmaciones es falsa al aplicar la segmentación?

- a) La preparación completa de cada producto sigue requiriendo 50 s (igual que con una sola persona empaquetando los productos)

-> b) Cada 50 s saldrá un nuevo producto empaquetado, el mismo tiempo que cuando no había cadena de

empaqueamiento

c) La ganancia de velocidad (aceleración) con la segmentación es de 5

d) Una vez en funcionamiento la segmentación, se tardará 100 segundos en tener 10 productos empaquetados,

mientras que con un solo operario se tardaría 500 segundos.

34. Sobre la segmentación:

-> a) Existen limitaciones al rendimiento provocadas por las instrucciones de salto y por las dependencias de datos.

b) La frecuencia de reloj viene impuesta por la etapa más corta.

c) Es una técnica para comenzar simultáneamente la ejecución de varias instrucciones con el fin de reducir el tiempo

de ejecución.

d) Un procesador superescalar no puede estar segmentado.

35. Respecto al salto retardado y al salto anulante, ¿cuál permite que se ejecute la siguiente instrucción, y cuál no?

a) el retardado la ejecuta sólo si no se cumple la condición de salto, el anulante no la ejecuta nunca

b) el retardado la ejecuta sólo si se cumple la condición de salto, el anulante sólo si no se cumple

-> c) el retardado la ejecuta siempre, el anulante la ejecuta sólo si se cumple la condición de salto

T4 tr.38

d) el retardado ejecuta la siguiente instrucción (con el correspondiente retraso), el anulante no la ejecuta (de hecho

la anula)

36. ¿Cuál de los siguientes modos de direccionamiento es \*menos\* preferible para un procesador con

segmentación de cauce?

-> a) Indirecto a través de memoria

b) Indirecto a través de registro

c) Registro

d) Indexado (o relativo a base, o base+índice)

37. En un procesador con segmentación de cauce, aumentar el número de etapas (p.ej. de 2 a 4, o de 4 a 8), tiene

en general como consecuencia:

- > a) Un incremento de las prestaciones
- b) Un mayor retraso en la ejecución de los programas debido al incremento del número de etapas
- c) Una disminución de la máxima frecuencia de reloj a la que puede operar el cauce
- d) Una disminución en la posible dependencia de datos

38. Un sistema no segmentado tarda 20 ns en procesar una instrucción. Las instrucciones pueden ser procesadas

en un cauce (pipeline) de 4 segmentos con un ciclo de reloj de 6 ns. Cuando se procesan muchas instrucciones, la

ganancia máxima de velocidad que se obtiene se aproxima a:

- a) 4
- b) 0,3
- c) 5
- > d) 3,33

39. Un procesador está segmentado en cinco etapas. Cada una de ellas consume un tiempo t. La aceleración ideal

(si no hay riesgos) al ejecutar n instrucciones respecto a un procesador no segmentado será:

- > a.  $5n / (4+n) // T4 \text{ tr.18}$ , kn/(k+n-1), con k=5
- b.  $(4+n) / 5t$
- c.  $4n / (5+n)$
- d.  $(5+n) / 4t$

40. Suponer un procesador tipo MIPS segmentado en etapas IF, ID, EX, MEM, WB, como el presentado de ejemplo en clase. ¿Cuál afirmación es correcta?

- a) La aceleración del procesador no segmentado es 5x
- b) El periodo de reloj se escoge de acuerdo a la etapa más rápida del cauce
- c) Conviene tener memorias separadas de código y datos, para evitar conflictos entre las etapas IF y WB
- >d) Conviene tener cache que permita acceso en un único ciclo de reloj

41. Suponer que un procesador ideal que ejecuta cada instrucción en T segundos se segmenta en cuatro etapas ideales de duración T/4. ¿Cuál razonamiento es correcto?

- a) Se espera una reducción de prestaciones porque además de ejecutar las instrucciones hay que segmentarlas (coste de la segmentación)
  - b) Se espera una reducción de prestaciones porque la duración del ciclo de reloj vendrá impuesta por la etapa más lenta
  - c) Se espera un aumento de prestaciones debido al efecto de los riesgos (hazards) sobre el avance de las instrucciones en el cauce
- >d) Se espera un aumento de prestaciones debido a que las cuatro etapas solapan su funcionamiento, con una aceleración ideal de 4x

42. Precaptar instrucciones antes de que sean necesarias y almacenarlas en una cola de instrucciones, es una técnica que se usa para...

- >a) evitar cierto tipo de riesgos estructurales
- b) reducir riesgos por dependencias de datos
- c) corregir algunos riesgos de control
- d) calcular las predicciones de saltos

43. Un sistema no segmentado tarda 10 ns en procesar una tarea. La misma tarea puede ser procesada en un cauce (pipeline) de 4 segmentos con un ciclo de reloj de 4 ns. Cuando se procesan muchas tareas, la ganancia máxima de velocidad que se obtiene se acerca a:

- >a) 2.5
- b) 4
- c) 10
- d) 40

44. Un procesador x86 a 4 GHz dispone de 7 unidades de ejecución en paralelo, con 20 etapas de segmentación, y es capaz de emitir (comenzar a ejecutar) 4 instrucciones en cada ciclo de reloj. ¿Qué velocidad aproximada de ejecución de instrucciones será capaz de alcanzar (MIPS = millones de instrucciones por segundo)?

- a) 28000 MIPS
- >b) 16000 MIPS
- c) 4000 MIPS
- d) 80000 MIPS

45. Un salto condicional de tipo "annulling branch", o salto anulante, ejecuta la(s) instrucción(es) siguiente(s)...

- >a) sólo si el salto se produce (las ignora si NO se produce), de manera que instrucción(es) en el destino del salto podrían adelantarse tras la propia instrucción de salto
- b) sólo si el salto NO se produce (las ignora si se produce), de manera que instrucción(es) en el destino del salto podrían adelantarse tras la propia instrucción de salto

c) siempre, de manera que instrucción(es) anterior(es) al salto podrían colocarse tras la propia instrucción de salto

d) nunca, de manera que instrucción(es) anterior(es) al salto podrían adelantarse tras la propia instrucción de salto

46.Respecto a saltos retardados y anulantes, condicionales o no, NO sería apropiado intentar reordenar instrucciones...

a)en el destino del salto condicional anulante para ponerlas después (en mem.) del salto (y avanzando el destino del salto)

eso sí se hace, ver tr.38 (annulling)

b) en el destino del salto incondicional retardado para ponerlas después (en mem.) del salto (y avanzando el destino del salto)

eso sí se hace, ver tr.37 (retardado)

->c) anteriores al condicional anulante para ponerlas después (en memoria)

exacto, no, ver tr.38 (annulling)

d) anteriores al condicional retardado para ponerlas después (en memoria)

eso sí se hace, ver tr.36 (retardado)

## TEMA 5

1. ¿A qué tipo de interrupciones corresponde la forma de determinar la dirección de comienzo de una rutina de interrupción en la que se envía una instrucción de bifurcación completa?

- a) Interrupciones no vectorizadas
- b) Interrupciones con direcciones fijas
- > c) Interrupciones vectorizadas
- d) Interrupciones encadenadas

2. Si se dice que en un sistema computador cada dirección especifica uno o dos puertos de E/S, se refiere a que:

- a) La misma dirección (por ejemplo 0x0210) puede ser una posición de memoria o un puerto de E/S, según IO/M#
- > b) Un puerto será de sólo lectura, otro de sólo escritura, y ambos se decodifican en la misma dirección
- c) La misma dirección puede usarse para transferir un byte o una palabra de mayor tamaño (ese byte y el siguiente)
- d) La pregunta es capciosa, una dirección puede especificar un puerto, no dos

3. ¿Qué técnica de E/S se dice controlada por hardware?

- a) E/S programada
- b) E/S controlada por interrupciones
- > c) Acceso directo a memoria
- d) Ninguna de las anteriores

4. Se dispone de un procesador con una frecuencia de reloj de 1 GHz. Se le conecta un dispositivo que genera 100.000 interrupciones por segundo. La rutina de servicio de interrupción ejecuta 500 instrucciones. El número medio de ciclos por instrucción es 2. ¿Qué porcentaje del tiempo dedica el procesador al dispositivo?

- a) 90%
- > b) 10%
- c) 50%
- d) 1%

5. Señale cuál de las siguientes opciones es una técnica habitual para llevar a cabo la transferencia de datos entre el computador y los dispositivos de E/S externos:

a) E/S por nivel

-> b) Acceso directo a memoria (DMA)

c) E/S por flanco

d) Acceso indirecto a memoria (IMA)

6. El fragmento de código ensamblador de un microprocesador de 8 bits

lds IOBuf ; Apuntar puntero pila a

; ...área mem.intermedia

Idx Count ; Inicializar X-contador

poll Ida a Status; Leer estado en A

bpl poll ; Signo(A)!=1 => repetir

Ida a Data ; Leer dato en A

psh a ; Transferir dato a pila

dex ; Decrementar contador X

bne poll ; Seguir leyendo si X!=0

corresponde a:

-> a) Entrada programada con consulta de estado

Ida a Data -> lectura del puerto de Datos

Ida a Status -> poll (consulta) de estado

b) Entrada programada sin consulta de estado

c) Salida programada sin consulta de estado

d) Salida programada con consulta de estado

7. La instrucción máquina DI (Disable Interrupts), conocida como CLI (Clear Interrupt Flag) en

x86, se utiliza para desactivar:

-> a) Todas las interrupciones enmascarables

Tema 5 tr.50

b) Determinados niveles de interrupción de forma selectiva

c) Las interrupciones de inferior o igual prioridad a una dada

d) Las interrupciones software

8. Técnicas que se pueden usar para determinar la causa de una interrupción (señalar

la opción incorrecta)

a) interrupciones vectorizadas

-> b) línea de reconocimiento INTA#

c) consulta de estado, o polling

d) múltiples líneas de interrupción INT1#, INT2#...

9. [T5.1] Señale cuál de las siguientes opciones no es un modo para llevar a cabo la transferencia de datos entre el computador y los dispositivos de E/S externos:

-> a) E/S por flanco

b) E/S iniciada por interrupción

c) Acceso directo a memoria (DMA)

d) E/S programada

10. ¿Cuál de las siguientes tareas no es responsabilidad de un circuito de interfaz o controlador de periféricos sencillo?

-> a) Ejecutar el programa de transferencia de información entre el procesador y los dispositivos de E/S

b) Adaptar el formato de las señales

c) Recibir señales de control desde el procesador

d) Ajustar la temporización entre el procesador y los dispositivos de E/S

11. Las interrupciones generadas por el teclado interrumpirán al procesador:

a) sólo cuando el procesador no esté realizando un trabajo útil

b) sólo si el procesador está chequeando el estado del teclado

-> c) sólo si el procesador tiene activado el indicador de habilitación de interrupciones

d) siempre que el usuario pulse una tecla en el teclado

12. ¿Cuántos niveles de interrupción podremos gestionar si disponemos de 7 controladores de interrupciones programables 8259?

-> a) 50

ver T5 tr.77:  $1 \text{ master} + 6 \text{ slaves} = 2 + (8 \times 6) = 50 \text{ IRQ}$

b) 45

c) 48

d) 56

13. Sobre la E/S mapeada en memoria podemos decir que:

-> a) Usa direcciones de memoria para acceder a puertos de E/S

ver T5 tr.24

- b) La CPU necesita el pin IO/M#
- c) Dispone de instrucciones especiales de E/S
- d) Todas las respuestas anteriores son falsas

14. Con tres controladores de interrupciones 8259 se pueden manejar exactamente:

- a) 8 niveles de prioridad
  - b) 16 niveles de prioridad
  - c) 24 niveles de prioridad
- > d) Ninguna de las anteriores es cierta

15. Un computador con 15 líneas de direcciones tiene 3 módulos de memoria de  $2^{13}$  palabras y utiliza E/S mapeada en memoria. ¿Cuál es el número máximo de periféricos que pueden conectarse, si cada uno de ellos utiliza 8 direcciones?

- a)  $2^{13}$
  - b)  $2^{11}$
  - c)  $2^{12}$
- > d)  $2^{10}$

16. ¿Cuál de las siguientes funciones no corresponde a un controlador (interfaz) de E/S?

- a) Almacenamiento temporal de datos
- b) Comunicación con el dispositivo
- > c) Almacenamiento de programas
- d) Comunicación con el microprocesador

17. ¿Qué técnica de E/S requiere menos atención por parte del procesador?

- a) Todas requieren la misma atención
- > b) E/S mediante acceso directo a memoria
- c) E/S mediante interrupciones
- d) E/S programada

18. Supongamos dos procesadores con bus de direcciones con idéntico número de líneas. Si uno de ellos emplea E/S mapeada en memoria y el otro E/S independiente, ¿cuál podrá acceder a una mayor cantidad de memoria?

- > a) El que tiene E/S independiente
- b) Depende del tamaño del bus de direcciones
- c) El que tiene E/S mapeada en memoria

d) Ambos podrán acceder a la misma cantidad de memoria

19. ¿En qué tipo de transferencias es necesario establecer un periodo de tiempo máximo después del cual se considera que ha fallado?

a) En las transferencias síncronas

-> b) En las transferencias asíncronas

c) En ambas

d) En ninguna

20. Respecto a los conceptos de procesador de E/S, canal de E/S, dispositivos de E/S:

-> a) Un procesador o canal tiene un repertorio de instrucciones específico para manejar los dispositivos E/S

b) Cada canal es una línea de comunicación entre el procesador y un dispositivo de E/S.

c) Al conjunto de conexiones entre el procesador y los dispositivos se le denomina canal de E/S (de ese ordenador)

d) La pregunta es capciosa, el procesador no es E/S, son otros dos componentes von Neumann distintos (ALU+UC)

21. La E/S programada:

a) Mejora las prestaciones globales del sistema respecto a la E/S por interrupciones porque la CPU tiene el control de toda la operación.

b) Empeora las prestaciones globales del sistema respecto a la E/S por interrupciones porque una

instrucción de transferencia individual de datos con la interfaz del periférico (por ej. IN, OUT) es más lenta en E/S programada que en E/S por interrupciones.

c) Mejora las prestaciones globales del sistema respecto a la E/S por interrupciones porque la CPU es más rápida que el controlador de interrupciones y la interfaz del periférico.

-> d) Empeora las prestaciones globales del sistema respecto a la E/S por interrupciones porque la CPU debe encargarse de la sincronización con la interfaz del periférico haciendo una espera activa.

22. Alguna de las siguientes técnicas NO es de utilidad para determinar la causa de una interrupción

a) Múltiples líneas de interrupción INT1#, INT2#...

b) Consulta de estado, o polling

-> c) Línea de reconocimiento INTA#

d) Interrupciones vectorizadas

23. ¿En qué método para determinar la dirección de comienzo de una rutina de servicio de interrupción se envía parte de dicha dirección?

-> a) Direccionamiento relativo

b) Direccionamiento absoluto

c) Direccionamiento indirecto

d) Direccionamiento mediante instrucción de bifurcación

24. ¿Es posible utilizar 4 GB de memoria en un sistema cuya CPU emplea E/S mapeada en memoria, cuyo bus de direcciones es de 32 bits y que tiene al menos un puerto de E/S?

Supondremos que no se puede emplear ninguna técnica de extensión del bus de direcciones.

a) Sí

-> b) No

c) Depende de si el número puertos de E/S es muy elevado

d) Ninguna de las respuestas anteriores es correcta

25. Respecto a los conceptos de interfaz de dispositivo, controlador(a), puerto de E/S:

-> a) La controladora o interfaz contiene los puertos necesarios para utilizar el dispositivo

b) El puerto, o interfaz, contiene los controladores necesarios para comunicar el dispositivo con el procesador

c) Cada puerto o interfaz es una línea de comunicación con el procesador. El conjunto de ellos forma el controlador.

d) El interfaz contiene las controladoras necesarias para conectar los puertos con el procesador

26. ¿En qué técnica para determinar la dirección de comienzo de la rutina de servicio de interrupción se fija dicha dirección en los circuitos de la CPU?

a) Envío de instrucción de bifurcación completa.

b) Direccionamiento absoluto.

c) Direccionamiento relativo.

-> d) Direcciones fijas.

27. ¿De cuántos canales de E/S independientes dispone el controlador de acceso directo a memoria 8237?

-> a) 4

ver T5 tr.91

b) 8

c) 2

d) 16

28. La consulta de estado que se puede llevar a cabo en la E/S programada sirve para...

-> a) consultar si el dispositivo está dispuesto para recibir datos o tiene datos disponibles

b) consultar qué dispositivo ha solicitado la interrupción

c) consultar si el dispositivo funciona correctamente

d) Ninguna de las respuestas anteriores es correcta

29. ¿Qué técnica de E/S consume menos tiempo del procesador?

a) E/S programada

b) E/S mediante interrupciones

-> c) E/S mediante DMA

d) Todas pueden consumir el mismo número de recursos en función de la velocidad del dispositivo de E/S al que se acceda

30. Con nueve controladores de interrupciones 8259 se pueden manejar exactamente:

a) 8 niveles de prioridad

b) 16 niveles de prioridad

c) 24 niveles de prioridad

-> d) Ninguna de las anteriores es cierta

En cascada, al 8259 maestro se conectan los otros 8, cada uno con 8 IRQs => 64 niveles

31. ¿Cuál de las siguientes afirmaciones acerca de las interrupciones en el PC (modo real) es cierta?

a) Existen 1024 vectores de interrupción.

b) Cada vector de interrupción es una doble palabra de 32 bits formada en primer lugar (dirección menor) por el segmento y seguida por el desplazamiento (dirección mayor) de cada rutina de servicio de interrupción

c) No todas las interrupciones se pueden generar por software

-> d) Ninguna de las anteriores afirmaciones es cierta

32. ¿Cuál de las siguientes afirmaciones acerca del daisy-chain es cierta?

- a) Todos los componentes se conectan directamente y con igual prioridad al procesador o gestor de interrupciones
- > b) Los componentes se comportan de forma cooperativa: sólo al de mayor prioridad se le concede la interrupción o se apodera del bus de comunicaciones
- lo del "bus de comunicaciones" no se explica en las transparencias actuales - el daisy-chain se puede usar también como método de arbitraje para conceder el bus - las señales suelen llamarse BR/BG (Bus request/grant) en lugar de INTR/INTA (IRQ request/acknowledge)
- c) Es incompatible con la técnica de sondeo o polling
- d) Los componentes están conectados todos con todos y un gestor centralizado decide la prioridad

33. Parecidos y diferencias entre los métodos de E/S (señalar la opción incorrecta)

- a) se suele avisar a la CPU (con una IRQ) de que debe realizar alguna tarea, tanto en E/S por IRQ como en E/S por DMA  
cubre los casos de IRQ de dispositivo e IRQ de DMA
- > b) sólo E/S por DMA libera a la CPU de realizar la consulta de estado del dispositivo  
no cubre el caso de E/S programada para dispositivos "sin estado" (ej: display 7 segmentos en las diapositivas)
- c) la consulta del estado del dispositivo por parte de la CPU se suele/puede incluir en E/S programada y en E/S por IRQ
- d) sólo E/S por DMA libera a la CPU de realizar la transferencia de los datos de E/S

34. ¿Cuántas señales de control se necesitan como mínimo para implementar un sistema de gestión de interrupciones?

- a) 3
- b) 4
- > c) 2
- d) 5

35. Se desea utilizar interrupciones vectorizadas para gestionar las interrupciones procedentes de 16 dispositivos. Si tenemos una CPU con un bus de datos de 8 bits, ¿es posible?

- a) Sí, empleando un decodificador
- > b) Sí, empleando un codificador

c) Sí, empleando un multiplexor

d) Ninguna de las respuestas anteriores es cierta

36. Si el acceso directo a memoria se realiza mediante robo de ciclo:

a) es posible que la ejecución de una operación elemental de transferencia en el bus sea

temporalmente detenida

-> b) es posible que la ejecución de una instrucción máquina sea temporalmente detenida

c) es necesario que termine de ejecutarse la instrucción máquina actual para comenzar una transferencia por DMA

d) ninguna de las anteriores es cierta

37. Las técnicas principales de E/S son (señalar la respuesta falsa)

a) DMA (por acceso directo)

-> b) E/S cableada (hardwired)

c) E/S programada

d) IRQ (por interrupciones)

38. ¿Qué método de control de acceso directo a memoria es preferible por velocidad (más rápida), economía (coste no prohibitivo) y conveniencia de diseño (compatible con memorias y sistemas actuales)?

a) DMA intercalado o transparente

lento, para sistemas monoprocesador con sólo 1 maestro bus

b) Robo de ciclo

lento, para sistemas monoprocesador con sólo 1 maestro bus

c) Memoria multipuerto

-> d) Transferencia de bloques o parada de CPU

39. El fragmento de código:

poll: in a, 0x20

cmp a, \$0

jnz poll

load a, 0x11

out 0x21, a

corresponde a:

a) E/S por interrupciones

-> b) E/S programada con consulta de estado

c) E/S por DMA

d) E/S programada sin consulta de estado

40. Algunas de las ventajas de la E/S aislada frente a la E/S mapeada en memoria son:

-> a) Los programas pueden ser más rápidos, y las instrucciones de E/S son fácilmente reconocibles

b) Facilita la protección de E/S, y el diseño de la CPU es menos complejo

c) Las respuestas a y b son ciertas

d) Las respuestas a y b son falsas

41. Al método de interacción con los periféricos, en los que el procesador vigila

periódicamente el estado de los dispositivos mediante una encuesta activa se le denomina:

a) interrupción

-> b) polling

c) daisy-chain

d) DMA

42. Las interrupciones iniciadas por un dispositivo de E/S son normalmente:

a) espurias

-> b) externas

c) software

d) internas

43. Una instrucción típica de entrada / salida tiene

a) tiene un argumento: un registro del procesador

incluso aunque la transparencia 22 sugiera que es "IN puerto" y "OUT puerto", el argumento no es un registro

b) no tiene ningún argumento

c) tiene tres argumentos: un registro del procesador, una dirección de puerto de E/S y una dirección de memoria

-> d) tiene dos argumentos: un registro del procesador y una dirección de puerto de E/S  
claramente inspirado en el repertorio IA32 - no encuentro dónde aparece en las transparencias  
- "Ida a Data" en transparencia 40 es mapeado a memoria - evitar volver a poner esta pregunta en exámenes

44. En la E/S controlada por interrupciones:

- a) La CPU lee y comprueba el estado de los dispositivos de E/S (en el caso de consulta de estado).

poll? obligatorio con IRQ?

- b) El controlador de DMA envía una petición de interrupción a la CPU.

DMA?

- > c) La CPU transfiere el control a una rutina de servicio cuando recibe una interrupción.

Exacto, la ISR

- d) El controlador de DMA transfiere bloques de datos por el bus del sistema.

45. Tipos de interrupción que suelen contemplar las CPUs comerciales actuales (señalar la opción incorrecta)

- a) externas (IRQs hardware): generadas por un dispositivo externo a la CPU, activan la línea INTR# (o equivalente)

- b) software: generadas al ejecutar la instrucción INT (o equivalente)

- c) internas (excepciones o traps): generadas internamente por la CPU para indicar una condición que requiere atención (división por cero, codop inválido, etc)

- > d) firmware (faults): generadas por el microcódigo de la CPU (segmentation fault, page fault, etc)

46. Suponiendo que varios dispositivos comparten una única línea de solicitud de interrupción y que varios de ellos solicitan una interrupción al mismo tiempo, ¿qué dispositivo tendría mayor prioridad a la hora de ser atendidas sus peticiones?

- a) Si se emplea una técnica de sondeo ("polling"), el dispositivo cuyo estado se consulte primero

- b) Si se emplea una técnica de encadenamiento ("daisy-chain"), el dispositivo al que primero se conecta la línea INTA# del procesador

- > c) Las respuestas a y b son ciertas

- d) Las respuestas a y b son falsas

47. ¿Cuál de las siguientes afirmaciones acerca de las interrupciones en el PC (modo real) es cierta?

- a) La tabla de interrupciones tiene un tamaño de 256 bytes

- b) Existen 1024 vectores de interrupción

c) Cada vector de interrupción es una palabra de 16 bits

-> d) Todas las interrupciones se pueden generar por software

48. ¿Es cierto que la consulta de estado permite averiguar el estado de un periférico y alterar la prioridad de los periféricos?

a) No, porque no permite averiguar el estado de un periférico

b) No, porque no permite ni averiguar el estado de un periférico ni alterar la prioridad de los periféricos

c) No, porque no permite alterar la prioridad de los periféricos

-> d) Sí, permite averiguar el estado de un periférico y también alterar la prioridad de los periféricos

49. ¿Cuántos puertos puede gestionar la interfaz de periféricos programable 8255?

a) Uno de 24 bits

b) Dos de 12 bits

-> c) Tres de 8 bits

d) Todas las combinaciones anteriores son válidas

50. ¿Qué método de identificación de la fuente de una interrupción suele ser más económico desde el punto de vista hardware?

-> a) La identificación mediante la técnica de sondeo

b) La identificación mediante interrupciones vectorizadas

c) El coste de las técnicas a) y b) es el mismo

d) No es posible comparar el coste de los métodos a) y b)

51. Respecto al sistema de Entrada / Salida, ¿cuál de las siguientes afirmaciones es incorrecta?

a) La CPU se comunica con el periférico por medio del controlador y de software de E/S.

b) Un protocolo sirve para “ponerse de acuerdo” en cosas como velocidad, paridad, nº de bits, etc.

c) Un controlador se encarga de la comunicación con la CPU.

-> d) La mayoría de los periféricos trabajan a velocidad muy superior a la CPU; por eso es necesario sincronizar.

52. ¿Cuántos bits hacen falta como mínimo para implementar tres niveles de inhibición de interrupciones (general, nivel y máscara) en un sistema con cuatro niveles de interrupción?

a) 5

b) 6

c) 4

-> d) 7

53. ¿Qué tipo de sincronización es más conveniente en el caso de tener dispositivos con distintos requisitos de temporización?

a) Síncrona

-> b) Asíncrona

c) No se pueden conectar dispositivos con distintos requisitos de temporización

d) Ninguna de las anteriores

54. La conexión entre un dispositivo de E/S y el procesador mediante bus:

a) Requiere multiplexores y demultiplexores para las señales de datos

-> b) Permite conectar en paralelo varios dispositivos

c) Requiere mucha circuitería

d) Es difícil de expandir

55. ¿Cuál de las siguientes afirmaciones es cierta?

a) La E/S en memoria emplea la patilla IO/M#

-> b) En E/S independiente, las instrucciones de acceso a memoria suelen ser más largas que las de E/S

c) La E/S en memoria facilita la protección

d) Ninguna de las anteriores es cierta

56. ¿Cuál de las siguientes afirmaciones es incorrecta?

a) Sólo E/S por DMA libera a la CPU de realizar la transferencia de los datos de E/S

b) Se suele avisar a la CPU (mediante una IRQ) de que debe realizar alguna tarea, tanto en E/S por IRQ (obligatoriamente, la tarea es la transferencia) como en E/S por DMA (optativamente, el controlador DMA puede avisar de que acabó)

-> c) Sólo E/S por DMA libera a la CPU de realizar la consulta de estado del dispositivo de E/S

d) La consulta del estado del dispositivo por parte de la CPU se suele hacer con E/S programada (salvo con dispositivos que siempre están listos para transferir) y con E/S por IRQ (cuando se usa polling para determinar el origen de la IRQ)

57. La técnica de sondeo, escrutinio o "polling"...

- a) Es incompatible con el daisy-chain
  - b) No permite establecer un mecanismo de asignación de prioridades a los distintos dispositivos
  - c) En caso de utilizarse, es necesario emplear varias líneas para que los dispositivos soliciten una interrupción
- > d) Se utiliza para identificar la fuente de una interrupción

58. Un computador con 15 líneas de direcciones tiene 3 módulos de memoria de  $2^{13}$  palabras y utiliza E/S mapeada en memoria. ¿Cuál es el número máximo de periféricos que pueden conectarse, si cada uno de ellos utiliza 8 direcciones?

- > a)  $2^{10}$
- b)  $2^{12}$
- c)  $2^{11}$
- d)  $2^{13}$

59. ¿Cuántos controladores de interrupciones 8259 hacen falta como mínimo para manejar 17 líneas de interrupción?

- a) 2
- b) 4
- > c) 3
- d) 5

60. ¿Cuál de los siguientes elementos no forma parte del canal de un controlador de acceso directo a memoria?

- a) Registro de dirección.
- > b) Registro de prioridades.
- c) Registro contador.
- d) Todos los elementos anteriores forman parte de un canal de DMA.

61. Respecto a las técnicas de direccionamiento por selección lineal, decodificación centralizada y distribuida

- a) todas ellas impiden que haya cortocircuito en el bus de datos  
selección lineal no lo garantiza - confía en diseño sensato
- > b) la selección lineal permitiría escribir un mismo dato a varios puertos E/S
- c) todas ellas impiden que haya cortocircuito en el bus de direcciones

El arbitraje se encarga de eso

d) usando decodificación centralizada es más fácil realizar expansiones al sistema de E/S

62. Alguna de las siguientes NO es una técnica de E/S de las estudiadas en clase:

a) E/S mediante Acceso Directo a Memoria

-> b) E/S asíncrona

c) E/S controlada por interrupciones

d) E/S programada

63. Un controlador de DMA suele ser programado con la siguiente información relativa a una operación de E/S:

a) tipo de operación, tamaño de bloque a transferir, dirección final de memoria

-> b) tipo de operación, tamaño de bloque a transferir, dirección inicial de memoria

c) tipo de operación, dirección inicial de memoria, dirección final de memoria

d) Ninguna de las anteriores respuestas es cierta

64. ¿Cuál de las siguientes afirmaciones acerca del concepto de interrupción no es cierta?

a) Es una bifurcación normalmente externa al programa en ejecución

b) Su objetivo es reclamar la atención del procesador

c) Sigue que se ejecute un programa específico para tratarla

-> d) Ninguna de las anteriores

65. ¿Cuántos controladores de interrupciones 8259 hacen falta como mínimo para manejar

25 líneas de interrupción?

a) 1

b) 3

c) 2

-> d) 4

66. ¿Qué modo de funcionamiento permite a la interfaz de periféricos programable 8255

utilizar un bus bidireccional?

a) 3

b) 1

c) 0

-> d) 2

67. La técnica de sondeo, escrutinio o "polling"...

- a) Se utiliza para identificar el destino de una interrupción
- > b) Permite establecer un mecanismo de asignación de prioridades a los distintos dispositivos
- c) En caso de utilizarse, es necesario emplear varias líneas para que los dispositivos soliciten una interrupción
- d) Todas las respuestas anteriores son falsas

68. La transferencia de datos en un computador y los dispositivos de E/S puede manejarse de diversos modos. Uno de los siguientes es falso; indíquelo:

- a) E/S programada
- b) Acceso directo a memoria (DMA)
- c) E/S iniciada por interrupción
- > d) Manejo de todas las líneas del bus de control, paralizando la CPU

69. Respecto a si un computador dispone de E/S independiente (separada) o usa E/S mapeada a memoria:

- > a) Si el repertorio del procesador tiene instrucciones del tipo IN y OUT, es que el computador dispone de E/S separada
- b) Si el encapsulado (chip) del procesador tiene patilla (pin) IO/M# (o patillas equivalentes), eso evidencia que el computador usa E/S mapeada a memoria
- c) Si el encapsulado del procesador no dispone de patilla IO/M# (ni equivalentes), el computador sólo dispone de E/S separada
- d) Si el repertorio del procesador tiene instrucciones del tipo LOAD y STORE, el computador sólo dispone de E/S mapeada a memoria

70. Con una línea de interrupción organizada en colector abierto:

- a) se pueden conectar varios dispositivos de manera sencilla sin necesidad de circuitos combinacionales
- b) a la patilla del circuito integrado que genera esa línea hay que conectarle un transistor
- c) un dispositivo conectado a ella puede solicitar la interrupción poniendo a nivel bajo la línea
- > d) a) y c) son ciertas

71. Un computador con 15 líneas de direcciones tiene 3 módulos de memoria de  $2^{13}$  palabras y utiliza E/S mapeada en memoria. ¿Cuál es el número máximo de periféricos que pueden conectarse, si cada uno de ellos utiliza 8 direcciones?

- a)  $2^{12}$

b)  $2^{13}$

c)  $2^{11}$

-> d)  $2^{10}$

72. Alguna/s de las ventajas de la E/S mapeada en memoria frente a la E/S aislada o independiente es/son:

a) Las instrucciones de E/S son fácilmente reconocibles.

b) Facilita la protección de E/S.

-> c) El diseño de la CPU es más sencillo.

d) Todas las respuestas son ciertas.

73. Supongamos dos CPU con idéntica anchura tanto en el bus de direcciones como en el de datos. Si una de ellas emplea E/S independiente y la otra mapeada en memoria, ¿cuál podrá acceder a una mayor cantidad de memoria?

-> a) La CPU con E/S independiente

b) La CPU con E/S mapeada en memoria

c) Ambas podrán acceder a la misma cantidad de memoria

d) Depende de la técnica de E/S utilizada

74. Respecto a la E/S programada...

a) No todos los pasos requieren la ejecución de instrucciones por parte de la CPU

b) La transferencia la realiza un procesador externo a la CPU

c) Las dos primeras afirmaciones son ciertas

-> d) Las dos primeras afirmaciones son falsas

75. ¿En qué tipos de técnicas de E/S la transferencia de información está bajo el control directo de la CPU?

-> a) E/S programada y E/S controlada por interrupciones

b) E/S programada y acceso directo a memoria

c) E/S controlada por interrupciones y acceso directo a memoria

d) Ninguna de las anteriores respuestas es correcta

76. Respecto a salvaguardar los registros de la CPU al inicio de una rutina de servicio de interrupción (ISR)

a) no es necesario salvar ninguno más, si el contador de programa y los flags de estado ya los salva la propia CPU como parte del mecanismo de interrupción

- b) se deben guardar los registros salva-invocado (p.ej. EBX, ESI, EDI en el caso de una CPU IA32), los registros salva-invocante ya los guarda el programa interrumpido
- > c) se deben guardar los registros que se modifiquen en la propia ISR. Eso es posible hacerlo porque el propio programador de la ISR conoce qué registros va a modificar
- d) se deben guardar todos los registros, para restaurarlos a la salida y así garantizar que el programa interrumpido no sufre ninguna modificación (salvo el inevitable retraso temporal) debido a la interrupción

77. Si varios dispositivos comparten una única línea de solicitud de interrupción:

- a) se podría usar "polling" para identificar el origen de una interrupción
- b) se podría utilizar "daisy-chain" para establecer la prioridad de los dispositivos
- c) se podría utilizar interrupciones vectorizadas
- > d) todas son ciertas

78. Alguna de las siguientes NO es una ventaja de la E/S independiente (separada, aislada)

- > a) Diseño del procesador más sencillo (E/S mapeada añade complejidad al diseño)
- b) Protección de E/S más fácil (E/S mapeada añade dificultad a la protección de E/S)
- c) Mayor aprovechamiento del espacio de memoria (E/S mapeada resta espacio a la memoria)
- d) Decodificación de memoria más elegante, limpia, sencilla (E/S mapeada añade complejidad a la decodificación)

79. Supongamos dos CPU con bus de direcciones de ancho idéntico. Si una de ellas emplea E/S independiente y la otra mapeada en memoria, ¿cuál podrá acceder a una mayor cantidad de memoria?

- > a) La CPU con E/S independiente.
- b) Ambas podrán acceder a la misma cantidad de memoria.
- c) La CPU con E/S mapeada en memoria.
- d) Depende del bus de datos.

80. ¿Con cuál de los siguientes dispositivos tendría sentido utilizar E/S programada sin consulta de estado?

- > a) Salida a un display de 7 segmentos
- b) Entrada desde un disco duro
- c) Salida a una impresora
- d) Con ningún dispositivo tiene sentido

81. Cuando se produce una interrupción hardware...

- > a) Se guarda el estado y se ejecuta la rutina de interrupción asociada
- b) Se aborta la ejecución del programa actual generando un fallo de segmentación
- c) Se salta a la dirección de memoria indicada en la instrucción actual
- d) Se salta al principio del programa actual

82. ¿A qué tipo de interrupciones pertenecen las condiciones de tiempo real y los fallos hardware?

- a) Enmascarables
- > b) No enmascarables
- c) Puede ser tanto enmascarables como no enmascarables
- d) Ninguna de las respuestas anteriores es correcta

83. Para determinar la causa de una interrupción se pueden usar las siguientes técnicas:  
(señalar la respuesta falsa)

- a) consulta de estado, o polling
- b) múltiples líneas de interrupción INT1#, INT2#...
- c) interrupciones vectorizadas
- > d) línea de reconocimiento INTA#

84. Utilizando E/S programada y como modo de direccionamiento selección lineal, ¿cuántos periféricos podrían conectarse a un 8086? Recordar que el 8086 disponía de E/S separada, con bus de direcciones de 20 bits para memoria y de 16 bits para E/S.

- a)  $2^{16}$  periféricos
- > b) 16 periféricos
- c)  $2^{20}$  periféricos
- d) 20 periféricos

85. Ventajas de la E/S independiente (separada, aislada) (señalar la opción incorrecta)

- a) mayor aprovechamiento del espacio de memoria (E/S mapeada resta espacio a la memoria)
- > b) diseño del procesador más sencillo (E/S mapeada añade complejidad al diseño)
- c) decodificación de memoria más elegante, limpia, sencilla (E/S mapeada añade complejidad a la decodificación)
- d) protección de E/S más fácil (E/S mapeada añade dificultad a la protección de E/S)

86. ¿Cuál de los siguientes es un registro de un controlador de DMA?

a) PC (Program Counter)

-> b) WC (Word Count)

c) IR (Instruction Register)

d) SP (Stack Pointer)

87. Un procesador de 8 bits, ¿a cuántos puertos de E/S podrá acceder?

a) 8

-> b) Depende del método de selección de periféricos que emplee

c) 256

d) 65536

88. Algunas de las ventajas de la E/S mapeada en memoria frente a la E/S aislada o

independiente son:

a) Los puertos de E/S no ocupan direcciones de memoria.

b) Facilita la protección de E/S.

-> c) El diseño de la CPU es más sencillo.

d) Todas las respuestas anteriores son ciertas

89. Un computador con 15 líneas de direcciones tiene 3 módulos de memoria de  $2^{13}$

palabras y utiliza E/S mapeada en memoria. ¿Cuál es el número máximo de periféricos que

pueden conectarse, si cada uno de ellos utiliza 8 direcciones?

a)  $2^{12}$

-> b)  $2^{10}$

c)  $2^{13}$

d)  $2^{11}$

90. ¿Cuál es el número máximo de niveles de interrupción que se pueden manejar utilizando

3 controladores de interrupciones programables 8259?

a) 24

-> b) 22

c) 20

d) 18

91. ¿Cuál de los siguientes elementos no forma parte de un canal de un controlador de

acceso directo a memoria?

a) Registro de dirección

b) Registro de órdenes

c) Registro contador

-> d) Registro de prioridades

92. ¿Cuántos puertos de E/S permite manejar la interfaz de periféricos programable 8255?

a) 2 puertos de 16 bits

-> b) 3 puertos de 8 bits

c) 4 puertos de 32 bits

d) Todas las respuestas anteriores son falsas

93. ¿Cuál de las siguientes afirmaciones es cierta?

-> a) La E/S independiente facilita la protección

b) En E/S independiente, las instrucciones de acceso a memoria se emplean tanto para memoria como para E/S

c) La E/S en memoria emplea la patilla IO/M#

d) La E/S en memoria es mucho más rápida que la E/S independiente

94. ¿Cuál de las siguientes afirmaciones acerca del concepto de interrupción es cierta?

a) Solicita que el procesador se aisle de los buses

b) Su objetivo es incrementar el ancho de banda con el dispositivo

-> c) Es una bifurcación normalmente externa al programa en ejecución

d) Permite realizar transferencias sin el control de un programa

95. ¿Cuál de las siguientes afirmaciones acerca del concepto de interrupción es falsa?

a) Solicita que se ejecute un programa específico para tratarla

-> b) Ninguna de las afirmaciones es falsa

c) Su objetivo es reclamar la atención del procesador

d) Es una bifurcación normalmente externa al programa en ejecución

96. ¿Qué instrucciones son típicas del repertorio de un procesador que sólo disponga de E/S mapeada en memoria?

a) IN, LOAD, OUT

b) IN, LOAD, MOV

-> c) LOAD, MOV, STORE

d) Ninguno de los anteriores

97. De los siguientes grupos de técnicas de E/S, ¿cuáles están controlados por programa?

- > a) E/S programada y E/S mediante interrupciones.
- b) E/S programada y E/S mediante acceso directo a memoria.
- c) E/S mediante interrupciones y E/S mediante acceso directo a memoria.
- d) Todas las respuestas anteriores son ciertas.

98. Sobre las técnicas de transferencia en operaciones de E/S:

- a) Pueden ser controladas por programa o por hardware
- b) Si se emplea E/S programada puede hacerse con consulta de estado o sin consulta de estado
- c) En el caso de utilizar E/S mediante DMA hace falta emplear un controlador de DMA
- > d) Todas las respuestas anteriores son ciertas

99. En la práctica de E/S en Arduino, la instrucción SBI del repertorio del microcontrolador realiza...

- a. una suma de un valor inmediato
- b. una resta de un valor inmediato
- > c. una operación de bits //P5 guión, pág.9-10
- d. un salto incondicional.

100. En la práctica de E/S en Arduino, el DDRB es el...

- a. segundo canal de memoria DDR
- > b. registro de dirección de datos del puerto B // P5 guión, pág.9-10
- c. registro de datos y direcciones B
- d. registro buffer de datos D

101. En la práctica de E/S en Arduino, la instrucción SBIW del repertorio del microcontrolador realiza...

- a. una suma de un valor inmediato
- > b. una resta de un valor inmediato //P5 guión, pág.9-12
- c. una operación de bits
- d. un salto incondicional

102. En la práctica de E/S en Arduino, la instrucción CBI del repertorio del microcontrolador realiza...

- a. una comparación
- b. un complemento a uno

-> c. una operación de bits //P5 guión, pág.9-10

d. una llamada a subrutina

103.Señale cuál de las siguientes opciones es una técnica para llevar a cabo la transferencia de datos entre el computador y los dispositivos de E/S externos:

->a) E/S programada

b) E/S por flanco

c) E/S por nivel

d) Acceso indirecto a memoria

104.La técnica de Consulta de Estado (polling) puede usarse para...

a) identificar el origen de una interrupción

b) consultar si el dispositivo está dispuesto para entregar o recibir datos

->c) consultar el sentido del DMA en curso (desde memoria / hacia memoria)

d) establecer un mecanismo software de asignación de prioridades a los dispositivos

105.Un computador con 20 líneas de dirección y memoria de bytes tiene 640KB de RAM, 128KB de ROM, y utiliza E/S mapeada en memoria. ¿Cuál es el número máximo de periféricos que pueden conectarse, si cada uno de ellos utiliza 32 direcciones?

a)  $2^{10}$

b)  $2^{11}$

c)  $2^{12}$

->d)  $2^{13}$

106.Respecto a las técnicas de direccionamiento por selección lineal, decodificación centralizada y distribuida

a) la selección lineal permitiría leer una palabra simultáneamente desde varios puertos de E/S

->b) usando decodificación distribuida es más fácil realizar expansiones al sistema de E/S

c) todas ellas impiden que haya cortocircuito en el bus de datos

d) todas ellas impiden que haya cortocircuito en el bus de direcciones

107.Respecto a la interfaz de E/S, ¿cuál de las siguientes afirmaciones es \*FALSA\*?

UsuarioProfesores

a) Involucra tareas que se pueden implementar parte en hardware y parte en software.

b) Permite configurar el funcionamiento del periférico en un momento determinado, y además conocer su estado.

c) Puede guardar temporalmente en registros internos tanto datos generados por el periférico para ser enviados al procesador, como datos que son enviados desde el procesador al periférico.

->d) Una interfaz de entrada recibe los datos desde el procesador y los transforma y envía al periférico en formato digital.

no, se dice Entrada (y Salida) desde el punto de vista del procesador

108. En la práctica del Theremín de luz de Arduino se pide combinar el código del Proyecto p06 (sólo fotocélula y zumbador) con el de la Lección 26 (fotocélula y leds). Para conseguir que se enciendan y apaguen correctamente todos los led, lo mejor sería:

- a) Reutilizar la variable pitch calculada mediante map() como la nueva numLEDSLit
  - b) Fusionar las variables sensorValue y reading de forma que sólo se llame una vez a analogRead()
  - >c) Calcular numLEDSLit con otra función map() de 0 a 8, en lugar de barrer de 50 a 4000
  - d) Copiar uno y otro código en las secciones correspondientes, no se puede fusionar nada
109. En un sistema de interrupciones vectorizado y en daisy-chain, ¿cuál de las siguientes afirmaciones es cierta?
- >a) La gestión de prioridades queda establecida por el orden en que los dispositivos reciben la señal INTA y el dispositivo se identifica por un dato que deposita en el bus
  - b) El procesador informa de un ciclo de reconocimiento de interrupción con la señal de reconocimiento de interrupción (INTA) y la identificación de los dispositivos se realiza por consulta de estado
  - c) El daisy-chain asigna a todos los dispositivos la misma prioridad y la identificación de los dispositivos se realiza leyendo sus registros de estado
  - d) La gestión de prioridades queda establecida por el orden en que los dispositivos reciben la señal INTA y la identificación de los dispositivos se realiza leyendo sus registros de estado

## TEMA 6

1. Una jerarquía de memoria consta de una cache de con una tasa de aciertos del 92% y 4 ns de tiempo de acceso y una memoria principal con una tasa de aciertos del 100% y 100 ns de tiempo de acceso. ¿Cuál es el tiempo promedio estimado de acceso a memoria?

- a) 8 ns
- b) 6 ns
- c) 10 ns
- > d) 12 ns

2. El ancho de banda de memoria es:

- a) el número de bits que se pueden transferir entre ésta y la CPU en paralelo en una sola operación de lectura o escritura
- b) el tiempo que se tarda en transferir una palabra entre memoria y CPU
- c) el intervalo de frecuencias de reloj permitidas entre memoria y CPU
- > d) ninguna de las anteriores es cierta

3. En un sistema con memoria de bytes y líneas de cache de 32 bytes, ¿dónde empieza el bloque de memoria que contiene la posición 0xAC72?

- > a) 0xAC60
- b) 0xAC6E
- c) 0xAC70
- d) 0xAC72

4. En un sistema con direcciones de 32bits, memoria de bytes, cache de 1MB asociativa por conjuntos de 4 vías y líneas de 64B, el campo etiqueta en el formato de dirección cache es de

- a) 16 bits
- > b) 14 bits
- c) 10 bits
- d) 12 bits

5. Una memoria que está organizada en palabras de 8 bits tiene una capacidad de 32 Kbits.  
¿Cuántas líneas de dirección tiene dicha memoria?

- > a) 12
- b) 4
- c) 8

d) 32'

6. Considere un sistema de memoria para un procesador de 32 bits con caches separadas para código y datos. Suponga que el procesador direcciona la memoria por bytes y realiza accesos a palabras de 32 bits y que el espacio de direcciones es de 2<sup>32</sup> bytes. La cache de datos tiene las siguientes características: 64 KB de capacidad, asociativa por conjuntos con 2 vías, y bloques de 2 palabras. ¿Cuántos bits tiene el campo etiqueta de una dirección de memoria?

a) 13

b) 11

c) 15

-> d) 17

7. ¿Cuál de las siguientes afirmaciones acerca de las memorias SRAM es falsa?

-> a) Las operaciones de lectura son destructivas

b) Son más veloces que las memorias RAM dinámicas

c) El número de transistores necesario para implementar cada celda es mayor que en las memorias RAM dinámicas

d) Los datos almacenados se mantienen por un tiempo indefinido mientras se mantenga la alimentación

8. Un circuito SRAM con una capacidad de 256 Kbits tiene las patillas de direcciones A<sub>14</sub> a A<sub>0</sub>. ¿Con cuál de las siguientes expresiones indicaría las características de capacidad en direcciones y datos del circuito?

-> a) 32 K x 8

b) 8 K x 32

c) 64 K x 4

d) 256 K x 1

9. Un procesador accede en el instante de tiempo t a una posición de memoria d(t). Poco tiempo después (en el instante de tiempo t+k) accede a la posición anterior d(t)-1. Esos dos accesos son un ejemplo de...

-> a) Localidad espacial

d(t+k)=d(t)-1

b) Localidad temporal

c) No tiene nombre, ese tipo de localidad con incremento negativo ( $d(t)-1$ ) no se ha estudiado en clase

d) No es una localidad, esa condición no guarda relación con el concepto de localidad

10. Si el tiempo de acceso a la memoria cache es de 2 ns y el tiempo necesario para tratar un fallo de cache es de 80 ns, ¿cuál es la tasa de aciertos necesaria para que el tiempo medio de acceso al sistema de memoria sea de 10 ns?

a) 0.8

b) 0.75

-> c) 0.9

d) 0.95

11. La política de correspondencia de una memoria cache con la mitad de conjuntos que líneas es:

a) Asociativa por conjuntos con 2 líneas

con cuántas líneas? será cuántas vías!

b) Directa con 2 líneas

en directa no hay que decir vías ni líneas ni nada, basta saber los tamaños de línea y cache

c) Totalmente asociativa de media vía

media vía?

-> d) Asociativa por conjuntos de 2 vías

"vía" se refiere a cuántas "alternativas" (líneas) hay en un conjunto.

Además, todas las demás opciones tienen defectos

12. ¿Cuál de las siguientes afirmaciones acerca de las memorias RAM dinámicas es cierta?

a) Los datos permanecen en cada celda indefinidamente

b) Las operaciones de lectura no son destructivas

c) Las celdas de almacenamiento son complejas

-> d) Las operaciones de escritura sirven como operaciones de refresco

13. ¿En qué se diferencian las estrategias de mantenimiento de coherencia en memoria "escritura directa" y "post-escritura"?

-> a) En cuándo tiene lugar la actualización

b) En cómo tiene lugar la actualización

c) Tanto en a) como en b)

d) Ni en a) ni en b)

14. La "postescritura (write-back) marcada"

a) requiere más bits de modificación ("bits sucios") cuando aumenta el número de vías

b) requiere menos hardware que la "postescritura siempre"

-> c) es más eficiente que la "postescritura siempre"

d) provoca una menor tasa de faltas que la "postescritura siempre"

15. Un sistema basado en un microprocesador con un bus de datos de n bits y un bus de direcciones de 16 bits direcciona la memoria por palabras de n bits y dispone de una memoria SRAM formada por dos módulos de 16 K x n cada uno. ¿Qué porcentaje del mapa de memoria está ocupado por la SRAM?

-> a) 50%

16bits => 64Kpal

16+16 Kpal = 32Kpal = 50% 64Kpal

b) 100%

c) 25%

d) 12,5%

16. ¿Cuál de los siguientes es el ejemplo más acertado de localidad espacial?

a) Reservar dinámicamente (malloc) espacio para una estructura o union

b) Referenciar continuamente la misma variable local

c) Iterar repetidamente el cuerpo de un bucle

-> d) Referenciar elementos de un array sucesivamente

17. ¿En qué tipo de memorias coincide el tiempo de acceso y el tiempo de ciclo?

-> a) SRAM

b) DRAM

c) Tanto en a) como en b)

d) Ninguna de las anteriores

18. ¿Qué conjunto de componentes permite construir una memoria 256Mx32? (sin que sobren componentes)

a) 16 chips 64Mx4

-> b) 32 chips 64Mx4

c) 16 chips 64Mx16

d) Ninguna de las anteriores

19. ¿Cuál es el tamaño de la marca de cache si el bus de direcciones es de 48 bits (256 TB de memoria principal) y hay 8MB de cache L3, con un tamaño de línea de 64 B y correspondencia asociativa por conjuntos con 16 vías?

a) 13 bits

b) 6 bits

-> c) 29 bits

d) 48 bits

20. La cache con correspondencia directa se puede considerar como un caso límite de la asociativa por conjuntos, en donde...

-> a) solo hay 1 línea por conjunto

b) solo hay 1 palabra por bloque

c) solo hay 1 conjunto por cache

d) ninguna de las anteriores

21. ¿Qué necesitamos para construir una memoria de 1K x 8 bits?

a) 64 memorias de 128 x 1 bits

b) 8 memorias de 512 x 2 bits

-> c) 8 memorias de 256 x 4 bits y un decodificador de 2 a 4

d) Ninguna de las anteriores respuestas es cierta

22. ¿Qué política de colocación en cache necesita más comparadores, la correspondencia asociativa por conjuntos o la correspondencia por sectores?

a) Correspondencia por sectores

b) Correspondencia asociativa por conjuntos

c) Depende de si es mayor el número de conjuntos o el número de sectores

-> d) Depende de si es mayor el número de bloques por conjunto o el número de sectores

23. La cache es gestionada por:

a) el programador

b) el sistema operativo

-> c) unidades de "manejo" (gestión) hardware

d) ninguna es cierta

24. La memoria cache en un sistema computador es:

- a) Más rápida que la memoria principal
  - b) De menor capacidad que la memoria principal
- > c) a) y b) son correctas
- d) Ninguna de las anteriores es correcta

25. La memoria DRAM:

- a) Se inventó en la década de los 90
  - b) Necesita 6 transistores por cada celda
  - c) Es menos densa que la memoria SRAM
- > d) Se denomina dinámica porque para mantener almacenado un dato hay que recargarlo cada cierto tiempo en un ciclo de refresco

26. En un sistema con memoria de bytes y líneas de cache de 64 bytes, ¿dónde empieza el bloque de memoria que contiene la posición 0xBEE3DE72?

- a) 0xBEE3DE6E
  - b) 0xBEE3DE70
- > c) 0xBEE3DE40
- d) 0x0EE3DE72

27. Una SRAM de 32Kx8bit (256Kbit) puede venir organizada en 512 filas, dedicando por tanto al decodificador de columnas...

- a) 8 bits
- > b) 6 bits
- c) 7 bits
- d) 9 bits

28. Un computador con 10 bits en el bus de direcciones puede direccionar como máximo:

- > a) 1024 palabras
- b) 1000 palabras
- c) 65535 palabras
- d) 65536 palabras

29. ¿Cuál es el ancho del bus de direcciones de un chip DRAM de 1G palabra, siendo la longitud de palabra de 16 bits?

- a) 30
- > b) 15

c) 16

d) 20

Redacción original: "para una memoria DRAM"

Tal vez convendría modificar aún más el enunciado y preguntar el "nº de bits de direccionamiento en un chip DRAM..."

30. ¿Cuál de las siguientes afirmaciones acerca de la jerarquía de memoria es \*FALSA\*?

a) Una memoria principal constituida por la tecnología más rápida es órdenes de magnitud más cara que la DRAM

b) Acceder a los discos es órdenes de magnitud más lento que acceder a la RAM

-> c) La velocidad de acceso a la memoria principal ha crecido proporcionalmente a la velocidad del procesador

d) Un computador puede tener una pequeña cantidad de memoria rápida además de una gran cantidad de memoria más lenta.

31. ¿En qué tipo de refresco de memoria DRAM CAS# permanece a 0 después del ciclo de lectura o escritura precedente?

a) RAS# antes de CAS#

b) Sólo RAS#

-> c) Refresco transparente

d) Ninguna de las anteriores respuestas es correcta

32. ¿Cuál de las siguientes afirmaciones sobre memorias cache es cierta? Recordar que llamamos "latencia" al tiempo transcurrido desde que se envía una dirección a la cache hasta que se obtiene el dato (suponiendo que se trata de un acierto), y depende por tanto del tiempo que emplee la circuitería hardware en obtenerlo. Suponer similares condiciones para ambas caches (tamaño de bloque, tamaño total, tecnología, frecuencia...)

a) Ambas tienen en general similar latencia y tasa de fallos

-> b) Las caches totalmente asociativas ofrecen tasas de fallo más bajas, mientras que las de correspondencia directa tienen mejor latencia

c) Las caches de correspondencia directa tienen mejor latencia y tasa de fallos

d) Las caches totalmente asociativas ofrecen mejor latencia, mientras que las de correspondencia directa tienen tasas de fallo más bajas

33. ¿Cuál de las siguientes afirmaciones es falsa?

- > a) Las memorias DRAM son en general mucho más rápidas que las SRAM
- b) Una celda DRAM sólo necesita un transistor y un condensador
- c) Las memorias DRAM presentan generalmente una capacidad de almacenamiento mucho mayor que las SRAM
- d) La operación de lectura de una celda DRAM es destructiva

34. En una cache asociativa por conjuntos, la vía i está constituida por:

- > a) todos los bloques  $i$ -ésimos de cada conjunto
- b) todos los bloques del conjunto  $i$
- c) todos los conjuntos del bloque  $i$
- d) ninguna de las anteriores es cierta

35. ¿Cuántas líneas de dirección son necesarias en un memoria RAM dinámica de 256 K palabras? ¿Y en una estática?

- a) 9/9
- b) 18/9
- > c) 9/18
- d) 18/18

36. Las técnicas write-through y write-back están relacionadas con

- > a) coherencia de cache
- b) arbitraje de buses
- c) métodos de E/S
- d) etapas de la unidad de control

37. ¿Cuál de las siguientes afirmaciones acerca de la memoria es \*FALSA\*?

- a) La memoria dinámica usa señales de control RAS# y CAS#
- b) Las celdas de memoria dinámica están constituidas por un transistor y un condensador
- > c) Las celdas de memoria estática tienen que ser constantemente refrescadas
- d) La memoria estática se emplea en las caches L1 y L2

38. Se dispone de un computador cuyo tiempo medio de acceso al sistema de memoria cache y memoria principal es de 18 ns. Si la tasa de fallo de la cache es de 0,2 y el tiempo de acceso a la memoria principal es 50 ns. ¿Cuál es el tiempo de acceso a la cache?

- a) 6 ns
- b) 24,4 ns

c) 10 ns

-> d) 8 ns

39. Un sistema de memoria tiene un tiempo medio de acceso de 10 ns por operación de lectura o escritura y un ancho de datos de 32 bits. ¿Cuál es el ancho de banda del sistema de memoria?

a) 2,5 ns

b) 32 Mbytes por segundo

-> c) 400 millones de bytes por segundo

d) 32 bits

40. En un sistema Linux IA32, ¿cuántos enteros se podrían almacenar en una línea de cache, si la cache del procesador fuera de 4 KB, asociativa por conjuntos de 4-vías, y contuviera 4 conjuntos?

a) 16

b) 128

c) 32

-> d) 64

41. ¿Cuál de los siguientes métodos para incrementar el ancho de banda de memoria es más económico?

a) Utilizar memorias de alta velocidad

b) Duplicar el tamaño de la memoria

-> c) Organizar la memoria jerárquicamente

d) Utilizar memorias asociativas

42. Un módulo de memoria de 16 GB está formado por varios chips DRAM de 1024Mx4.

¿Cuántos chips DRAM necesita el módulo?

-> a) 32

$$16\text{Gx8} = 16 \cdot (1024\text{M}) \times 2 \cdot (4)$$

b) 8

c) 16

d) 4

43. Con 8 circuitos de memoria RAM de 1K x 8 se puede crear un memoria de:

a) 1K x 64

b) 8K x 8

c) 2K x 32

-> d) Todas las combinaciones anteriores son posibles

44. En EC podemos usar la palabra directo para referirnos a... (señalar la opción incorrecta)

-> a) dispositivo de almacenamiento secuencial directo (DASD)

DASD significa Direct Access Storage Device

b) acceso directo a memoria

c) cache con correspondencia directa

d) modo de direccionamiento directo

45. ¿Cuál de las siguientes es una idea fundamental de la jerarquía de memoria?

a) Que dispositivos más grandes y lentos sirvan de cache para dispositivos más pequeños y rápidos

b) Crear una pequeña cantidad de almacenamiento que sea caro y lento

-> c) Que dispositivos más pequeños y rápidos sirvan de cache para dispositivos más grandes y lentos

d) Crear una gran cantidad de almacenamiento que sea caro y rápido

46. En el siguiente código, ¿qué reordenamiento de los bucles muestra mejor localidad?

// X, Y, Z ctes #define previo

int a[X][Y][Z]

int i, j, k, sum = 0;

for (i = 0; i < Y; i++)

for (j = 0; j < Z; j++)

for (k = 0; k < X; k++)

sum += a[k][i][j];

a) El orden de los bucles no afecta a la localidad

b) j externo, k central, i interno

c) i externo, j central, k interno (el orden en que están ahora)

-> d) k externo, i central, j interno

47. Una SRAM de 1Mx4bit (4Mbit) puede venir organizada en 2048 filas, dedicando por tanto al decodificador de columnas...

-> a) 9 bits

$$1M = 2^{20} = 2^{11} \cdot 2^9 = 2048 \text{ filas} \cdot 512 \text{ columnas}$$

observar que no es 1Mbit organizado en 2048 filas (en cuyo caso la respuesta sería 7bits)

b) 7 bits

c) 8 bits

d) 6 bits

48. Un computador emplea un sistema de memoria principal de 128 palabras y una memoria cache de 32 palabras. La organización de la memoria cache es totalmente asociativa y el tamaño de bloque es de 8 palabras. Se emplea el algoritmo de reemplazo FIFO. Si inicialmente la memoria cache está totalmente vacía, calcule el número de fallos cuando se lee la secuencia de direcciones de la memoria principal: 0000100, 1000001, 0000101, 0010011, 0100010, 1000100, 0000111.

a) 3 fallos

b) 5 fallos

-> c) 4 fallos

d) 6 fallos

49. ¿Cuál de las siguientes afirmaciones es cierta?

a) La memoria SRAM es más lenta que la DRAM

b) La lectura en la memoria SRAM es destructiva

c) La memoria DRAM es más cara que la SRAM

-> d) Ninguna de las anteriores

50. En una memoria DRAM que permite el acceso en modo página se accede a la palabra 0x1234. Si emplea páginas de 256 palabras, ¿Cuál será la menor dirección a la que podremos acceder rápidamente?

a) 0x1000

-> b) 0x1200

c) 0x1230

d) Otra

51. En los sistemas con una jerarquía de memoria dividida en varios niveles se da el problema de la consistencia o coherencia de los datos entre los distintos niveles. Si una palabra se modifica en un nivel, en algún momento habrá que traspasar ese cambio a los niveles inferiores (más lejanos al procesador). Para ello hay varias políticas:

- > a) Post-escritura: se retrasa la actualización en los niveles inferiores hasta que el bloque modificado tenga que ser reemplazado
- b) Escritura indirecta: si se modifica una palabra, inmediatamente se modifican los niveles superiores
- c) Las respuestas a y b son ciertas
- d) Las respuestas a y b son falsas

52. En las políticas anticipativas de extracción de cache, ¿cuál de ellas se caracteriza por preextraer el bloque  $i+1$  si se referencia al bloque  $i$  y se produce falta de bloque?

- a) Preextracción marcada
- b) Preextracción siempre
- c) Preextracción indexada
- > d) Preextracción por falta

53. Para construir una DRAM de 4GB con pastillas de 512Mx4bit hacen falta

- a) 64 pastillas
- b) 8 pastillas
- > c) 16 pastillas
- x8 "a lo alto", x2 "a lo ancho"
- d) 32 pastillas

54. Una cache de 256 B asociativa por conjuntos de 4-vías con líneas de 16 B tendría

- a) 64 conjuntos
- b) ningún conjunto
- > c) 4 conjuntos

$$256 \text{ B} = 2^8 \text{ B} = 2^4 \text{ líneas} \cdot 2^4 \text{ B/lín}$$

$$2^4 \text{ lín} = 2^2 \text{ conj.} \cdot 2^2 \text{ vías}$$

- d) 16 conjuntos

55. El primer nivel de una jerarquía de memoria tiene una tasa de aciertos del 75% y las peticiones de memoria tardan 12 ns en completarse si dicha posición se encuentra en ese nivel y 100 ns si no es así. ¿Cuál es el tiempo medio de acceso de la jerarquía?

- > a) 34 ns
- b) 25 ns
- c) 112 ns

d) 88 ns

56. ¿En qué tipo de ciclo de refresco se hace RAS# = 0?

a) Sólo RAS#

b) CAS# antes de RAS#

c) Refresco transparente

-> d) En todos los anteriores

57. Sea un computador de 32 bits que dispone de una memoria cache de 512 KB y líneas de 64 bytes. ¿Cuántas líneas tiene la cache?

a) 1024

b) 64

c) 65536

-> d) 8192

58. Para direccionar una memoria de 16K x 16 necesitamos un bus de direcciones de:

-> a) 14 bits

b) 16 bits

c) 4 bits

d) Otro valor

59. Sea una cache asociativa por conjuntos de 4-vías. ¿Cuál de las siguientes afirmaciones es cierta?

-> a) La cache tiene 4 líneas por conjunto

b) La cache tiene 4 conjuntos por bloque

c) La cache tiene 4 bloques por línea

d) La cache tiene 4 conjuntos por línea

60. Utilizar una cache en el mismo chip del procesador:

-> a) reduce los tiempos de ejecución

b) aumenta el tamaño de los bloques enviados entre cache y procesador

c) aumenta la tasa de aciertos

d) reduce el tamaño del bus

61. ¿Qué tipo de localidad de las referencias a memoria se define como: "si se referencia un elemento, los elementos cercanos a él serán referenciados pronto"?

-> a) Localidad espacial

b) Localidad secuencial

c) Localidad temporal

d) Ninguna de las respuestas anteriores es correcta

62. ¿A qué tipo de memoria cache corresponde la siguiente afirmación: "permite que cualquier dirección se pueda almacenar en cualquier marco de bloque de cache"?

a) Con correspondencia directa

-> b) Totalmente asociativa

c) Asociativa por conjuntos

d) Ninguna de las anteriores

63. Se desea construir una memoria de SRAM de tamaño 3G X 8 a partir de elementos de memoria SRAM más pequeños. Cuál de las siguientes soluciones sería correcta:

a) 256 chips de 16Mx 1 bits

b) 16 chips de 512 M x 2 bits

-> c) 12 chips de 512M x 4 bits

d) Ninguna de las anteriores es correcta

64. En una jerarquía de memoria, a medida que nos alejamos del procesador:

a) el tiempo de transferencia disminuye

-> b) el tamaño de la unidad de transferencia entre dos niveles aumenta

c) el tamaño de la memoria disminuye

d) el coste por byte aumenta

65. Un sistema tiene una cache asociativa por conjuntos de 2-vías con 16 conjuntos y líneas de 64B. ¿A qué conjunto le corresponde el byte con dirección Oxdeadbeef?

a) 13

b) 14

c) 7

-> d) 11

66. Un programa crea en memoria una larga secuencia de números de forma consecutiva.

¿Qué tipo de estrategia de mantenimiento de coherencia es más eficiente para ejecutar este programa en un sistema con jerarquía de memoria?

a) Escritura directa ("write-through")

-> b) Post-escritura ("write-back")

c) Tanto a) como b) son igual de eficientes

d) No puede saberse qué técnica es mejor

67. Variación de los parámetros de los distintos niveles en una jerarquía de memoria

(señalar la opción incorrecta)

a) ancho de banda:  $b_i \geq b_{i+1}$

-> b) unidad de transferencia:  $x_i \geq x_{i+1}$

c) tiempo de acceso:  $t_i \leq t_{i+1}$

d) tamaño del nivel:  $s_i \leq s_{i+1}$

68. ¿Qué tipo de localidad de las referencias a memoria se define como: "si se referencia un

elemento, volverá a ser referenciado pronto"?

a) Localidad espacial

b) Localidad lógica

-> c) Localidad temporal

d) Ninguna de las respuestas anteriores es correcta

69. ¿Qué tipo de localidad de las referencias a memoria se define como: "si se referencia un

elemento, volverá a ser referenciado pronto"?

a) Localidad secuencial

b) Localidad espacial

c) Localidad iterativa

-> d) Localidad temporal

70. En el contexto de las DRAM, RAS significa:

a) Random Access Strobe (muestreo de acceso aleatorio)

b) Random Access Shot (disparo de acceso aleatorio)

c) Refresh After Select (refresco después de selección de la memoria)

-> d) Row Access Strobe (muestreo de acceso a filas)

71. ¿Cuál es el tamaño de la etiqueta de cache en un ordenador capaz de direccionar por

bytes 1 MB de memoria principal y 32 KB de memoria cache y correspondencia asociativa

por conjuntos con 32 bytes por línea y 16 líneas por conjunto?

a) 7 bits

b) 6 bits

c) 8 bits

-> d) 9 bits

72. ¿Cuántas líneas de dirección son necesarias en un memoria RAM dinámica de 64 K palabras? ¿Y en una estática?

a) 16 / 16

b) 16 / 8

-> c) 8 / 16

d) 8 / 8

73. ¿Cuáles de las siguientes direcciones de memoria podrían estar simultáneamente en una memoria cache de 256 palabras con 16 palabras por bloque y con correspondencia directa ?

a) 0x0000 y 0xFFOF

-> b) 0xABAB y 0xABAC

c) 0x08E3 y 0x74E1

d) Ninguna de las combinaciones anteriores

74. Respecto al refresco de memorias DRAM, ¿cuál de las siguientes afirmaciones es falsa?

a) Se precisa una circuitería auxiliar, externa al chip DRAM o integrada en él, que produzca ciclos de refresco.

b) Los ciclos de refresco deben producirse cada pocos ms (milisegundos).

c) Los chips DRAM refrescan automáticamente la fila accedida en cualquier ciclo de lectura o escritura.

-> d) Una operación de refresco consiste en dar un impulso /CAS junto con una dirección de columna.

75. En una cache con 64 bytes de longitud de línea, ¿qué bits de una dirección de memoria de 64 bits se utilizan para determinar a qué byte dentro de la línea se refiere dicha dirección? (Memoria direccionable por bytes)

a) [8...6]

b) [11...6]

c) [5...3]

-> d) [5...0]

76. Si se necesitan 60 ns para escribir una palabra de datos de cache en memoria principal y cada bloque de cache tiene 8 palabras, ¿cuántas veces "seguidas" (sin que haya reemplazo) se tiene que escribir en un mismo bloque para que una cache de postescritura sea más

eficiente que una de escritura inmediata?

a) Más de 8 veces

-> b) No se puede responder con los datos proporcionados

c) La cache de postescritura no puede ser más eficiente que la de escritura inmediata

d) La cache de postescritura siempre será más eficiente que la de escritura inmediata

77. ¿Cuántas patillas de dirección tiene una memoria DRAM de 1G palabra, siendo la longitud de palabra de 16 bits?

a) 30

-> b) 15

c) 16

d) 20

78. Una memoria SRAM tiene una capacidad de 64 Kbits y utiliza 12 líneas para direccionamiento. Indique cuál es el tamaño de palabra de dicha memoria:

-> a) 16 bits

b) 8 bits

c) 32 bits

d) 64 bits

79. Los módulos de memoria dinámica compactos que suelen usarse en los portátiles se denominan:

-> a) SODIMM

b) SLIM

c) SIMM

d) MIN

80. ¿Cuál de las siguientes afirmaciones acerca de una jerarquía de memoria es cierta?

-> a) Para aumentar la eficiencia se transfieren bloques completos

b) Toda la información que la CPU necesita está en el nivel 1

c) Si una palabra no se encuentra en el tercer nivel entonces se busca en el segundo nivel

d) Todas las afirmaciones anteriores son falsas

81. La tasa de aciertos  $A_i$  del nivel  $i$  de una jerarquía de memoria no depende de

a) La unidad de la transferencia de información  $x_i$  entre el nivel  $i$  y el  $i+1$ .

b) La capacidad (tamaño)  $s_i$  del nivel  $i$ .

-> c) El ancho de banda  $b_i$  del nivel  $i$ .

Todas las demás se mencionan explícitamente en Tema 6 tr.24

d) La estrategia de administración de memoria.

81. ¿Cuál es el tamaño de la marca de cache en un microprocesador con 32 KB de memoria cache asociativa por conjuntos con 16 palabras de 32 bits por bloque y 8 bloques por conjunto, si el microprocesador es capaz de direccionar 1 MB de memoria principal (memoria de bytes)?

-> a) 8 bits

b) 7 bits

c) 10 bits

d) 6 bits

82. ¿Qué dice la ley de Moore?

a) Que el tamaño de los transistores se duplica cada 18 meses.

b) Que la memoria de los ordenadores se duplica cada 18 meses.

-> c) Que las prestaciones de los transistores en un chip se duplican cada 18 meses.

d) Todas las respuestas son ciertas.

83. La política de correspondencia de una memoria cache con 1 único conjunto es:

-> a) Totalmente asociativa

todas las líneas son distintas vías del único conjunto -> cualquier bloque de memoria puede ir a cualquier marco de cache. Ver Tema 6 tr.137

b) Asociativa por conjuntos con una única línea

c) Directa

d) Asociativa por conjuntos de una única vía

84. ¿Cuántas líneas de dirección (patillas) son necesarias para direccionar un chip de

memoria DRAM de 4096 x 4?

a) 11

-> b) 6

c) 12

d) 10

85. ¿Cuál de las siguientes afirmaciones acerca de las memorias RAM estáticas es falsa?

a) Los datos almacenados se mantienen por un tiempo indefinido

- b) Las operaciones de lectura no son destructivas
  - c) Son más veloces que las memorias RAM dinámicas
- > d) El número de transistores necesario para implementar cada celda es menor que en las memorias RAM dinámicas

86. Suponga el siguiente diseño cache/memoria: direcciones de 16 bits, direccionamiento por bytes, tamaño de cache 256 bytes, tamaño de bloque 8 bytes, tamaño de etiqueta 11 bits. ¿Cuál es la asociatividad (cuántas vías hay en la cache)?

- a) 4
- > b) 8
- c) 2
- d) 5

87. Sólo una de las siguientes afirmaciones sobre memorias ROM es correcta. ¿Cuál?

- a) Una PROM (Programmable ROM) se puede grabar usando un dispositivo programador que selectivamente funde contactos aplicándoles altas temperaturas mediante diminutas cabezas soldadoras ("equipo de puntas")
- b) Una EEPROM (Erasable EPROM) se puede grabar (eléctricamente), y borrar (usando rayos ultravioleta)
- > c) Para fabricar una ROM se deben conocer los datos que se desea que almacene
- d) Una EPROM (Electrically Progr. ROM) se puede grabar eléctricamente, sin fundir contactos, pero no se puede borrar

88. En un sistema con memoria de bytes, ¿cuál sería el tamaño de una línea de cache, si la cache del procesador fuera de 4MB, asociativa por conjuntos de 16-vías, y contuviera 4096 conjuntos?

- > a)  $64 \text{ B} // 4\text{MB} = 2^{22} \text{ B} = 2^{12} \text{ conj.} \cdot 2^4 \text{ lin./conj. (vías)} \cdot 2^6 \text{ B/lin.}$
- b) 16 B
- c) 128 B
- d) 32 B

89. Cada celda de un chip de memoria DRAM de 1M x 1, organizada en una matriz de 512 filas x 2048 columnas, necesita ser refrescada cada 16 ms. ¿Cada cuánto tiempo ha de realizarse una operación de refresco en el chip?

- a) 61 nanosegundos

b) 7,8125 microsegundos

-> c) 31,25 microsegundos

d) 8192 milisegundos

90. ¿En qué tipo de ciclo de refresco RAS# permanece a 1?

a) Sólo RAS#

b) CAS# antes de RAS#

c) Refresco transparente

-> d) En ninguno de los anteriores

91. ¿Cuál de las siguientes secuencias de tipos de memoria está ordenada de menores a mayores prestaciones?

a) DDR, SDRAM, FPM

b) EDO, SRAM, FPM

-> c) FPM, EDO, RDRAM

d) SDRAM, DDR, EDO

92. En una memoria organizada en forma jerárquica, ¿qué suele ocurrir con respecto al tamaño de las unidades de transferencia entre niveles, conforme se baja de nivel hacia el procesador?

a) Aumenta

-> b) Disminuye

c) Aumenta en algunos niveles y disminuye en otros

d) Todas las posibilidades anteriores suelen darse

93. En un sistema con memoria de bytes y líneas de cache de 64 bytes, ¿qué bits de una dirección de memoria se utilizan para determinar a qué byte dentro de la línea se refiere dicha dirección?

-> a) Los 6 bits menos significativos

b) Los 6 bits más significativos

c) Los 4 bits menos significativos

d) Los 4 bits más significativos

94. Para diseñar una memoria con ancho de palabra  $k \cdot m$  (y mismo nº palabras que los módulos) a partir de módulos con ancho de palabra  $m$ , se utilizan  $k$  módulos

(mediante  $_$  se representa subíndice)

-> a) repartiendo líneas datos: el 1º se conecta a D\_0...D\_m-1, el 2º a D\_m...D\_2m-1, etc

Corregida errata D2\_m-1

b) repartiendo las líneas de datos entre los k módulos: el primero se conecta a D\_0...D\_k-1, el segundo a D\_k...D\_2k-1, etc

c) repartiendo líneas dirección: el 1º se conecta a A\_0...A\_m-1, el 2º a A\_m...A\_2m-1, etc

d) repartiendo las líneas de dirección: el 1º se conecta a A\_0...A\_k-1, el 2º a A\_k...A\_2k-1, etc

95. Se dispone de un circuito integrado que actúa como módulo básico de memoria de 8K ×

4. ¿Qué circuitos necesitamos para construir una memoria de 16K × 8 para usar con un bus de direcciones de 14 bits?

a) 2 módulos de memoria

b) 2 módulos de memoria y un decodificador

c) 4 módulos de memoria

-> d) 4 módulos de memoria y un inversor

96. Una memoria que está organizada en palabras de 16 bits tiene una capacidad de 64 Kbits. ¿Cuántas palabras tiene?

a) 65536

b) 4000

-> c) 4096

d) 64000

97. Una puerta AND con 16 entradas conectada a un bus de direcciones de 16 bits, con todos los bits negados excepto A10 y A6, permite seleccionar un dispositivo (con CS activa en alta) en la dirección:

-> a) 0x0440

b) 0x0220

c) 0xFDDF

d) 0xFBFF

98. En la cache L1 de instrucciones, la tasa de fallos:

-> a) Siempre tiende a disminuir si el tamaño total de L1 crece

b) Siempre tiende a crecer si el tamaño total de L1 crece

c) Siempre tiende a disminuir si el número de vías disminuye

d) Siempre tiende a crecer si el número de vías crece

99. Las celdas de memoria estática...

- a) son más pequeñas y lentes que las celdas de memoria dinámica
  - b) almacenan la información en un condensador
- > c) mantienen las informaciones almacenadas por tiempo indefinido mientras se mantenga la alimentación
- d) Ninguna de las respuestas anteriores es cierta

100. ¿Cuál de las siguientes afirmaciones sobre memorias es correcta?

- a) La memoria principal se construye con tecnología electrónica de tipo SRAM.  
no, MP es DRAM
- >b) Los chips de memoria DRAM se conectan entre sí en un circuito impreso constituyendo lo que se denomina DIMM.
- c) Las memorias SRAM no son volátiles; es decir, cuando no están alimentadas eléctricamente siguen guardando toda la información.  
no, claro que no guardan info apagadas
- d) La memoria cache se construye con tecnología electrónica de tipo DRAM.  
no, cache es SRAM

## Preguntas SWAD (SOLO TEMAS)

### Tema 0

1 El temario de teoría es aproximadamente el siguiente...

- \* a) Intro, Nivel máquina, UC, Segm. cauce, E/S, Memoria
- b) Intro, Unidades Funcionales, circ. combinacionales, circ. secuenciales, diseño RTL
- c) Intro, Entrada, Salida, Unidad de Control, Memoria, ALU
- d) Intro, Entrada, Salida, Memoria, ALU, Unidad de Control

2 Esta asignatura es...

- a) continuación de TOC en 1º curso 1º cuatr. y previa a ISE en 3º curso 1º cuatr.
- \* b) continuación de TOC en 1º curso 2º cuatr. y previa a AC en 2º curso 2º cuatr.
- c) continuación de TOC en 1º curso 2º cuatr. y previa a ISE en 2º curso 2º cuatr.
- d) continuación de EC en 1º curso 2º cuatr. y previa a AC en 3º curso 1º cuatr.

3 El orden en que se cubren las asignaturas del área de ATC en el Plan de Estudios de grado GII es...

- a) EC, TOC, ISE, AC
- \* b) TOC, EC, AC, ISE
- c) TOC, EC, AS, IC
- d) ISE, EC, TOC, AC

4 En esta asignatura... (marcar la opción FALSA)

- \* a) Hay que obtener  $NTeo \geq 2.5p$  y  $NPra \geq 2.5p$ , para poder sumar teoría con prácticas y poder aprobar (obtener nota final no suspenso)
- b) se podrían acumular  $NC=2p$  y  $NL=2p$  respondiendo tests
- c) se pueden entregar ejercicios y prácticas, aunque cuentan muy poca o ninguna nota
- d) se podría obtener 10p en el examen final sin asistir un solo día ni a clase ni a prácticas

5 El temario de prácticas es aproximadamente el siguiente...

- a) Intro, progr. ASM, progr. C-ASM, depuración, microcontrolador, memoria
- b) Intro, popcount, media, bomba, cache
- c) cache, Arduino, bomba, imágenes, audio, Intro
- \*     d) Intro, circ. combinacionales, circ. secuenciales, bomba, Arduino, cache

### Tema 1

1 No en todas las instrucciones máquina hay una fase de

- a) decodificación
- b) ejecución
- c) captación
- \*     d) captura de operandos

2 ¿En qué generación, dentro de la historia de los computadores digitales, se alcanzaron tiempos de conmutación del orden de nanosegundos?

- a) primera
- b) segunda
- \*     c) tercera

d) cuarta

3 Si queremos almacenar la palabra de 16 bits 0x8965 en memoria según little-endian, quedará almacenada a partir de la posición 0x1000 como:

- a) en el byte 0x1000 se guarda 0xA6 y en el 0x1001 0x91
- b) en el byte 0x1000 se guarda 0x89 y en el 0x1001 0x65
- c) en el byte 0x1000 se guarda 0x91 y en el 0x1001 0xA6
- \* d) en el byte 0x1000 se guarda 0x65 y en el 0x1001 0x89

4 El espacio direccionable de memoria de un computador depende del diseño del:

- \* a) Bus de direcciones
- b) Bus de datos
- c) a) y b) son correctas
- d) Ninguna de las anteriores es correcta

5 Justo antes de que una instrucción máquina escriba un resultado en memoria:

- a) en IR está el resultado y en MAR la dirección donde se almacenará
- b) en IR está el resultado y en MBR la dirección donde se almacenará
- \* c) en MBR está el resultado y en MAR la dirección donde se almacenará
- d) en MAR está el resultado y en MBR la dirección donde se almacenará

6 El bus del sistema es

- a) en un sistema con bus único, todo el bus salvo la parte relacionada con E/S (SATA, GPU, USB, Ethernet, etc)
- \* b) el que conecta CPU-M, ya sea un sistema con bus único o con múltiples buses
- c) en un sistema con buses separados, el que conecta el sistema E/S con el resto
- d) el que conecta las distintas partes del sistema: UC, ALU, E/S, M

7 ¿Cuál de las siguientes direcciones está alineada a double (8-byte)?

(Al no poder escribir el 2 como subíndice, aclaramos que ")2" indica binario)

- a) 1110110101110111)2
- b) 1110110101110100)2
- \* c) 1110110101110000)2
- d) Ninguna de ellas

8 ¿En qué registro está contenido el último dato (o instrucción) leído de memoria, o el dato que se va a escribir en memoria?

- \* a) mbr/mdr
- b) registro de propósito general
- c) mar
- d) pc

9 En las instrucciones aritméticas con dos operandos de un procesador con arquitectura de pila, los dos operandos...

- a) son dos registros del procesador.
- b) pueden estar en cualquier posición de la pila.
- \* c) son la cima de la pila y el elemento siguiente de la cima de la pila.
- d) se introducen en la pila tras realizar la operación.

10 Si queremos almacenar la palabra de 64bits 0x00000001f ffffffe0 en una memoria de bytes según la convención little-endian a partir de la posición 0x0804913c, quedará

- a) 0x00 en 0x0804913c y 0xe0 en 0x08049143

- \* b) 0x1f en 0x0804913c y 0xe0 en 0x08049140
- c) 0xe0 en 0x0804913c y 0x1f en 0x08049140
- d) Todas las respuestas anteriores son incorrectas

11 ¿Qué novedad se desarrolló en la tercera generación de computadores?

- a) Los primeros lenguajes de programación de alto nivel
- b) Los microprocesadores RISC
- \* c) Los circuitos integrados
- d) Los microprocesadores CISC

12 Un computador que utilice el sistema big-endian, almacena el número 0x2143 a partir de la dirección 0 como:

- \* a) M[0]=0x21 y M[1]=0x43
- b) M[0]=0x43 y M[1]=0x21
- c) M[0]=0x12 y M[1]=0x34
- d) M[0]=0x34 y M[1]=0x12

13 ¿En qué registro está contenido el último dato (o instrucción) leído de memoria, o el dato que se va a escribir en memoria?

- \* a) MBR.
- b) MAR.
- c) Acumulador.
- d) PC.

14 Un modo de direccionamiento en el que se especifica un registro y una dirección de memoria cuyo contenido se suma al contenido del registro base para obtener la dirección efectiva, se conoce como:

- a) base con desplazamiento
- b) directo o absoluto
- c) indirecto a registro través de memoria
- \* d) ninguno de los anteriores

15 En una arquitectura RISC típica:

- a) la programación resulta mucho más simple que en una arquitectura CISC
- b) se usan pocas instrucciones de las disponibles en el conjunto de instrucciones
- \* c) suele usarse segmentación
- d) la UC es más compleja que en una arquitectura CISC

16 ¿Cuál de las siguientes expresiones representa un direccionamiento inmediato?

- a) 8(%ebp)
- b) %eax
- c) (%eax)
- \* d) \$0x400

17 Un sistema con direcciones de 8bits utiliza una puerta NAND conectada a las líneas A7...A5 para atacar la entrada CS# (activa baja) de un módulo de memoria. En el mapa de memoria las siguientes posiciones corresponderán a dicho módulo

- \* a) 0xe0 a 0xff
- b) 0x70 a 0x7f y 0xf0 a 0xff
- c) 0x00 a 0x0f y 0x80 a 0x8f
- d) 0x00 a 0x1f

18 ¿Cuál de las siguientes afirmaciones es incorrecta?

- \* a) El formato de una instrucción nos indica el significado de cada bit de la instrucción
- b) Todas las instrucciones deben tener operando fuente y operando destino
- c) Todas las instrucciones deben tener código de operación
- d) No siempre es necesario indicar la dirección de la siguiente instrucción

19 ¿Cuál es el contenido de una pila al terminar de ejecutarse la siguiente secuencia de operaciones push y pop?:

```
push #1  
push #2  
push #3  
pop a  
push #4  
pop a  
pop a  
    a) 1 y 2  
    b) 10  
*     c) 1  
    d) 1, 2, 3 y 4
```

20 Para obtener una única velocidad comparativa final, el benchmark SPEC CPU combina las ganancias en velocidad de ejecución de una serie de tests, respecto a un ordenador de referencia, usando...

- a) la media aritmética
- b) la mediana
- c) la moda
- \* d) la media geométrica

21 ¿Cuál de las siguientes características es posterior a la segunda generación de computadores?

- a) Lenguaje ensamblador.
- b) Transistor.
- c) Memoria de núcleos de ferrita.
- \* d) RISC.

22 Si en un bus de direcciones de 32 bits se decodifica parcialmente la dirección de un dispositivo de 32 posiciones usando 22 bits, ¿cuántas veces aparecerá repetido en el mapa de memoria?

- \* a) 32
- b) 16
- c) 10
- d) 1024

23 ¿En qué pareja de registros están el dato/instrucción que se leerá o escribirá en memoria, y la dirección de memoria?

- \* a) MBR y MAR
- b) MAR y ACUMULADOR
- c) IR y ACUMULADOR
- d) MBR y PC

24 ¿En qué pareja de registros están la dirección de memoria y el dato que se leerá o escribirá en memoria?

[T1.2ConBas]

[E20Ene] teo.16 - inspirada en [E12Feb] teo.18 - [E12Sep] teo.13

- a) pc y mar
- b) ir y pc
- \* c) mar y mdr/mbr
- d) mdr/mbr y pc

25 ¿En qué generación, dentro de la historia de los computadores digitales, aparece la segmentación de cauce?

- a) primera
- b) segunda
- \* c) tercera
- d) cuarta

26 El ancho de palabra de una memoria corresponde a:

- \* a) La cantidad de bits que caben en una sola posición
- b) El número que identifica únicamente cada posición de la memoria.
- c) La longitud del registro de direcciones de la memoria.
- d) El número de posiciones que la componen.

27 En una arquitectura de acumulador, la instrucción LOAD X:

- a) suma M(X) al acumulador
- \* b) transfiere el contenido de la posición de memoria X al acumulador
- c) transfiere el contenido del acumulador a la posición de memoria X
- d) transfiere el contenido del registro X a la memoria

28 ¿Cómo se almacenaría como palabra de 32 bits el número -128 en un sistema que utilice el criterio del extremo menor ("little endian")?

- a) posición 0: FF pos.1:FF pos.2: FF pos.3: 80
- \* b) 0:80 1:FF 2:FF 3:FF
- c) 0:00 1:01 2: 00 3:80
- d) Ninguna de las anteriores

29 En una máquina little-endian con memoria de bytes y representación en complemento a dos que permite accesos a memoria de tamaño byte (1 B), media palabra (2 B) y palabra (4 B), se almacenan a partir de la posición 0xCAFEBA0 cuatro palabras con valores -1, -2, -3, -4.

¿Qué se obtendría al consultar la media palabra de la posición 0xCAFEBABE?

- \* a) -1
- b) -4
- c) no se puede saber, faltan datos
- d) ninguna de las anteriores

30 Si almacenamos según el criterio little-endian la palabra de 64 bits 0xFACEB00C a partir de la dirección 0xCAFEBABE, el byte 0xCE quedará almacenado en la dirección:

- \* a) 0xCAFEBAC0
- b) 0xCAFEBAC1
- c) 0xCAFEBABF
- d) 0xCAFEBABE

31. Se pretende almacenar una palabra de 4 B en una memoria de bytes a partir de una dirección determinada. ¿Cuál de las siguientes es válida, si la palabra debe quedar alineada?

- a) 0xCAFEBABE

- \*     b) 0xDEADBEEF
- c) 0xFACEB00C
- d) 0xABADFOOD

32. Una dirección de memoria se refiere siempre a:

- a) una palabra
- b) 16 bits
- c) un byte
- \*     d) ninguna de las anteriores

33. El número -12 se almacenará en complemento a 2 en el registro %eax como:

- a) 0xFFOC
- b) 0xFFFFFFFFOC
- \*     c) 0xFFFFFFFF4
- d) 0xFFFF4

34. ¿Qué parámetro es más importante para comparar la velocidad de dos ordenadores diferentes?

- a) La arquitectura del procesador.
- \*     b) El resultado de la ejecución de un conjunto de programas de prueba.
- c) La frecuencia de reloj del procesador.
- d) El número medio de ciclos de reloj por instrucción.

35. ¿Cuál de las siguientes afirmaciones es falsa?

- \*     a) la anchura del bus de datos es de 16 bits
- b) el bus de control transporta señales de estado
- c) el bus de direcciones es unidireccional
- d) el bus de datos es bidireccional

36. En un sistema con un único bus...

- a) sólo un dispositivo puede escribir en un instante dado en el bus
- b) se utilizan las mismas líneas de control para conectar todos los dispositivos
- c) el procesador y los periféricos pueden funcionar a diferentes velocidades si el funcionamiento del bus es asíncrono
- \*     d) Todas las respuestas anteriores son ciertas

37. El instrumento GIADA de la sonda espacial ROSETTA (diseñado en Granada) está basado en un microprocesador 8086 y el siguiente mapa de memoria:

RAM volátil: 00000 - 0FFFF

RAM no volátil: 10000 - 1FFFF

ROM: F0000 - FFFFF

¿Cuál es el tamaño total de la memoria?

- a) 2MB
- \*     b) 192KB
- c) 48KB
- d) 3MB

38. Para obtener una única velocidad comparativa final, el benchmark SPEC CPU combina las velocidades de ejecución de una serie de tests, respecto a un ordenador de referencia, usando la media...

- \*     a) geométrica
- b) aritmética

- c) armónica
- d) ponderada

39. ¿Cuál de las siguientes características sobre RISC es \*FALSA\*?

- a) Para acelerar un procesador RISC se deberían emplear técnicas de segmentación.
- \* b) Las instrucciones máquina en un procesador RISC deberían ser complejas y potentes.
- c) La decodificación de las instrucciones debe ser simple: un procesador RISC debería emplear pocos formatos de instrucción.
- d) La unidad de control de un procesador RISC debería ser cableada, no microprogramada.

40. Respecto a los dispositivos activos y pasivos en un bus podemos decir que:

- a) Los dispositivos pasivos sólo pueden convertirse en esclavos
- b) Sólo los dispositivos activos pueden convertirse en maestros
- \* c) Las respuestas a y b son ciertas
- d) Las respuestas a y b son falsas

41. ¿Cuál es el contenido de la pila al terminar de ejecutarse la siguiente secuencia de instrucciones de una arquitectura de pila?:

```
push #4  
push #7  
add  
push #10  
sub  
*      a) 4, 7, 10  
      b) 11, 1  
      c) 4, 11, 1  
      d) 1
```

42. ¿Cuál de las siguientes afirmaciones sobre el direccionamiento directo es falsa?

- a) El objeto está en una posición de la memoria
- \* b) El tamaño del operando direccionado queda limitado por el nº de bits del campo de direccionamiento
- c) La instrucción contiene la dirección de memoria en la que se encuentra el objeto
- d) El rango de posiciones direccionables queda limitado por el tamaño del campo de direccionamiento

43. Se dice que las máquinas con arquitectura Von Neumann siguen un modelo de programa...

- \* a) Almacenado.
- b) Cableado.
- c) Microprogramado.
- d) Externo.

44. En la captación de un operando que reside en memoria:

- a) en MBR indicamos la dirección donde está y en MAR lo recogemos
- b) en MBR indicamos la dirección donde está y en IR lo recogemos
- c) en MAR indicamos la dirección donde está y en IR lo recogemos
- \* d) en MAR indicamos la dirección donde está y en MBR lo recogemos

45. Una CPU con bus de direcciones de 64 bits y bus de datos de 32 bits tiene un registro de 64 bits conectado al bus de direcciones de la memoria. Probablemente se trata del registro

- \* a) MBR
- b) MAR
- c) Acumulador
- d) IR

46. En una máquina little-endian con memoria de bytes y representación en complemento a dos que permite accesos a memoria de tamaño byte (1B), media palabra (2B) y palabra (4B), si se almacena en la posición 0xBABC una palabra de valor -2, ¿qué se obtendría al consultar la media palabra en la posición 0xBABE?

- \* a) -1
- b) 0
- c) -2
- d) 1

47. En el direccionamiento inmediato, tras captarse completamente la instrucción:

- a) se accede al operando, que está contenido en una posición de memoria principal.
- b) el código de operación contiene el operando.
- c) se accede al operando, que se encuentra almacenado en uno de los registros programables.
- \* d) se accede al operando, que es una constante contenida en la propia instrucción.

48. Para direccionar una memoria de bytes en la que quepan 1G palabras de 32 bits se necesitarán:

- a) 33 bits
- b) 21 bits
- c) 31 bits
- \* d) 32 bits

49. ¿Cuál de las siguientes no es una unidad de la arquitectura Von Neumann?

- a) Memoria principal
- b) Sistema de entrada/salida
- \* c) Núcleo del sistema operativo
- d) Unidad central de proceso

50. Un datapath con bus de direcciones de 32 bits y bus de datos de 16 bits tiene un registro de 16 bits conectado al bus de datos y a la unidad de control. Puede tratarse del registro

- a) MAR
- b) PC
- \* c) IR
- d) SP

51. ¿Cuál de las siguientes direcciones NO está alineada a double (8-byte)?

- \* a) 1110110101110100)2
- b) 1110110101101000)2
- c) 1110110101110000)2
- d) Todas están alineadas a double

52. En las últimas generaciones de computadores la mejora de prestaciones viene dada por:

- \* a) avances en las tecnologías de fabricación.
- b) avances en los sistemas operativos y aplicaciones.
- c) avances en tecnología y avances en la estructura y arquitectura del computador.
- d) avances en la estructura y arquitectura del computador.

53. Una memoria que está estructurada en palabras de 8 bits tiene una capacidad de 64 Kbits. ¿Cuántas líneas de dirección tiene dicha memoria?

- a) 8
- b) 24
- \* c) 13
- d) 12

54. El conjunto de todos los atributos de un sistema que son visibles para el programador y son necesarios para programar en lenguaje máquina se denomina:

- \* a) arquitectura del computador
- b) conjunto de componentes físicos del computador
- c) repertorio de instrucciones máquina
- d) organización del computador

55. ¿Cuál de los siguientes registros se utiliza para guardar la dirección de memoria donde se localiza la instrucción siguiente?

- a) Memory Data Register
- b) Instruction Register
- c) Memory Address Register
- \* d) Program Counter

56. ¿Cuál de las siguientes afirmaciones es incorrecta?

- a) En el direccionamiento inmediato el dato se encuentra en la propia instrucción
- b) En el direccionamiento implícito no se indica la ubicación del operando
- c) El direccionamiento indexado es útil para manejo de vectores
- \* d) El direccionamiento indirecto indica la dirección del operando

57. ¿Cuál es la característica tecnológica principal de la segunda generación de computadores?

- a) La gran integración de los circuitos (VLSI)
- b) Las válvulas
- c) Los circuitos integrados
- \* d) Los transistores

58. Respecto a la ecuación de rendimiento  $T=(N \cdot S)/R$ , el objetivo de un diseño CISC es:

- a) aumentar S (número medio de ciclos por instrucción)
- b) disminuir R (frecuencia de reloj)
- c) aumentar R (frecuencia de reloj)
- \* d) disminuir N (número de instrucciones a ejecutar por el programa)

59. ¿En qué generación, dentro de la historia de los computadores digitales, aparece la memoria cache?

- a) primera
- b) segunda
- \* c) tercera
- d) cuarta

60. El registro MBR..

- a) especifica la dirección en memoria de la palabra que va a ser escrita o leída
- b) contiene la dirección de la próxima instrucción que va a ser captada de memoria
- c) contiene el código de operación de la instrucción que se está ejecutando

- \* d) contiene el valor que va a ser almacenado en la memoria, o bien se usa para recibir un valor procedente de la memoria

61. Si queremos almacenar la palabra de 16 bits 0x9660 en una memoria de bytes según "little-endian", quedará almacenada a partir de la posición 0x1000 como:

- \* a) M[0x1000]=0x60 y M[0x1001]=0x96
- b) M[0x1000]=0x69 y M[0x1001]=0x06
- c) M[0x1000]=0x06 y M[0x1001]=0x69
- d) M[0x1000]=0x96 y M[0x1001]=0x60

62. En la ejecución de una instrucción...

- a) siempre se altera el registro de estado
- b) el registro de instrucción se va incrementando para apuntar a la siguiente instrucción
- c) la UC activa las señales de control que envía por el bus de direcciones
- \* d) la ALU realiza las operaciones aritméticas y lógicas

63. Una máquina superescalar es aquella que:

- a) basa su funcionamiento en la segmentación software como forma de incrementar el paralelismo.
- b) las instrucciones tienen un campo por cada unidad funcional al realizarse varias operaciones por instrucción.
- \* c) emite simultáneamente múltiples instrucciones por ciclo de reloj, por ejemplo, una entera y otra de coma flotante.
- d) ninguna respuesta de las anteriores es correcta.

64. Escoger de entre las 4 operaciones la de mayor valor que pueda calcularse con enteros de 4B con signo sin problemas

- a) 100.000.000 + 100.000.000
- b) 3.000.000.000 + 3.000.000.000
- c) 300.000.000 + 300.000.000
- \* d) 1.000.000.000 + 1.000.000.000

65. Un bus se compone de:

- \* a) líneas de control/estado, líneas de dirección y líneas de datos
- b) líneas de estado y líneas de control
- c) líneas de datos y líneas de dirección
- d) líneas de alimentación

66. Un modo de vídeo de 512 x 256 píxeles y 16 colores por píxel ocupa una memoria de:

- a) 2 MB
- b) 8 KB
- c) 128 KB
- \* d) 64 KB

67. El direccionamiento directo a memoria utiliza

- \* a) un desplazamiento
- b) un registro
- c) dos desplazamientos contenidos en la propia instrucción
- d) un registro y un desplazamiento contenidos en la propia instrucción

68. En la captación de la instrucción:

- \*        a) en MAR indicamos la dirección donde está la instrucción y en MBR recogemos la instrucción.
- b) en MBR indicamos la dirección donde está la instrucción y en la ALU recogemos la instrucción.
- c) en MAR indicamos la dirección donde está la instrucción y en la ALU recogemos la instrucción.
- d) en MBR indicamos la dirección donde está la instrucción y en MAR recogemos la instrucción.

69. Una CPU con bus de direcciones de 16 bits y bus de datos de 8 bits tiene un registro de 8 bits conectado al bus de datos y a la unidad de control. Puede tratarse del registro

- \*        a) Puntero de pila
- b) De instrucción
- c) De direcciones
- d) Contador de programa

70. En el contexto del lenguaje máquina, el acrónimo ISA suele referirse a:

- a) Intel Standard Architecture
- b) Industry Standard Architecture
- c) Information Security Architecture
- \*        d) Instruction Set Architecture

71. En un sistema con dos buses separados, uno para el subsistema de memoria y otro para la E/S...

- \*        a) el bus que une la memoria y el procesador suele funcionar a la velocidad de la memoria
- b) el bus de E/S funciona a la velocidad del periférico más rápido
- c) ambos buses tienen que tener el mismo ancho de banda
- d) Ninguna de las respuesta anteriores es cierta

72. Si un computador X ejecuta un programa de 450 millones de instrucciones en 26 segundos y un computador Y tarda 14 segundos en ejecutar ese mismo programa, ¿cuántas veces es más rápido el computador Y que el X?

- a) 1,538
- b) 12
- c) 0,538
- \*        d) 1,857

73. ¿Cuál de las siguientes no es una característica de los procesadores RISC?

- a) Un computador RISC no debe emplear microprogramación.
- b) La decodificación de las instrucciones debe ser simple: un computador RISC debería emplear un único formato de instrucción
- \*        c) Las funciones que realizan los computadores RISC deben ser lo más complejas y potentes que sea posible.
- d) Para acelerar el computador RISC se emplean técnicas de pipelining.

74. Una instrucción máquina del tipo "Add M,R" podría formar parte del repertorio de

- a) una máquina de acumulador
- b) una máquina pila
- c) una máquina con arquitectura R/R
- \*        d) una máquina con arquitectura M/M

75. ¿Cuál de las siguientes afirmaciones es verdadera?

- a) La arquitectura Von Neumann en la que se basan los computadores tradicionales consiste en tener los datos separados de las instrucciones en memorias distintas.
- b) El registro de estado es un registro transparente al usuario, ya que éste no puede utilizarlo en las instrucciones máquina.
- \* c) El registro de instrucción es un registro transparente al usuario, ya que éste no puede utilizarlo en las instrucciones máquina.
- d) La unidad de control necesita como entrada el registro contador de programa, para saber cuál es la instrucción que debe ejecutar a continuación.

76. ¿Cuál de los siguientes microprocesadores NO es de 64 bits?

- a) Core i3
- b) Core i7
- \* c) AVR ATMega
- d) Itanium

77. Si queremos almacenar la palabra de 16 bits 0x8965 en una memoria de bytes según "big-endian", quedará almacenada a partir de la posición 0x1000 como:

- a) M[0x1000]=0xA6 y M[0x1001]=0x91
- \* b) M[0x1000]=0x89 y M[0x1001]=0x65
- c) M[0x1000]=0x91 y M[0x1001]=0xA6
- d) M[0x1000]=0x65 y M[0x1001]=0x89

78. ¿Cuál de las siguientes características es posterior a la segunda generación de computadores?

- a) Memoria de núcleos de ferrita
- b) Compilador
- \* c) Memoria cache
- d) Lenguaje ensamblador

79. Un computador con 8 bits en el bus de direcciones puede direccionar como máximo:

- \* a) 256 palabras
- b) 1024 palabras
- c) 16384 palabras
- d) 8192 palabras

80. Son funciones de la unidad de control:

- a) la codificación de las instrucciones máquina
- b) la lectura de memoria principal de la instrucción apuntada por el µPC
- \* c) el secuenciamiento de las instrucciones máquina
- d) todas las respuestas son ciertas

81. ¿Cuál es el valor mínimo (más negativo) que puede tomar un entero de 32 bits en complemento a dos?

- \* a)  $-2^{32}$
- b)  $-2^{31}$
- c)  $-2^{31} + 1$
- d)  $-2^{32} + 1$

82. La idea de desarrollar máquinas CISC surgió para:

- a) simplificar el diseño hardware de la UC.
- \* b) tener instrucciones cercanas al lenguaje de alto nivel.

- c) conseguir un conjunto de instrucciones cortas y sencillas de decodificar.
- d) ninguna de las respuestas anteriores es cierta.

83. ¿Cuál de las siguientes afirmaciones es cierta?

- \* a) el registro de estado (flags) es un registro de propósito específico cuyo contenido puede ser visto directa o indirectamente por el usuario mediante el uso de ciertas instrucciones específicas
- b) la unidad de control necesita como entrada el registro contador de programa para saber cuál es la instrucción que debe ejecutar a continuación
- c) el registro de direcciones de memoria es un registro de propósito general que puede contener tanto direcciones como datos
- d) la arquitectura von Neumann de los computadores tradicionales consiste en tener almacenados los datos separados de las instrucciones en memorias distintas

84. ¿En qué generación, dentro de la historia de los computadores digitales, aparece la memoria virtual?

- a) primera
- b) segunda
- \* c) tercera
- d) cuarta

85. La primera generación de computadores se caracteriza por el uso de:

- a) Fibra óptica.
- \* b) Tubos de vacío.
- c) Transistores.
- d) Microporcesadores.

86. Según la clasificación m/n, las máquinas de acumulador son de tipo

- a) 1/2
- b) 0/0
- \* c) 1/1
- d) 2/2 ó 2/3

87. ¿Cuál es el complemento a 2 del número binario 1110 1101 1000?

- a) 0001 0010 0111
- \* b) 0001 0010 1000
- c) 0001 0010 0110
- d) 0001 0010 0101

88. ¿Cuál de los siguientes no es un tipo de bus?

- a) Paralelo
- b) E/S
- \* c) Secuencial
- d) Sistema

89. Un computador con 13 líneas de direcciones utiliza E/S mapeada a memoria. Si se supone que cada uno de los periféricos ocupa 4 direcciones y que el número de periféricos que se planea conectar es de  $2^{10}$ , ¿qué tamaño de memoria admite el computador?

- a)  $2^{10}$  palabras
- \* b)  $2^{12}$  palabras
- c)  $2^{13}$  palabras

d) Ninguna de las anteriores

90. El objetivo de un diseño CISC es...

- a) disminuir el número medio de ciclos por instrucción.
- b) disminuir la frecuencia de reloj.
- c) disminuir el tamaño medio de instrucción.
- \* d) disminuir el número de instrucciones a ejecutar por un programa.

91. ¿Cuál es el contenido de la pila al terminar de ejecutarse la siguiente secuencia de instrucciones de una arquitectura de pila?:

```
push #4  
push #7  
push #8  
add  
push #10  
sub  
mul
```

- \* a) 20
- b) 4, 7, 48
- c) 4, 7, 8, 10
- d) 4

92. ¿Cuál de las siguientes afirmaciones es incorrecta?

- a) al bus del sistema sólo se conecta la CPU y la MP, ya que el DMA se conecta directamente a MP para realizar las transferencias de datos.
- \* b) la CPU puede dejar las transferencias entre MP y periféricos en manos de este controlador (DMA) y seguir ejecutando otras instrucciones.
- c) podemos prescindir de controladores de E/S ya que el controlador de DMA se ocupa de controlar las transferencias hacia/desde los periféricos.
- d) la velocidad de este controlador establece la velocidad del bus del sistema.

93. ¿En qué generación, dentro de la historia de los computadores digitales, aparece la microprogramación?

- a) primera
- b) segunda
- \* c) tercera
- d) cuarta

94. En una memoria de bytes que contuviera a partir de la posición 0 los valores 1,0,0,0,0xFE,0xFF,0xFF,0xFF, se puede decir que...

- a) Hay una palabra de 16bit big-endian con valor 1 en la posición 0
- b) Hay una palabra de 16bit little-endian con valor 254 en la posición 3
- c) Hay una palabra de 32bit little-endian con valor -1 en la posición 4
- \* d) Todas las respuestas anteriores son incorrectas

95. En una arquitectura de registros de propósito general (a nivel de lenguaje máquina):

- a) operar usando registros es más rápido.
- b) la generación de código resulta más simple que en arquitecturas de pila o acumulador.
- c) se evita el cuello de botella (por ejemplo, pila, o acumulador) que otras arquitecturas presentan al evaluar expresiones aritméticas complejas

- \*       d) todas las respuestas anteriores son ciertas.

96. ¿En qué generación, dentro de la historia de los computadores digitales, aparecieron la microprogramación, la segmentación de cauce, la memoria cache, los S.O. multiusuario y la memoria virtual?

- a) 2<sup>a</sup> generación (1955-65)
- \*       b) 3<sup>a</sup> generación (1965-75)
- c) 4<sup>a</sup> generación (1975-85)
- d) esas innovaciones se repartieron a lo largo de varias generaciones, no sólo una

97. Según la clasificación m/n, las máquinas con arquitectura R/R son de tipo

- a) 0/0
- \*       b) 0/x con x=2,3
- c) x/x con x=2,3
- d) x/0 con x=2,3

98. ¿Qué tipo de instrucciones se emplean más en una arquitectura de acumulador?

- a) aritmético-lógicas
- \*       b) de transferencia de datos con memoria
- c) de desplazamiento y rotación
- d) de transferencia de datos entre registros

99. El programador de lenguaje ensamblador necesita conocer:

- a) la microarquitectura del procesador.
- \*       b) la arquitectura del ordenador.
- c) el diseño RTL del procesador.
- d) todas las anteriores son ciertas.

100. ¿En qué generación, dentro de la historia de los computadores digitales, aparecen los sistemas operativos multiusuario?

- a) primera
- b) segunda
- \*       c) tercera
- d) cuarta

101. El direccionamiento relativo a registro base utiliza...

- \*       a) un registro y un desplazamiento.
- b) dos registros.
- c) un registro y un factor de escala
- d) dos desplazamientos contenidos en la propia instrucción.

102. ¿Cuál de los siguientes elementos no forma parte de la Arquitectura del Repertorio de Instrucciones (ISA)?

- a) Descripción de los registros de datos, registros de estado y control.
- \*       b) Descripción de los campos de bits en los que están organizadas conceptualmente las microinstrucciones.
- c) Descripción del espacio de direccionamiento de la memoria y de la E/S.
- d) Descripción de los tipos de datos sobre los que opera el lenguaje máquina.

103. En una estructura de computador de bus único (bus del sistema):

- a) es la estructura más usada en los PC actuales
- \*       b) sólo una unidad funcional puede tener el control del bus en cada momento

- c) la UC concede el acceso al bus, por lo que éste funciona a la velocidad de la CPU
- d) es necesario el arbitraje entre los maestros potenciales, no es suficiente la técnica de robo de ciclo ni otras similares.

104. En un procesador de la familia 80x86 una variable de 32 bits, entera con signo, almacenada a partir de la dirección n contiene: 0xFF en la dirección n, 0xFF en la dirección n+1, 0xFF en la dirección n+2 y 0xF0 en la dirección n+3. ¿Cuánto vale dicha variable?

- a) 4294967280
- \* b) -251658241
- c) -16
- d) 16

105. Para direccionar una memoria de bytes en la que quepan 2G palabras de 32 bits se necesitarán:

- a) 32 bits exactamente
- \* b) 33 bits como mínimo
- c) 21 bits como máximo
- d) 31 bits como mínimo

106. ¿Cuál de las siguientes definiciones de modos de direccionamiento es \*incorrecta\*?

- a) Inmediato: el dato está codificado dentro de la propia instrucción, en uno de los campos en los que se divide el formato de instrucción.
- b) Indirecto: el dato está contenido en una posición de memoria que es apuntada por un registro de propósito general.
- c) Registro: el dato se encuentra en un registro de propósito general.
- \* d) Directo: la dirección se calcula como la suma de un dato codificado en la propia instrucción y el contenido de un registro de propósito general.

107. En una CPU de 32 bits con memoria de bytes, el problema es que...

- a) No hay problema, cuando se salva un registro a memoria se escribe en la posición deseada
- \* b) Hay que respetar el ordenamiento de bytes y reglas de alineamiento con que se diseñó la CPU
- c) Hay que usar 4 instrucciones de lectura (o escritura) para leer (o escribir) un registro completo
- d) No tiene sentido, un registro no cabría en memoria

108. En el direccionamiento indirecto a través de registro, la dirección efectiva...

- a) se encuentra en una dirección de memoria.
- b) se calcula como la suma del contenido de dos registros.
- \* c) se encuentra en un registro general del procesador.
- d) se encuentra en el registro de instrucción.

109. El registro MBR...

- a) contiene el código de operación de la instrucción que se está ejecutando
- b) especifica la dirección en memoria de la palabra que va a ser escrita o leída
- c) contiene la dirección de la próxima instrucción que va a ser captada de memoria
- \* d) contiene el valor que va a ser almacenado en la memoria, o bien se usa para recibir un valor procedente de la memoria

110. ¿Qué arquitectura es típica en procesadores RISC?

- \* a) registro-registro

- b) registro-memoria
- c) memoria-registro
- d) memoria-memoria

111. En las arquitecturas RISC hay...

- a) pocos registros y muchos tipos de instrucciones.
- \* b) muchos registros y pocos modos de direccionamiento.
- c) pocas instrucciones muy rápidas con muchos modos de direccionamiento.
- d) pocos modos de direccionamiento y muchos formatos de instrucción.

112. ¿De qué depende el tamaño del contador de programa?

- a) de la longitud del código de operación
- b) del ancho del bus de datos
- c) el tamaño no importa
- \* d) ninguna de las anteriores es cierta

113. ¿Qué tipo de direccionamiento se usa para el registro destino en la instrucción x86-64 add array(%rbx,4), %edx?

- a) Direccionamiento relativo a registro base
- b) Direccionamiento indexado
- c) Direccionamiento inmediato
- \* d) Direccionamiento a registro

114. Si queremos almacenar la palabra de 16 bits 0x8965 en una memoria de bytes según "little-endian", quedará almacenada a partir de la posición 0x8600 como:

- a) M[0x8600]=0x69 y M[0x8601]=0x85
- b) M[0x8600]=0x89 y M[0x8601]=0x65
- \* c) M[0x8600]=0x65 y M[0x8601]=0x89
- d) M[0x8600]=0x85 y M[0x8601]=0x69

115. ¿Cuál de las siguientes afirmaciones es incorrecta?

- \* a) El tamaño de una instrucción en lenguaje máquina siempre ocupa dos bytes en los procesadores RISC.
- b) Las arquitecturas RISC simplifican la decodificación.
- c) En las arquitecturas CISC hay más instrucciones que en las RISC.
- d) Las arquitecturas RISC son del tipo registro-registro.

116. En el arbitraje de un bus...

- a) los dispositivos pasivos pueden requerir el uso del bus para iniciar una transferencia
- \* b) si hay un único dispositivo pasivo, siempre funciona como esclavo
- c) si hay varios dispositivos activos, siempre funcionan como maestros
- d) todas las respuestas anteriores son ciertas

117. En la memoria de un procesador de la familia Intel 64 (x86-64) se almacena a partir de la dirección N los siguientes contenidos: 0xff, 0xff, 0xff, 0xff, 0x00, 0x00, 0x00, 0x01, 0xff, 0xff, 0xff, 0x00, 0x00, 0x00, 0x00, 0x02. Posteriormente se lee (carga) %rax desde (a partir de) la dirección N. ¿Cómo es %rax entonces, considerándolo con signo?

- a) menor que -2000 millones
- b) entre -2000 millones y -1
- c) entre 0 y 2000 millones
- \* d) mayor que 2000 millones

118. Un procesador con E/S mapeada a memoria tiene un bus de direcciones de 10 líneas y uno de datos de 8. El mapa de memoria tiene 512 posiciones para código (ROM), 256 para datos (RAM) y 256 para E/S, en ese orden. Los rangos de direcciones para esas tres zonas serán:

- \* a) 000 a 7FF, 800 a BFF y C00 a FFF
- b) 000 a 1FF, 200 a 2FF y 300 a 3FF
- c) 000 a 5FF, 600 a 7FF y 800 a 9FF
- d) 000 a 9FF, A00 a CFF y D00 a FFF

119. Si N es el número de instrucciones máquina de un programa, F es la frecuencia de reloj, y C el número promedio de ciclos por instrucción, el tiempo de ejecución del programa será:

- \* a)  $N \cdot C / F$
- b)  $N \cdot F \cdot C$
- c)  $N / (F \cdot C)$
- d)  $N \cdot F / C$

120. En un procesador de la familia 80x86 las posiciones de memoria que representan una variable int (entero 4B compl.2) contiene los bytes: F0 FF FF FF. ¿Cuánto vale dicha variable?

- a) 4294967280
- b) 4043309055
- c) 16
- \* d) -16

121. ¿Cuál de las siguientes afirmaciones es incorrecta?

- \* a) Los operandos siempre están almacenados en memoria.
- b) El repertorio de instrucciones debe ser capaz de realizar una tarea en un tiempo finito.
- c) El modo de direccionamiento permite determinar un operando o la ubicación del operando.
- d) El repertorio de instrucciones es el conjunto de operaciones que es capaz de interpretar la unidad de control.

122. ¿De qué tipo son los procesadores Intel que usamos en los laboratorios?

- a) big-endian
- b) el concepto de endian no es aplicable a estas máquinas, ya que un registro del procesador no cabe en una posición de memoria
- c) puede ajustarse mediante un bit de control en el registro CR0 que funcionen como little- o big-endian
- \* d) little-endian

123. Si usamos una estructura de bus con DMA:

- \* a) la CPU puede dejar las transferencias entre MP y periféricos en manos de este controlador (DMA) y seguir ejecutando otras instrucciones.
- b) podemos prescindir de controladores de E/S ya que el controlador de DMA se ocupa de controlar las transferencias hacia/desde los periféricos.
- c) al bus del sistema sólo se conecta la CPU y la MP, ya que el DMA se conecta directamente a MP para realizar las transferencias de datos.
- d) la velocidad de este controlador establece la velocidad del bus del sistema.

124. ¿Cuál de las siguientes no es una característica de los computadores RISC?

- a) Para acelerar el computador RISC se emplean técnicas de pipelining.
- b) Un computador RISC no debe emplear microprogramación.

- c) Las funciones que realizan los computadores RISC deben ser lo más complejas y potentes que sea posible.
- d) La decodificación de las instrucciones debe ser simple: un computador RISC debería emplear un único formato de instrucción
125. En la arquitectura Von Neumann...
- a) los bloques principales son la unidad de control, la ALU y la CPU.
  - \* b) el programa se encuentra residente en memoria.
  - c) los registros se encuentran en la memoria principal.
  - d) Todas son ciertas.
126. ¿Cuál es la característica tecnológica principal de la tercera generación de computadores?
- a) Las válvulas
  - b) La gran integración de los circuitos (VLSI)
  - c) Los transistores
  - \* d) Los circuitos integrados
127. ¿Cómo se representa el valor -1 como entero con signo en 14 bits?
- a) 0xFFFF
  - \* b) 0x3FFF
  - c) las respuestas anteriores no son válidas porque usan hexadecimal; habría que usar binario
  - d) no se puede porque 14 no es múltiplo de 4
128. ¿Cuál de las siguientes afirmaciones es \*incorrecta\*?
- a) El direccionamiento indirecto indica un puntero al operando
  - b) En el direccionamiento inmediato el dato se encuentra en la propia instrucción
  - \* c) El direccionamiento indexado es útil para manejo de estructuras
  - d) En el direccionamiento implícito no se indica la ubicación del operando
129. ¿Cuál de las siguientes afirmaciones sobre el benchmark SPEC CPU es falsa?
- a) La última versión es SPEC CPU2017
  - b) Se cronometran unos 10 tests de enteros y unos 14 tests de punto flotante
  - c) Se usa como referencia un computador UltraSPARC-IV+ 2100MHz, y para cada test se calcula una velocidad cociente entre el tiempo de ejecución en el computador de referencia y en el computador a testear
  - \* d) El resultado final es la media aritmética de las (10 ó 14) velocidades, bien sea de enteros ó de punto flotante (SPECspeedINT ó SPECspeedFP)
130. ¿Por qué se impusieron las arquitecturas de registros de propósito general a las arquitecturas basadas en pila?
- \* a) Porque las basadas en registros son capaces de lograr un mejor rendimiento cuando se asignan variables a registros
  - b) Porque las basadas en registros permiten reducir el tamaño del programa
  - c) Porque no se puede programar una arquitectura de pila en un lenguaje de alto nivel
  - d) Porque la memoria es más cara que los registros
131. El direccionamiento directo a memoria utiliza...
- a) un registro y un desplazamiento contenidos en la propia instrucción.
  - b) un registro.
  - \* c) un desplazamiento.
  - d) dos desplazamientos contenidos en la propia instrucción.

## Tema 2

1. La instrucción movq %rsp,%rbp
  - \* a) Copia el contenido del registro RSP en el registro RBP.
  - b) Introduce en la pila el contenido del registro RBP.
  - c) Intercambia los contenidos de los registros RSP y RBP.
  - d) Mueve el contenido del registro RSP al registro RBP, poniendo a 0 el registro RSP.
2. Estudiando el listado de una función C presuntamente compilada con gcc en modo 64bit (x86-64), nos dicen que la primera instrucción, movl %eax, (%rdi), carga en EAX el valor adonde apunta el primer argumento.
  - \* a) Está mal, porque EAX no se carga con ningún valor
  - b) Está mal, porque EAX no se puede usar en modo 64bit, debería ser RAX
  - c) Está mal, porque el primer argumento de una función C no se pasa en RDI
  - d) Está bien, y pone a cero los 32 bits más significativos de RAX
3. De entre las siguientes construcciones de flujo de control en lenguaje C, la que se traduce más directamente a lenguaje ensamblador es...
  - a) El bucle while
  - \* b) El bucle do-while
  - c) La selección switch-case
  - d) El bucle for
4. ¿Qué modo de direccionamiento usa el operando fuente en la instrucción mov (%rcx), %al?
  - a) Directo a memoria
  - b) Registro
  - c) Inmediato
  - \* d) Indirecto a memoria a través de registro
5. En x86-64 la pila es:
  - \* a) un conjunto de posiciones de memoria usadas para almacenar información temporal durante la ejecución del programa
  - b) un registro de 32 bits en el microprocesador
  - c) una dirección de memoria de 64 bits almacenada en el contador de programa
  - d) un registro de 64 bits en el microprocesador
6. Una instrucción de "salto si igual" tiene que comprobar el valor de:
  - \* a) el bit de cero
  - b) el bit de signo
  - c) el bit de acarreo
  - d) los bits de signo y desbordamiento
7. En un sistema de 32bits, ¿cuál de las siguientes expresiones C es equivalente a la expresión  $(x[2] + 4)[3]$ ?
  - \* a)  $\ast((\ast(x + 2)) + 7)$
  - b)  $\ast((\ast x) + 2) + 7$
  - c)  $\ast((\ast(x + 8)) + 28)$
  - d)  $(\ast\ast(x + 2) + 7)$
8. ¿Cuál de las siguientes secuencias de instrucciones multiplica %rax por 10?

a) leaq (%rax,%rax,4), %rax  
 salq \$2, %rax  
 b) imulq \$0x10, %rax  
 c) addq %rax, %rax  
 shlq \$5, %rax  
 \* d) Varias o ninguna de las respuestas anteriores son correctas, no se puede marcar una y sólo una

9. Dada la siguiente declaración en lenguaje C, una estructura de este tipo podría ocupar, bien sea en un sistema Linux IA32 o bien en uno x86-64, un total de...

```
struct a{
  int i;
  double d;
  char c;
  short s; };
    a) 18 B
    b) 20 B
    c) 22 B
    * d) 24 B
```

10. ¿Cuál de las siguientes listas está correctamente ordenada temporalmente?

- \* a) 8086, 486, Pentium MMX, Pentium III, Pentium 4, Core 2
- b) 486, 8086, Pentium MMX, Core 2, Pentium III, Pentium 4
- c) 486, 8086, Core 2, Pentium III, Pentium 4, Pentium MMX
- d) 8086, 486, Pentium III, Pentium MMX, Core 2, Pentium 4

11. El primer microprocesador de 32 bits de la familia x86 fue el:

- a) Pentium
- \* b) 80386
- c) 80286
- d) 80486

12. Un procedimiento llamado por una instrucción call debe guardar y restaurar los registros siguientes siempre que los altere:

- a) %rsi, %rdi
- \* b) %rbx, %rbp
- c) %rax, %rbx, %rcx, %rdx
- d) %rax, %rdx, %rcx

13. Una función C declarada como int get\_var\_digit(size\_t index, size\_t digit) genera como código ensamblador

```
leaq (%rdi,%rdi,4), %rax
addq %rax, %rsi
movl var(%rsi,4), %eax
ret.
```

Se puede adivinar que:

- a) var es un array multi-nivel (punteros a enteros) de cuatro filas
- b) var es un array multi-nivel pero no se pueden adivinar las dimensiones
- c) var es un array bidimensional de enteros, no se pueden adivinar dimensiones
- \* d) var es un array bidimensional de enteros, con cinco columnas

14. En la secuencia de programa siguiente:

```
400544: e8 07 00 00 00 callq 400550 <mult2>
400549: 48 89 03          mov %rax,(%rbx)
```

¿cuál es el valor que introduce en la pila la instrucción call?

- a) 0x400550
- b) 0x40054b
- c) 0x400544
- \* d) 0x400549

15. Respecto a registros base e índice en x86-64, la excepción es que

- a) RBP no puede ser registro índice
- b) RSP no puede ser registro base
- \* c) RSP no puede ser registro índice
- d) RBP no puede ser registro base

16. Si RBX=0x00002a00 y RDI=0x0000000a, ¿cuál es la dirección efectiva en la instrucción add 0x64(%rbx,%rdi,2),%eax?

- a) 0x00002a70
- b) 0x000054dc
- c) 0x00005478
- \* d) 0x00002a78

17. Si declaramos int val[5]={1,5,2,1,3}; entonces

- a) val[5] es de tipo int y vale 3
- b) val+4 es de tipo int\* y se cumple que \*(val+4)==5
- \* c) &val[2] es de tipo int\* y vale lo mismo que (void\*)val+8
- d) val+1 es de tipo int y vale 2

18. Respecto a registros salva-invocante y salva-invocado en GCC/Linux x86-64, ¿cuál de éstos es de distinto tipo que el resto?

- a) RAX
- \* b) RBX
- c) RCX
- d) RDX

19. Si AX = 0xFA50 y ejecutamos AND \$0xFF, %AX

- a) El registro AL se pone a FF
- \* b) El registro AH se pone a 0
- c) El registro AH se pone a FF
- d) El registro AL se pone a 0

20. Para comprobar si el entero almacenado en EAX es cero (y posiblemente saltar a continuación usando JZ/JNZ), gcc genera el código

- a) cmp %eax, \$0
- b) cmp %eax
- \* c) test %eax, %eax
- d) test %eax

21. ¿Cuál es el popcount (peso Hamming, nº de bits activados) del número 29?

- \*      a) 2
- b) 3
- \*      c) 4
- d) 5

22. En el fragmento de programa siguiente:

```
66b: e8 8a ff ff ff callq 5fa <f>
670: 48 83 c4 10    add $0x10,%rsp
```

la instrucción callq suma al contador de programa la cantidad:

- a) 0x76
- b) 0x5fa
- \*      c) -0x76
- d) 0xffffffff8a

23. Despues de ejecutar una instrucción de suma sobre dos números con signo de la que sabemos que no provocará overflow (los dos números son pequeños en valor absoluto), queremos comprobar si el resultado de la suma es menor que 0. ¿Qué flag necesita comprobar la instrucción de salto condicional equivalente a... ?

if (resultado<0) then goto label

- \*      a) SF
- b) CF
- c) ZF
- d) OF

24. Respecto a registros salva-invocante y salva-invocado en GCC/Linux IA32, ¿cuál de éstos es de distinto tipo que el resto?

- \*      a) EBX
- b) ECX
- c) EAX
- d) EDX

25. Respecto a direccionamiento a memoria en ensamblador x86-64 (sintaxis AT&T), de la forma D(Rb, Ri, S), sólo una de las siguientes afirmaciones es FALSA. ¿Cuál?

- a) RSP no se puede usar como registro índice
- \*      b) RBP no se puede usar como registro base
- c) El desplazamiento D puede ser una constante literal (1, 2 ó 4 bytes)
- d) El factor de escala S puede ser 1, 2, 4, 8

26. El cuerpo del siguiente código C:

```
unsigned copy(unsigned u) {return u;}
```

puede traducirse a ensamblador como:

- \*      a) mov %edi, %eax
- b) mov %rdi, (%rax)
- c) mov %eax, %esi
- d) movl 8(%rsp), %rax

27. Para traducir una asignación condicional (  $a = b ? c : d$  ; ) de lenguaje C a lenguaje ensamblador, gcc puede que utilice...

- a) Un salto incondicional, según la condición expresada en el código C, y otro salto incondicional
- b) Un salto condicional, según la condición opuesta a la del código C, y otro salto condicional
- \* c) Una instrucción de movimiento condicional, pero sólo si el procesador es Pentium Pro/I o superior
- d) Una instrucción de movimiento incondicional, pero sólo si el S.O. es de 64bits

28. Sobre el direccionamiento relativo a contador de programa:

- a) Favorece la implementación de código reubicable.
- b) Su uso en los saltos y llamadas a subrutinas reduce el tamaño de la instrucción.
- c) Es adecuado para alcanzar instrucciones próximas a la que se está ejecutando.
- \* d) Todas las respuestas son ciertas.

29. En arquitectura x86, el registro SP / ESP / RSP...

- a) es un registro transparente al usuario y contiene la dirección de memoria a la que se está accediendo
- b) es un registro transparente al usuario y contiene la instrucción que se está ejecutando
- c) es un registro de propósito específico y contiene la dirección de la siguiente instrucción a ejecutar
- \* d) es un registro de propósito específico y contiene la dirección de la cima de la pila

30. ¿Cuál de las siguientes instrucciones es errónea? (sale mensaje de error al intentar ensamblar):

- a) movw %dx, (%rax)
- \* b) movb \$0xFF, (%dl)
- c) movswl (%eax), %edx
- d) movzbl %dl, %eax

31. ¿Qué valor contendrá el registro RDX tras ejecutar las dos instrucciones siguientes?

```
movq $-1, %rdx
shr $16, %rdx
    a) 0xFFFF FFFF FFFF 0000
    b) 0xFFFF FFFF FFFF FFFF
*   c) 0x0000 0000 0000 FFFF
    d) 0xFFFF FFFF 0000 FFFF
```

32. Si rcx vale -1, tras ejecutar las instrucciones

```
rol $1, %cl
rcr $2, %rcx
```

el nuevo valor de RCX y del flag CF es

- a) hay algún fallo de sintaxis o gramática en esas instrucciones
- b) RCX≠-1, CF mantiene su valor
- \* c) RCX=-1, CF=1
- d) no se puede marcar ninguna de las opciones anteriores

33. En la nomenclatura del ensamblador de IA32, una cantidad de 16 bits es designada como:

- a) half word

- \*     b) word
- c) double word
- d) quad Word

34. Para traducir una construcción if-then-else de lenguaje C a lenguaje ensamblador, gcc utiliza generalmente

- a) dos saltos condicionales y dos saltos incondicionales
- b) un salto condicional, según la condición expresada en el código C
- c) dos saltos condicionales (uno para la parte if y otro para la parte else)
- \*     d) un salto condicional, según la condición opuesta a la del código C, y otro salto incondicional

35. ¿Cuál fue el primer procesador de Intel de 64 bits en la familia x86(-64)?

- \*     a) Pentium 4F
- b) Core i7
- c) 8086
- d) 386

36. ¿Cuál es la diferencia entre las instrucciones mov y lea?

- a) mov puede usarse para copiar un registro a otro, mientras que lea no
- b) lea accede a la posición indicada, mientras que mov no lo hace
- \*     c) mov accede a la posición indicada, mientras que lea no lo hace
- d) lea puede usarse para copiar un registro a otro, mientras que mov no

37. La extensión de signo a m bits de un número original N de n bits, con m > n, consiste en:

- \*     a) Incrementar la cantidad de bits a m preservando el signo y el valor del número.
- b) Realizar la operación  $2^m - N$
- c) Incrementar la cantidad de bits a m rellenando con unos por la izquierda.
- d) Realizar la operación  $2^m - N - 1$

38. En x86-64, el registro contador de programa se denomina:

- a) PCR
- b) PC
- \*     c) RIP
- d) EIP

39. ¿Cuál sería el “equivalente x86-64” del “pseudo-código C” `rcx=((long*)rax)[rbx]`?

- \*     a) `mov (%rax,%rbx,8), %rcx`
- b) `lea (%rax,%rbx,4), %rcx`
- c) `lea (%rax,%rbx,8), %rcx`
- d) `mov (%rax,%rbx,4), %rcx`

40. ¿Qué valor contendrá el registro rdx tras ejecutar las dos instrucciones siguientes?

```
movq $-1, %rdx
movl $1, %edx
a) 0xFFFF FFFF 0001
b) 0xFFFF FFFF 0000 0001
c) 0xFFFF 0000 0000 0001
*   d) 0x0000 0000 0000 0001
```

41. Cuál de las instrucciones máquina siguientes es incorrecta en x86-64:

- a) addq \$1, %rcx
- b) movl (%rdi,%rcx,4), %edx
- c) testl %edx, %edx
- \* d) movl %r8, %eax

42. ¿Cuál de los siguientes microprocesadores no es de 64 bits?

- a) Core i7
- \* b) Pentium III
- c) Itanium
- d) Core 2

43. ¿Cuál de las siguientes secuencias de instrucciones multiplica el (contenido del) registro RAX por 18?

- a) shlq \$18, %rax
  - \* b) leaq (%rax,%rax,8), %rax
  - leaq (%rax,%rax), %rax
  - c) imulq \$0x18, %rax
  - d) sarq \$1, %rax
- imulq \$9, %rax

44. Un overflow nunca puede ocurrir cuando:

- a) se suman dos números negativos
- b) se suman dos números positivos
- c) se resta un número positivo de un número negativo
- \* d) se suma un número positivo a un número negativo

45. ¿Cuál de los siguientes grupos de instrucciones IA32 sólo modifican los indicadores de estado sin almacenar el resultado de la operación?

- \* a) CMP, TEST
- b) IMUL, IDIV
- c) ADC, SBB
- d) AND, OR, XOR

46. Los arrays bidimensionales en lenguaje C se almacenan en orden...

- a) "de fila a columna" (file-to-column)
- \* b) "mayor-de-fila" (row-major)
- c) "de mayor a menor" (major-to-minor)
- d) "mayor-de-columna" (column-major)

47. En un sistema Linux x86-64, ¿cuál de las siguientes variables ocupa más bytes en memoria?

- a) char a[7]
- \* b) int \*c
- c) float d
- d) short b[3]

48. ¿Cuál de las siguientes afirmaciones es falsa?

- \* a) Las subrutinas necesitan recibir parámetros desde el programa que hace la llamada que indiquen qué registros pueden alterar y cuáles no
- b) Las subrutinas necesitan ser capaces de reservar espacio en memoria para las variables locales sin sobrescribir ningún dato usado por el programa que hace la llamada
- c) Los programas necesitan una forma de pasar parámetros a las subrutinas y de recibir las salidas de vuelta

d) Las subrutinas necesitan algún modo de saber desde dónde han sido llamadas para poder volver al programa que realizó la llamada cuando se completa la subrutina

49. En el fragmento de programa siguiente:

```
66b: e8 00 00 00 00 callq 670 <nxt>
670: 58      pop %rax
```

¿Qué valor termina almacenado en %rax?

- a) 0x66b
- \* b) 0x670
- c) 0xe8000000
- d) 0x58

50. En la convención cdecl, es responsabilidad del procedimiento llamado salvaguardar los registros:

- a) %eax, %ebx, %ecx, %edx
- b) %esi, %edi
- \* c) %ebx, %esi, %edi
- d) %eax, %edx, %ecx

51. La diferencia entre las instrucciones test y cmp consiste en que

- a) test realiza una operación and lógico, mientras que cmp realiza una resta
- b) test modifica sólo los flags lógicos (ZF,SF) mientras que cmp modifica los aritméticos (ZF,SF,CF,OF)
- \* c) ambas respuestas son correctas
- d) ambas respuestas son incorrectas

52. La instrucción cmovb %rdx, %rax

- a) copia en %rax el byte de memoria apuntado por la dirección contenida en %rdx
- b) copia el byte bajo de %rdx en el byte bajo de %rax
- \* c) copia en %rax el contenido de %rdx si el indicador de acarreo es 1
- d) copia en %rax el contenido de %rdx si %rax es menor que %rdx

53. Si EAX=2, EBX=5 y M[3]=3, ¿qué valores quedan tras ejecutarse la instrucción XOR %BL, 1(%EAX)?

- a) EAX=6 , EBX=5 , M[3]=2
- \* b) EAX=2 , EBX=5 , M[3]=6
- c) EAX=6 , EBX=2 , M[3]=5
- d) EAX=5 , EBX=6 , M[3]=2

54. ¿Cuál de las siguientes secuencias de instrucciones calcula a=b-a, suponiendo que %eax es a y %ebx es b?

- a) subl %eax, %ebx
- b) subl %ebx, %eax
- c) notl %eax
- addl %ebx, %eax
- \* d) Varias o ninguna de las respuestas anteriores son correctas, no se puede marcar una y sólo una

55. Considerar las siguientes declaraciones de estructuras en una máquina Linux de 64-bit.

```
struct RECORD {
```

```

int value2;
short ref_count;
char tag[10];
};

struct NODE {
    long value;
    struct RECORD record;
    char string[8];
};

```

También se declara una variable global "my\_node" como sigue:

```
struct NODE my_node;
```

Si la dirección de my node es 0x600940, ¿cuál es el valor de &my node.record.tag[1]?

- a) 0x60094a
- b) 0x60094e
- \* c) 0x60094f
- d) Ninguna de las anteriores

56. Una sentencia en C del tipo "while (test) body;" puede transformarse en código "goto" como:

- a) loop:  
if (test) goto done;  
body;  
goto loop;  
done:
- \* b) if (!test) goto done;  
loop:  
body;  
if (test) goto loop;  
done:
- c) if (test) goto true;  
goto done;  
true:  
body;  
done:
- d) loop:  
body;  
if (test) goto loop;

57. Un archivo .o que contiene código objeto:

- a) Puede ejecutarse directamente.
- \* b) Contiene instrucciones máquina binarias.
- c) Contiene las direcciones definitivas de las variables globales.
- d) Incluye el código de las funciones de biblioteca a las que llame.

58. Si A y B son dos enteros almacenados respectivamente en %eax y %ebx, ¿cuál de las siguientes implementaciones de if (!A && !B) {...then part...} es incorrecta?

- \* a). test %ebx, %eax  
jne not\_true  
...then part...

```

not_true:
...
b) . test %eax, %eax
jne not_true
test %ebx, %ebx
jne not_true
...then part...
not_true:
...
c) . or %ebx, %eax
jne not_true
...then part...
not_true:
...
d) . cmp $0, %eax
jne not_true
cmp $0, %ebx
jne not_true
...then part...
not_true:
...

```

59. En x86-64, es responsabilidad del procedimiento llamado (callee) salvaguardar, entre otros, los registros:

- a) %rbx, %rsi, %rdi
- b) %rax, %rdx, %rcx
- c) %rax, %rbx, %rcx, %rdx
- \* d) %rbx, %rbp

60. ¿Cuál de las siguientes afirmaciones NO es cierta? (entender que x86=IA32)

- a) Las disciplinas de pila para x86 y x86-64 son diferentes
- \* b) x86-64 usa %rbp como puntero base para el marco de pila
- c) x86-64 proporciona un espacio de memoria virtual mayor que x86
- d) x86 usa %ebp como puntero base para el marco de pila

61. Se definen la unión, variables, y función C, siguientes:

```

typedef union {
    float f;
    unsigned u;
} bit_float_t;

float f1;
unsigned u1=0x80000000;
float f2;

float bit2float(unsigned u) {
    bit_float_t arg;
    arg.u = u;
    return arg.f;
}

```

¿Cuál afirmación es verdadera?

- a) Si asignamos  $f1=\text{bit2float}(u1)$ ;  $f2=\text{float}(u1)$ ; entonces  $f1==1073741824.00$  y  $f2==1073741824.00$
- b) Si asignamos  $f1=\text{bit2float}(u1)$ ;  $f2=\text{float}(u1)$ ; entonces  $f1==f2$ , ambos distintos de cero
- \* c) Si asignamos  $f1=\text{bit2float}(u1)$ ;  $f2=(\text{float})u1$ ; entonces  $f1== -0.0$  pero  $f2==2147483648.00$
- d) Si asignamos  $f1=\text{bit2float}(u1)$ ;  $f2=(\text{float})u1$ ; entonces  $f1== 0.0$  pero  $f2==4294967296.00$

62. ¿Cuál de las siguientes instrucciones situada al principio de una función se utilizará probablemente para crear espacio en la pila para variables locales sin inicializar?

- a) sub \$0x30, %rbp
- b) add \$0x30, %rbp
- c) add \$0x30, %rsp
- \* d) sub \$0x30, %rsp

63. ¿Cuál de las siguientes instrucciones es incorrecta?

- a) shr \$1,%rdx
- b) shr %rdx
- c) shr %cl,%rdx
- \* d) shr %rcx,%rdx

64. ¿Cuál es la diferencia entre las instrucciones mov y lea?

- \* a) mov referencia (accede) la posición indicada, mientras que lea no lo hace
- b) lea referencia (accede) la posición indicada, mientras que mov no lo hace
- c) lea puede usarse para copiar un registro a otro, mientras que mov no
- d) mov puede usarse para copiar un registro a otro, mientras que lea no

65. Una función C declarada como `int get_var_digit(size_t index, size_t digit)` genera como código ensamblador

```
movq var(%rdi,8), %rax  
movl (%rax,%rsi,4), %eax  
ret
```

Se puede adivinar que:

- a) var es un array multi-nivel (punteros a enteros) de cuatro filas
- \* b) var es un array multi-nivel pero no se pueden adivinar las dimensiones
- c) var es un array bidimensional de enteros, con ocho columnas
- d) var es un array bidimensional de enteros, con cinco columnas

66. Se ha declarado en un programa C la variable `int val[5]={1,5,2,1,3};` ¿Cuál de las siguientes afirmaciones es cierta?

- a)  $\text{val}[1] == 1$
- \* b)  $\&\text{val}[3] == (\text{char} *)\text{val} + 12$
- c)  $\text{sizeof}(\text{val}) == 5$
- d)  $\&\text{val}[2]$  es de tipo `int *` y vale lo mismo que  $\text{val} + 8$

67. ¿Cuál de las siguientes instrucciones convierte  $\%rax = 5 * \%rax$ ?

- 1)  $\text{mov} (\%rax, \%rax, 4), \%rax$
- 2)  $\text{lea} (\%rax, \%rax, 4), \%rax$

- \* a) Ambas 1 y 2
- b) Sólo 2
- c) Sólo 1
- d) Ninguna

68. Sabiendo que las instrucciones de salto condicional codifican la dirección de salto con direccionamiento relativo a contador de programa (de 8 o 32 bits con signo), indicar cuál es la dirección de la instrucción pop en el siguiente desensamblado, donde se ha tachado la parte de las direcciones.

- ```
xxxxxx: 77 02  ja 400547
xxxxxx: 5d      pop %rbp
          a) 400543
          b) 400547
          c) 400549
*
```
- d) 400545

69. Respecto a requisitos de alineamiento de structs en gcc/IA32 x86 y x86-64, una de las siguientes afirmaciones es \*FALSA\*

- a) en x86 Linux alinea long double a 4x
- b) en x86 Linux alinea double a 4x
- c) en x86-64 Linux alinea double a 8x
- \*
- d) en x86-64 Linux alinea float a 8x

70. Las siguientes afirmaciones sugieren que el tamaño de varios tipos de datos en C (usando el compilador gcc) son iguales tanto en IA32 como en x86-64. Sólo una de ellas es FALSA.

¿Cuál?

- a) El tamaño de un unsigned es 4 bytes
- \*
- b) El tamaño de un long es 4 bytes
- c) El tamaño de un int es 4 bytes
- d) El tamaño de un short es 2 bytes

71. ¿Cómo se devuelve en ensamblador x86 Linux gcc el valor de retorno de una función al terminar ésta?

- a) Se almacena en pila justo encima del (%rbp) del invocado
- b) Se almacena en pila justo encima de los argumentos de la función
- \*
- c) Por convención se guarda en %rax
- d) La instrucción RET lo almacena en un registro especial de retorno

72. Si AX = 0xFA50 y ejecutamos XOR \$0xFF, AX

- a) El registro AL se pone a 0.
- b) El registro AH se pone a 0.
- c) Se realiza el complemento a 1 de AH.
- \*
- d) Se realiza el complemento a 1 de AL.

73. En un sistema x86-64, si %rsp tiene el valor 0x7fffff0000 inmediatamente antes de ejecutar una instrucción retq, ¿cuál es su valor inmediatamente después?

- \*
- a) 0x7fffff0008
- b) 0x7fffff0000
- c) 0x7fffff0004
- d) 0x7fffffefff8

74. El rasgo distintivo de la traducción “salta-en-medio” que gcc hace de un bucle while de lenguaje C a lenguaje ensamblador es...

- \*     a) el salto incondicional hacia adelante
- b) el salto incondicional hacia atrás
- c) el salto condicional hacia adelante
- d) el salto condicional hacia atrás

75. Habiendo declarado int array={-4,-3,-2,-1}; y char \*ptr=array; ¿cuánto vale ptr[1]?

- \*     a) -1
- b) -3
- c) -4
- d) -2

76. Si la estructura struct a ocupa un espacio de 28 bytes en memoria, ¿cuántos bytes ocupa la siguiente estructura struct b cuando se compila en 64 bits?

```
struct b {  
    struct a a1;  
    int i;  
    struct a a2;  
};
```

- a) 24 bytes
- b) 64 bytes
- \*     c) 60 bytes
- d) 84 bytes

77. ¿Qué valor contendrá el registro rdx tras ejecutar las dos instrucciones siguientes?

```
movq $-1, %rdx  
movb $1, %dl
```

- a) 0xFFFF FFFF FFFF FFF1
- \*     b) 0xFFFF FFFF FFFF FF01
- c) 0xFFFF FFFF FFFF 0001
- d) 0x0000 0000 0000 0001

78. En una resta de dos números en complemento a dos, se produce desbordamiento cuando...

- a) los dos operandos son negativos y el resultado es positivo.
- b) los dos operandos son positivos y el resultado es negativo.
- c) Las dos respuestas a y b son correctas.
- \*     d) Ninguna de las anteriores es correcta.

79. ¿Cuál de las siguientes afirmaciones es correcta?

- a) El lenguaje máquina es igual para todos los computadores.
- b) El lenguaje ensamblador es igual para todos los computadores.
- c) Las instrucciones en lenguaje ensamblador se almacenan y tratan como cadenas de unos y ceros.
- \*     d) Las instrucciones en lenguaje máquina se almacenan y tratan como cadenas de unos y ceros.

80. Se definen las variables, unión y función C siguientes:

```

float f1;
unsigned u1=0x80000000;
float f2;

typedef union {
    float f;
    unsigned u;
} bit_float_t;

float bit2float(unsigned u) {
    bit_float_t arg;
    arg.u = u;
    return arg.f;
}

```

¿Cuál afirmación es verdadera?

- a) Si asignamos  $f2 = \text{float}(u1)$ ; entonces  $f2 == 0.0$
- b) Si asignamos  $f1 = \text{bit2float}(u1)$ ; entonces  $f1 == 2147483648.00$
- c) Si asignamos  $f2 = (\text{float})u1$ ; entonces  $f2 == 4294967296.00$
- \* d) Si asignamos  $f1 = \text{bit2float}(u1)$ ; entonces  $f1 == -0.0$

81. La instrucción `movlq %eax, %rax`

- \* a) no existe, se debe usar `mov %eax, %eax`
- b) copia en `%rax` el valor de `%eax` si el indicador de cero está activado
- c) copia en `%rax` el valor sin signo almacenado en `%eax`, llenando con ceros
- d) pone a 0 el registro `%rax`

82. La instrucción necesaria para cargar 0x07 en `%eax` es:

- \* a) `movl $0x07,%eax`
- b) `movb $0x07,%al`
- c) `movb $0x07,%eax`
- d) `movl $0x07,%ah`

83. La instrucción `seta %al` (`seta` significa "set if above"):

- a) pone AL a 1 si ZF=1 y CF=1
- b) pone AL a 1 si ZF=0 o CF=1
- \* c) pone AL a 1 si ZF=0 y CF=0
- d) pone AL a 1 si ZF=1 o CF=0

84. La instrucción `movzbl %al, %eax`

- a) Copia en `%eax` el valor de `%al` si el indicador de cero está activado
- b) Copia en `%eax` el valor del indicador de cero
- c) Pone a 0 el registro `%eax`
- \* d) Copia en `%eax` el valor sin signo almacenado en `%al`, llenando con ceros

85. La instrucción `xor $3, %eax` tiene como resultado:

- a) Poner a 0 los últimos 3 bits del registro EAX
- \* b) Cambiar  $0 <-> 1$  (complemento a 1 de) los últimos 2 bits del registro EAX
- c) Poner a 1 el último bit del registro EAX
- d) Ninguno de los anteriores resultados

86. ¿Qué salida produce el siguiente código? Asumir representación de datos de arquitectura x86-64.

```
unsigned int x = 0xDEADBEEF;
unsigned short y = 0xFFFF;
signed int z = -1;
if (x > (signed short) y)
    printf("Hello");
if (x > z)
    printf("World");
```

(Recordar que:

- 1.- las extensiones de tamaño se hacen según tenga o no signo el tipo fuente
- 2.- en comparaciones con un dato unsigned se pasa el otro dato a unsigned

- )
- a) Imprime "World"
  - b) Imprime "Hello"
  - \* c) No imprime nada
  - d) Imprime "HelloWorld"

87. Se ha declarado en un programa C la variable int val[5]={1,5,2,1,3} . ¿Cuál de las tres primeras opciones (a, b, o c) es FALSA?

- \* a) val[1] == 1
- b) &val[3] == val+3
- c) sizeof(val) == 20
- d) No se puede marcar ninguna de ellas, todas (a, b y c) son ciertas

88. ¿Cuál de las siguientes afirmaciones sobre compilación C->ASM es falsa?

- \* a) Puede que el compilador altere por optimización el orden de los parámetros pasados en una llamada a función, si el marco de pila resultante es más eficiente
- b) Puede que el compilador altere el orden del código C, apareciendo antes la traducción de sentencias C posteriores
- c) Puede suceder que varias sentencias C se traduzcan por una única instrucción ASM
- d) Puede que el compilador elimine por optimización construcciones C enteras (como un bucle for completo), si se conoce el resultado en tiempo de compilación

89. Usando el repertorio x86-64, para intercambiar el valor de 2 registros se pueden usar...

- a) una instrucción mov y una instrucción lea
- \* b) 3 mov, no menos (se le llama "intercambio circular")
- c) 4 mov, no menos (debido a la arquitectura R/M)
- d) dos instrucciones mov

90. ¿Cuál de las siguientes instrucciones es errónea? (sale mensaje de error al intentar ensamblar):

- a) movb %sil, (%rax)
- b) pushq \$0xFF
- c) movsbw (%rax), %dx
- \* d) movzlq %edx, %rax

91. Al ejecutar el fragmento de código:

```
# leal válida, dir.64b truncada a 32b
```

```
leal -1(%rax), %edx
cmpl $9, %edx
ja .L2
```

se salta a .L2 si el contenido del registro %eax:

- a) es menor o igual que 1
- \* b) está fuera del intervalo [1,10]
- c) está dentro del intervalo [1,10]
- d) es mayor o igual que 10

92. Si el registro %eax contiene el siguiente valor binario:

11111111 10101010 01010101 11110000

¿Cuál será el valor de %eax tras ejecutar la instrucción xorb %al, %al?

- a) 00000000 00000000 00000000 00000000
- b) 11111111 10101010 01010101 11110000
- c) 00000000 10101010 01010101 11110000
- \* d) 11111111 10101010 01010101 00000000

93. La instrucción x86-64 test sirve para...

- a) Testear el código de condición indicado, y poner un byte a 1 si se cumple
- b) Realizar la operación resta (a-b) pero no guardar el resultado, sino simplemente ajustar los flags
- c) Mover el operando fuente al destino, pero sólo si se cumple la condición indicada
- \* d) Realizar la operación and lógico bit-a-bit (a&b) pero no guardar el resultado, sino simplemente ajustar los flags

94. Si el registro rax contiene un long (64 bits CON signo) x, la secuencia de instrucciones siguiente:

```
cmpq $10, %rax
ja dest
```

saltará a la etiqueta dest si:

- a) x>9
- b) x<10
- \* c) x<=-1 || x>=11
- d) x>=0 && x<=10

95. Una función C llamada get\_el(...) genera el siguiente código ensamblador.

```
leaq (%rdi,%rdi,4), %rax
addq %rax, %rsi
movl arr(%rsi,4), %eax
ret
```

Se puede adivinar que:

- a) arr es un array multi-nivel (punteros a enteros) de cuatro filas
- \* b) arr es un array bidimensional de enteros, con cinco columnas
- c) arr es un array bidimensional de enteros, no se pueden adivinar dimensiones
- d) arr es un array multi-nivel pero no se pueden adivinar las dimensiones

96. En un microprocesador de 4 bits, una operación en la que el bit 0 de un registro se copia en el acarreo, después el bit 1 se copia en el bit 0, después el bit 2 se copia en el bit 1, y por último el bit 3 se copia en el bit 2, es:

- \* a) Una rotación a la derecha a través de acarreo.
- b) Un desplazamiento aritmético a la derecha.
- c) Una rotación a la derecha.
- d) Un desplazamiento lógico a la derecha.

97. La operación aritmética calculada por el programa

```
mov $5, %eax  
mov $3, %ebx  
mov $7, %ecx  
mov $8, %edx  
imul %ebx, %ecx  
add %ecx, %eax  
sub %edx, %eax
```

es:

- a)  $(3 \times 7) + 8 - 5$
- b)  $8 - 5 + (3 \times 7)$
- c)  $8 - (3 \times 7) + 5$
- \* d)  $5 + (3 \times 7) - 8$

98. Habiendo declarado int array={0,1,2,3}; y long long \*ptr=array; ¿cuánto vale ptr[1]?

- a) 0x0000 0001 0000 0000
- b) 0x0003 0002 0001 0000
- \* c) 0x0000 0003 0000 0002
- d) 0x0000 0001 0002 0003

99. Si rcx vale -1, tras ejecutar la instrucción sal \$3,%ecx el nuevo valor de RCX es

- a) 0xffff ffff ffff ffff
- b) 0xffff ffff ffff fff8
- c) 0x1fff ffff ffff ffff
- \* d) 0xffff fff8

100. Considerar las siguientes declaraciones de estructuras en una máquina Linux de 64-bit.

```
struct RECORD {  
    int value2;  
    short ref_count;  
    char tag[10];  
};  
  
struct NODE {  
    long value;  
    struct RECORD record;  
    char string[8];  
};
```

También se declara una variable global "my\_node" como sigue:

```
struct NODE my_node;
```

¿Cuál es el tamaño de my\_node en bytes?

- \* a) 28
- b) 32
- c) 40
- d) Ninguno de los anteriores

101. ¿Cuál de las siguientes instrucciones es \*errónea\*?

- a) lea (%rdx,%rsi,2), %rdi
- b) mov (%rax,%rbx,1), %rcx
- c) mov (%r9,%r10,8), %r11
- \* d) lea (%rbp,%rsp,4), %r8

102. Sabiendo que la instrucción de llamada mostrada abajo codifica la dirección de la subrutina con direccionamiento relativo a contador de programa (de 32 bits con signo), indicar el valor de los bytes tachados en el siguiente desensamblado.

400544: e8 XX XX XX XX callq 400550 <f>

400549: 48 89 03    mov %rax,(%rbx)

- a) 00 40 05 50
- b) 00 00 00 01
- c) 50 05 40 00
- \* d) 07 00 00 00

103. En la secuencia de programa siguiente:

628: e8 cd ff ff ff callq 5fa <suma>

62d: 48 83 c4 20    add \$0x20,%rsp

¿cuál es el valor que introduce en la pila la instrucción callq?

- a) 0xffffffffcd
- b) 0x5fa
- c) 0x628
- \* d) 0x62d

104. ¿Cuál es la diferencia entre las instrucciones mov y lea?

- a) lea puede usarse para copiar un registro a otro, mientras que mov no
- b) lea referencia (accede) la posición indicada, mientras que mov no lo hace
- \* c) mov referencia (accede) la posición indicada, mientras que lea no lo hace
- d) mov puede usarse para copiar un registro a otro, mientras que lea no

105. ¿Cuál de las siguientes afirmaciones es falsa?

- \* a) Las subrutinas necesitan recibir parámetros desde el programa que hace la llamada que indiquen qué registros pueden alterar y cuáles no
  - b) Las subrutinas necesitan algún modo de saber desde dónde han sido llamadas para poder volver al programa que realizó la llamada cuando se completa la subrutina
  - c) Los programas necesitan una forma de pasar parámetros a las subrutinas y de recibir las salidas de vuelta
  - d) Las subrutinas necesitan ser capaces de reservar espacio en memoria para las variables locales sin sobrescribir ningún dato usado por el programa que hace la llamada

106. ¿Cuál de los siguientes fragmentos de código deja en %eax un resultado distinto a los otros tres fragmentos?

- \* a) mov \$-1, %edx

```

sub %eax, %edx
mov %edx, %eax
    b) xor %edx, %edx
sub %eax, %edx
mov %edx, %eax
    c) neg %eax
    d) not %eax
add $1, %eax

```

107. Usando el repertorio x86-64, para intercambiar el valor de 2 variables (por ejemplo A: .int 1 y B: .int 2) se pueden usar...

- a) dos instrucciones mov
- b) una instrucción mov y una instrucción lea
- \* c) 4 mov, no menos (debido a la arquitectura R/M)
- d) 3 mov, no menos (se le llama "intercambio circular")

108. ¿Cuál de las siguientes instrucciones es errónea? (sale mensaje de error al intentar ensamblar):

- a) movb %sil, (%rax)
- b) pushq \$0xFF
- c) movsbw (%rax), %dx
- \* d) movzlq %edx, %rax

109. Para poner a 1 el bit 5 del registro %edx sin cambiar el resto de bits podemos usar la instrucción máquina:

- a) and \$0x5, %edx
- b) or \$0b101, %edx
- c) and \$32, %edx
- \* d) or \$0x20, %edx

110. El sufijo q de la instrucción movq significa:

- a) Que la instrucción realiza un movimiento rápido, trabajando con la mitad menos significativa de los operandos (move quick).
- b) Que la instrucción realiza un movimiento rápido, trabajando con la mitad más significativa de los operandos (move quick).
- \* c) Que la instrucción trabaja con operandos de 64 bits (quad word).
- d) Que la instrucción trabaja con operandos de 128 bits (quad word).

111. Si el contenido de RBP es 0x13000, ¿a qué dirección se hace referencia con la instrucción INC -0x80(%RBP)?

- a) 0x80
- b) 0x13000
- \* c) 0x12F80
- d) 0x13080

112. Si el contenido del registro %rax es 0x10 antes de ejecutar la instrucción shl \$0xc,%rax, ¿cuánto es su contenido tras ejecutarla?

- \* a) 0x10000
- b) 0x1000
- c) 0x4000
- d) 0x800

113. Dado el código C siguiente:

```
struct data {  
    char str[16];  
};  
  
char *f(struct data *ptr) {  
    return &(ptr->str[2]);  
}
```

la función se traducirá a ensamblador de x86-64 como:

- a) movq 2(%rdi), %rax
- ret
- b) movq (%rdi,2), %rax
- ret
- \* c) leaq 2(%rdi), %rax
- ret
- d) leaq (%rdi,2), %rax
- ret

114. Considerar la declaración C

```
long array[10] = {0, 1, 2, 3, 4, 5, 6, 7, 8, 9};
```

Suponer que el compilador tiene la dirección de array en el registro %rcx. ¿Cómo se movería el valor array[3] al registro %rax? Asumir que %rbx es 3.

- \*
- a) movq (%rcx,%rbx,8),%rax
- b) movq 8(%rcx,%rbx,2),%rax
- c) leaq (%rcx,%rbx,8),%rax
- d) leaq 12(%rcx),%rax

115. La arquitectura x86-64 tiene:

- a) 64 registros RPG de 64 bits
- b) 8 registros de propósito general (RPG) de 64 bits (%rax, %rbx, ... %rsp, %rbp)
- \*
- c) 16 registros RPG de 64 bits
- d) 32 registros RPG de 64 bits

116. Siendo RDX=0xf000 y RCX=0x0100, ¿cuál de las siguientes instrucciones tiene como dirección efectiva 0xf400?

- a) movq 0x8(%rdx), %rax
- b) addq (%rdx, %rcx), %rax
- c) leal 0x80(%rdx, 2), %eax
- \*
- d) xorl (%rdx, %rcx, 4), %eax

117. En el fragmento de código

```
400544: e8 07 00 00 00 callq 400550 <mult2>  
400549: 48 89 03      mov %rax,(%rbx)
```

la instrucción call suma al contador de programa la cantidad:

- a) 0x48
- b) 0x400544

- \*       c)     0x00000007
- d)     0x00400549

118. Diferencias gcc Linux IA32/x86-64: marcar la respuesta falsa

- a)     long double pasa de 10/12 B a 16 B
- \*       b)     el tipo double pasa de 4 B a 8 B
- c)     los enteros largos (long) pasan de 32 a 64 bits
- d)     los punteros (void\*) pasan de 32 a 64 bits

119. En el fragmento de programa siguiente:

```
66b: e8 8a ff ff ff callq 5fa <f>
670: 48 83 c4 10 add $0x10,%rsp
```

¿Cuál es el valor que introduce en la pila la instrucción callq?

- a)     0x5fa
- b)     0x66b
- c)     0xffffffff8a
- \*       d)     0x670

120. ¿Cuál es el popcount (peso Hamming, nº de bits activados) del número 19?

- a)     2
- \*       b)     3
- c)     4
- d)     5

121. La secuencia de instrucciones

```
leaq (%rdi, %rdi, 2), %rax
salq $3, %rax
```

produce el efecto de...

- a)     leer en RAX a partir de la posición  $2*RDI+2$  y multiplicar dicho contenido por 8
- b)     calcular RAX =  $(2*RDI+2) * 3 = 6*(RDI+1)$
- \*       c)     calcular RAX =  $24 * RDI$
- d)     calcular RAX =  $9 * RDI$

122. En x86-64, una función con 10 parámetros de tipo long que devuelve el valor del 8º parámetro y no modifica el puntero de pila puede traducirse a ensamblador como:

```
ret
*       b)     movq 16(%rsp), %rax
ret
c)     movq %r10, %rax
ret
d)     movq %r8, %rax
ret
```

123. ¿Cuál de las siguientes listas menciona registros x86-64 del mismo tipo respecto a convenio de uso? (salva-invocante, invocado, etc)

- a)     RAX, RBX, RCX, RDX
- b)     RBX, RSI, RDI

- \*        c) CL, DX, R8d, R9
- d) RSP, RBP

124. Si el registro rax contiene x, la sentencia en C

x &= 0x1;

se traducirá a ensamblador como:

- \*        a) andq \$1, %rax
- b) orq \$0x1, %rax
- c) shrq %rax
- d) sarq %rax

125. ¿Cuál de los siguientes no es un modo de direccionamiento x86-64?

- a) Inmediato
- b) Memoria
- \*        c) Cache
- d) Registro

126. ¿Qué valor contendrá el registro edx tras ejecutar las dos instrucciones siguientes?

movl \$-1, %edx

movb \$1, %dl

- a) 00000001 00000000 00000000 00000000
- b) 00000000 00000000 00000000 00000001
- \*        c) 11111111 11111111 11111111 00000001
- d) 11111111 11111111 11111111 11111111

127. Las siguientes afirmaciones sugieren que el tamaño de varios tipos de datos en C (usando el compilador gcc) son iguales tanto en IA32 como en x86-64. Sólo una de ellas es FALSA.

¿Cuál?

- \*        a) El tamaño de un short es 2 bytes
- b) El tamaño de un puntero es 4 bytes
- c) El tamaño de un double es 8 bytes
- d) El tamaño de un int es 4 bytes

128. ¿Cuál de las siguientes afirmaciones es falsa?

- a) Las subrutinas necesitan ser capaces de reservar espacio en memoria para las variables locales sin sobrescribir ningún dato usado por el programa que hace la llamada
- \*        b) Las subrutinas necesitan recibir parámetros desde el programa que hace la llamada que indiquen qué registros pueden alterar y cuáles no
- c) Los programas necesitan una forma de pasar parámetros a las subrutinas y de recibir las salidas de vuelta
- d) Las subrutinas necesitan algún modo de saber desde dónde han sido llamadas para poder volver al programa que realizó la llamada cuando se completa la subrutina

129. En la convención de llamada SystemV AMD64 seguida por gcc Linux/x86-64...

- a) RBX es un registro salva-invocado, por eso si es necesario conservar su valor hay que salvarlo antes de llamar a función
- b) RBP es un registro salva-invocante, por eso si es necesario conservar su valor hay que salvarlo antes de llamar a función
- \*        c) R12 es un registro salva-invocado, por eso en cualquier función hay que salvarlo antes de modificarlo

- d) R11 es un registro salva-invocante, por eso en cualquier función hay que salvarlo antes de modificarlo

130. ¿Cómo se devuelve en ensamblador x86-64 Linux gcc el valor de retorno de una función long int al terminar ésta?

- a) La instrucción RET lo almacena en un registro especial de retorno.
- b) Por convención se guarda en %rax.
- c) Se almacena en pila justo encima de los argumentos de la función.
- \* d) Ninguna de esas formas es la correcta.

131. En una suma de dos números en complemento a dos, se produce desbordamiento cuando

- a) Sumamos un número positivo y uno negativo.
- b) Sumamos dos negativos y el resultado es negativo.
- \* c) Sumamos dos positivos y el resultado es negativo o bien sumamos dos negativos y el resultado es positivo.
- d) Sumamos dos positivos y el resultado es positivo.

132. El procesador utiliza el puntero de pila...

- \* a) En las instrucciones de llamadas y retornos de subrutinas
- b) En todas las instrucciones que tengan al menos dos accesos a memoria
- c) En todo tipo de instrucciones de saltos, incluyendo llamadas y retornos a subrutinas
- d) En todas las instrucciones

133. En x86-64 el puntero de pila es:

- a) una dirección de memoria de 64 bits almacenada en el contador de programa
- \* b) un registro de 64 bits en el microprocesador
- c) un registro de 32 bits en el microprocesador
- d) un conjunto de posiciones de memoria usadas para almacenar información temporal durante la ejecución del programa

134. Si desplazamos rbx mediante salq \$4, %rbx:

- a) Lo multiplicamos por 4
- b) Lo dividimos por 4
- \* c) Lo multiplicamos por 16
- d) Lo dividimos por 16

135. En un sistema Linux x86-64, ¿cuál de las siguientes expresiones es equivalente a la expresión C  $(x[2] + 4)[3]$ ? Suponer que previamente se ha declarado int \*\*x.

- \* a)  $*((x + 2) + 4) + 3$
- b)  $*((x + 16)) + 28$
- c)  $((*(x + 2) + 4) + 3)$
- d)  $*(((*x) + 2) + 7)$

136. ¿Cuál es el efecto de la instrucción siguiente?

mov 8(%rbp),%rcx

- a) Suma 8 al contenido de la posición de memoria cuya dirección está almacenada en rbp, y almacena la suma en rcx
- \* b) Suma 8 al contenido de rbp, trata la suma como una dirección de memoria y almacena el contenido de esa dirección en rcx
- c) Suma 8 al contenido de rbp y almacena la suma en rcx

- d) Suma los contenidos de rbp y de la dirección de memoria 8 y almacena la suma en rcx

137. El registro RAX contiene el número binario 111111101111000010. ¿Cuál será su contenido tras ejecutar la instrucción sar \$1,%ax?

- \* a) 0x7fde1
- b) 0xffffde1
- c) 0x3fde1
- d) 0xfffffffffffffde1

138. La instrucción jbe / jna provoca un salto si...

- a) SF == 1 || ZF == 1
- \* b) CF == 1 || ZF == 1
- c) CF == 1
- d) SF != OF

139. ¿Cuál de las siguientes expresiones toma el valor 0x01 si x es múltiplo de 32 y 0x0 en caso contrario? Asumir que x es unsigned int.

- a) (x | 0x3f)
- \* b) !(x & 0x1f)
- c) (x & 0x1f)
- d) !(x & 0x3f)

140. La instrucción not:

- a) realiza un salto condicional si negativo
- b) realiza la operación no-or (or negada)
- \* c) realiza el complemento a uno (cambiar unos por ceros y ceros por unos)
- d) realiza el complemento a dos

141. ¿Cuál de las siguientes instrucciones máquina copia en el registro RDX la dirección efectiva resultante de la operación RAX\*8 + RBX?

- a) movq (%rbx, %rax, 8), %rdx
- b) movq 8(%rdx, %rax), %rdx
- \* c) leaq (%rbx, %rax, 8), %rdx
- d) leaq 8(%rdx, %rax), %rdx

142. Si ECX vale 0, la instrucción adc \$-1,%ecx

- a) Cambia CF
- b) Pone CF=1
- c) Pone CF=0
- \* d) No cambia CF

143. ¿Qué combinación de flags aritmético-lógicos corresponde al código de condición b (below)?

- a) OF xor SF
- \* b) CF
- c) OF
- d) CF xor OF

144. ¿Cuál de las siguientes instrucciones \*NO\* es errónea?

- \* a) movb \$0xF, (%rax)
- b) movw (%rbx), 4(%rsp)

- c) movq %rdx, \$0x123
- d) movl %rcx, (%rsp)

145. Si la variable val está almacenada en ebx y la variable x está almacenada en eax, la sentencia val ^= x; se puede traducir a ensamblador como:

- a) andl %ebx,%eax
- b) testl %eax,%ebx
- \* c) xorl %eax,%ebx
- d) xorl %ebx,%eax

146. ¿Cuál de los siguientes microprocesadores no es de 64 bits?

- a) Itanium
- b) Core 2
- \* c) Pentium III
- d) Core i7

147. Si la dirección del primer elemento de un vector de enteros z está almacenada en el registro %rdi y el índice i está almacenado en el registro %rsi, la instrucción máquina que realiza la operación z[i]++ es:

- a) addl \$1, (%rsi,%rdi,4)
- \* b) addl \$1, (%rdi,%rsi,4)
- c) addl \$4, (%rsi,%rdi)
- d) addl \$1, (%rdi,%rsi)

148. Para traducir una construcción do-while de lenguaje C a lenguaje ensamblador, gcc utiliza generalmente

- \* a) un salto condicional hacia atrás, según la misma condición que en lenguaje C
- b) un salto condicional hacia adelante, según la misma condición que en lenguaje C
- C c) un salto condicional hacia atrás, según la condición opuesta a la de lenguaje C
- d) un salto condicional hacia adelante, según la condición opuesta a la de lenguaje C

149. La instrucción JNGE / JL provoca un salto si...

- a) SF = 1
- b) CF = 1
- \* c) SF ≠ OF
- d) SF = CF

150. Estudiando el listado de una función C presuntamente compilada con gcc en modo 64bit (x86-64), nos dicen que la instrucción "movl (%rdi), %eax" carga en el registro EAX el valor adonde apunta el primer argumento.

- a) Está mal, porque EAX no se puede usar en modo 64bit, debería ser RAX
- b) Está mal, porque EAX no se carga con ningún valor
- \* c) Está bien, y pone a cero los 32 bits más significativos de RAX
- d) Está mal, porque el primer argumento de una función C no se pasa en RDI

151. Considerar las siguientes declaraciones de estructuras en una máquina Linux de 64-bit.

```
struct RECORD {  
    long value2;  
    int ref_count;  
    char tag[4];
```

```

};

struct NODE {
    double value;
    struct RECORD record;
    char string[8];
};

```

También se declara una variable global "my\_node" como sigue:

```
struct NODE my_node;
```

Si la dirección de my\_node es 0x601040, ¿cuál es el valor de &my\_node.record.tag[1]?

- a) 0x601050
- b) 0x601054
- \* c) 0x601055
- d) Ninguna de las anteriores

152. El lenguaje máquina es...

- \* a) difícil de codificar manualmente.
- b) portable entre arquitecturas.
- c) fácilmente legible por el programador.
- d) una alternativa razonable al uso del lenguaje ensamblador.

153. ¿Cuál de los siguientes registros x86-64 es distinto del resto en convenio de uso? (salvo invocante/invocado)

- \* a) RBX
- b) RCX
- c) RSI
- d) R8

154. Si %rdx contiene 0xf000 y %rcx contiene 0x0100, el direccionamiento 0x80(%rcx,%rdx,2) se refiere a la posición

- a) 0xf182
- b) 0xf280
- \* c) 0x1e180
- d) Ninguna de las respuestas anteriores es correcta

155. En los casos concretos indicados para las siguientes instrucciones x86-64, ¿cuál no funciona como instrucción de transferencia?

- a) pushq %rax
- \* b) cmpq 0x6000f0, %rax
- c) leaq variable, %rax
- d) movq \$0x15, %rax

156. Sabiendo que las instrucciones de salto condicional codifican la dirección de salto con direccionamiento relativo a contador de programa (de 8 o 32 bits con signo), indicar cuál es la dirección de salto de la instrucción je en el siguiente desensamblado, donde se ha tachado precisamente dicha dirección.

```

40042f: 74 f4  je xxxxxx
400431: 5d          pop %rbp
*
```

- b) 400525
- c) 400431
- d) 40043d

157. La primera letra (l) de la instrucción lea:

- \* a) forma parte del nemotécnico de la instrucción
- b) indica que la instrucción trabaja con un operando destino de 32 bits (long word)
- c) indica que la instrucción afecta a los 16 bits menos significativos del operando destino (low word)
- d) indica que la instrucción usa ordenación de bytes little-endian

158. En x86-64, ¿cuál de los siguientes fragmentos de programa tiene un efecto sobre los flags distintos al resto?

- \* a) xor %edi, %edi  
add \$-1, %edi
- b) mov \$-1, %edi
- c) sub %edi, %edi  
adc \$0xFFFFFFFF, %edi
- d) mov \$-1, %edi  
add \$0, %edi

159. Un procesador cuya instrucción CALL guardara la dirección de retorno en un registro RL (llamado "de enlace"):

- a) Permitiría llamadas anidadas, pero no recursivas.
- b) Permitiría llamadas anidadas y recursivas.
- \* c) No permitiría llamadas anidadas ni recursivas.
- d) Permitiría llamadas recursivas, pero no anidadas.

160. En un sistema IA32 Linux, ¿cuál es el tamaño de un long?

- a) 6 bytes
- b) 2 bytes
- \* c) 4 bytes
- d) 8 bytes

161. Si la estructura struct a ocupa un espacio de 26 bytes en memoria, ¿cuántos bytes ocupa la siguiente estructura struct b cuando se compila en 64 bits?

```
struct b {  
    struct a a1;  
    int i;  
    struct a a2;  
};
```

- a) 24
- b) 58
- c) 60
- \* d) 64

162. Respecto a la convención de llamada SysV AMD64 usada en Linux/gcc:

- a) Todos los registros pueden ser modificados libremente por todas las subrutinas

- b) Una subrutina que modifique algún registro debe restaurar su valor anterior antes de retornar
- \* c) Hay registros que pueden ser modificados libremente por las subrutinas, y otros que, si se modifican, se deben restaurar posteriormente. Y también hay registros especiales
- d) Hay registros modificables, otros que deben ser restaurados, y las subrutinas anidadas deben respetar los registros modificables que están en uso por otras subrutinas

163. ¿Cuál de las siguientes instrucciones máquina copia en RAX la dirección efectiva resultante de la operación RDX\*8 + RBX?

- \* a) leaq (%rbx, %rdx, 8), %rax  
 b) movq (%rbx, %rdx, 8), %rax  
 c) leaq 8(%rdx, %rdx), %rax  
 d) movq 8(%rdx, %rdx), %rax

164. Una función C declarada como int \*get\_ap(struct rec \*r, size\_t idx) genera como código ensamblador leaq (%rdi,%rsi,4); ret. Se puede adivinar que:

- a) la estructura contiene un array de 4 elementos  
 \* b) la estructura comienza con un array de enteros (o tipo similar)  
 c) la estructura contiene 5 campos  
 d) r es un array de 4 estructuras

165. ¿Cuál es el resultado de evaluar la expresión 0b1110 ^ 0b1010 en lenguaje C?

(0b indica que el número está expresado en binario)

- \* a) 0b0100  
 b) 0b0110  
 c) 0b1111  
 d) 0b1010

166. La instrucción test es...

- \* a) Lo mismo que and, pero no guarda el resultado, sólo ajusta los flags  
 b) Lo mismo que sub, pero no guarda el resultado, sólo ajusta los flags  
 c) Lo mismo que sub  
 d) Lo mismo que and

167. Suponiendo que todos los registros inicialmente contienen el valor 1, ¿cuál es el valor de RCX tras la ejecución de la siguiente secuencia de instrucciones?

```
mov $4, %rax
mov $3, %rbx
lea (%rax, %rax), %rcx
sub %rbx, %rcx
imul %rcx, %rax
```

\* a) 20  
 b) 6  
 c) 5  
 d) 36

168. Dentro de una función declarada como void swap(long \*xp, long \*yp), que intercambia los valores de los dos enteros (de tamaño long) cuyas direcciones de memoria (punteros) son pasadas como parámetros a la función, la instrucción movq (%rsi),%rdx copia en %rdx...

- a) el valor del entero apuntado por el puntero pasado como primer parámetro

- \*     b) el valor del puntero pasado como segundo parámetro
- c) el valor del puntero pasado como primer parámetro
- \*     d) el valor del entero apuntado por el puntero pasado como segundo parámetro

169. Respecto a registros de propósito general (RPG), el 80386 tiene:

- a) 16 registros de 64 bits
- b) 16 registros de 16 bits
- \*     c) 8 registros de 32 bits
- d) 32 registros de 8 bits

170. Respecto a direccionamiento a memoria en ensamblador x86-64 (sintaxis AT&T), de la forma D(Rb, Ri, S), sólo una de las siguientes afirmaciones es FALSA. ¿Cuál?

- \*     a) Los registros base e índice (Rb y Ri) pueden ser cualesquiera de los 16 registros enteros (RAX...RSP, R8...R15)
- b) El desplazamiento D puede ser una constante literal (1, 2 ó 4 bytes)
- c) El factor de escala S puede ser 1, 2, 4, 8
- d) El desplazamiento D también puede ser el nombre de una variable (que se traduce por su dirección, de 4bytes)

171. Dada una función que devuelve la suma de 8 enteros en x86-64, ¿cuál de las siguientes instrucciones suma el 7º argumento?

- \*     a) add 0x8(%rsp), %eax
- b) add -0x4(%rsp), %eax
- c) add 0x4(%rsp), %eax
- d) add -0x8(%rsp), %eax

# PRACTICAS

## Sesión 1

**1. En x86-64, el registro contador de programa se denomina:**

- a) PC
- b) RIP
- c) EIP
- d) PCR

**2. El lenguaje máquina es...**

a) fácilmente legible por el programador.

no, son contenidos de memoria, bits en la memoria del ordenador (o imagen en disco)

- b) difícil de codificar manualmente.

sí, habría que codificarlo todo (instrucciones: consultar las tablas de codops; datos: binario, complemento a 2, ASCII, unicode...; direcciones: asignar espacio para variables manualmente...)

c) una alternativa razonable al uso del lenguaje ensamblador.

no, usar un ensamblador ahorra tener que realizar manualmente todas las codificaciones mencionadas (instrucciones, datos, direcciones...)

d) portable entre arquitecturas.

no, cada arquitectura tiene su propio repertorio, si coincide el repertorio es la misma arquitectura (ISA)

**3. Sobre el programa ensamblador:**

- a) La calidad de un programa ensamblador afectará menos al tiempo de ejecución de los programas generados por él que la calidad de un compilador.
- b) Las etiquetas permiten que el programador especifique el destino de un salto de forma que éste no tenga que modificarse manualmente cuando el programa varíe de tamaño.
- c) El lenguaje ensamblador elimina la posibilidad de errores en la generación de la representación en lenguaje máquina de cada instrucción.
- d) Todas las respuestas son ciertas.

**4. ¿Cuál de las siguientes afirmaciones es correcta**

- a) El lenguaje ensamblador es igual para todos los computadores.
- b) El lenguaje máquina es igual para todos los computadores.
- c) Las instrucciones en lenguaje ensamblador se almacenan y tratan como cadenas de unos y ceros.

- d) Las instrucciones en lenguaje máquina se almacenan y tratan como cadenas de unos y ceros.

**5. ¿Cuál de las siguientes afirmaciones es incorrecta?**

- a) En lenguaje ensamblador las instrucciones se escriben en binario.  
no, "representación textual" dice la tr.12
- b) En lenguaje ensamblador cada instrucción se corresponde con una instrucción máquina.  
sí, "representación textual" dice la tr.12
- c) En lenguajes de alto nivel no se tiene acceso a detalles del procesador como las tablas de páginas, el paso de modo usuario a modo supervisor, el bit de paridad, etc.  
exacto, todo eso es de bajo nivel, ni siquiera C contempla esos detalles, y eso que C no es de muy alto nivel
- d) El lenguaje de alto nivel es más portable que el lenguaje máquina.  
se puede llevar a cualquier máquina con compilador de ese lenguaje

**6. En un programa en ensamblador queremos crear espacio para una variable entera var inicializada a 1. La línea que hemos de escribir en la sección de datos es:**

- a) .int var 1
- b) int var 1
- c) var: .int 1
- d) .int: var 1

**7. La línea de código ensamblador: mov \$msg, %rsi**

- a) Copia en rsi los primeros 64 bits de memoria desde la posición apuntada por la etiqueta msg.
- b) Copia en rsi todo el contenido de la cadena apuntada por msg.
- c) Copia en rsi la dirección de memoria de 64 bits almacenada en memoria a partir de la posición indicada por la etiqueta msg.
- d) Copia en rsi los 64 bits de la dirección msg.

**8. ¿Cuál de los siguientes fragmentos es correcto para comenzar un programa en ensamblador que conste de un solo archivo.s ?**

- a) .text:  
\_start:
- b) .text  
.local \_start

```
_start:  
c) .text  
.start _global  
  
_start:  
• d) .text  
.global _start  
e. _start:
```

**9. El punto de entrada de un programa ensamblador en GNU/as Linux x86 se llama**

- a) main
- b) \_start
- c) \_init
- d) begin

**10. El sufijo q de la instrucción movq significa:**

- a) Que la instrucción realiza un movimiento rápido, trabajando con la mitad menos significativa de los operandos (move quick).
- b) Que la instrucción realiza un movimiento rápido, trabajando con la mitad más significativa de los operandos (move quick).
- c) Que la instrucción trabaja con operandos de 64 bits (quad word).

[T2.4.1Arrays] tr.4, byte/word/double/quad usa sufijos b/w/l/q

- d) Que la instrucción trabaja con operandos de 128 bits (quad word).

**11. Invocando a printf de libC (SystemV AMD64) desde ensamblador...**

- a) el primer argumento debe ponerse en %rax  
en %rdi
- b) el segundo argumento es el formato  
el primero
- c) si se desean imprimir tres long, el tercero debe ponerse en %rdx  
en %rcx
- d) si se desean imprimir cuatro int, el cuarto debe ponerse en %r8d  
diane's silk dress costs 89  
formato en rdi, ints en esi, edx, ecx, r8d

**12. ¿Cuál de los siguientes fragmentos es correcto para comenzar un programa en ensamblador que conste de un solo archivo .s?**

a) section .text  
.start \_global  
\_start:  
b) .section .text  
.local \_start  
\_start:  
• c) .section .text  
.global \_start  
\_start:

Práct.1, tr.11-14

d) .text:

\_start:

añadido (:) a (.text) para evitar subterfugios:

- a) puede que .global \_start se añada más tarde (y funciona)
- b) aunque no se ponga .global, Id supone dirección por defecto (y funciona)

Añadiendo (:) es error seguro (.text already defined)

### **13. El lenguaje máquina...**

- a) es un conjunto de nombres simbólicos o nemotécnicos.
- b) facilita la portabilidad de los programas.
- c) es el mismo para todos los computadores.

• d) Ninguna de las respuestas anteriores es correcta.

### **14. El sufijo l de la instrucción movl significa:**

- a) Que la instrucción trabaja con operandos de 32 bits (long word).  
correcto
- b) Que la instrucción usa ordenación de bytes little-endian en lugar de big-endian.  
IA32 y AMD64 son little-endian, las instrucciones no pueden escoger "endianness"
- c) Que la instrucción afecta a los 16 bits menos significativos de los operandos (low word).  
16 bits sería movw
- d) Que la instrucción afecta a la parte de 16 bits más a la izquierda de los operandos (left word).  
ni siquiera tienen nombre la parte superior de los registros de 32 bits

## **15 .¿Qué es el lenguaje máquina?**

- a) Conjunto de datos binarios que representan señales eléctricas internas de la unidad de control de un microprocesador.
- no, representan el programa a ejecutar
- b) Conjunto de sentencias en un lenguaje escrito que se utilizan para generar programas codificados en lenguaje ensamblador.
- c) Conjunto formado por las siglas asignadas a las instrucciones del repertorio de instrucciones más un conjunto de directivas que facilitan la generación del código binario.
- d) Conjunto de instrucciones en formato binario que entiende un determinado procesador.

## **16.Alguna de las siguientes líneas de código sirve para definir una variable entera llamada tam en GNU/as Linux x86. ¿Cuál?**

a) int tam;

[E14Feb] pra.07 - editada, original decía "gcc/as" e introducía una ambigüedad indeseada con lenguaje C

- b) tam: .int .-msg

Práct.1, tr.12

c) var tam : integer;

d) \_int tam = 0

## **17.El primer parámetro de printf:**

- a) es un número en coma flotante
- b) puede ser de cualquier tipo
- c) es un entero
- d) es un puntero

## **18.La etiqueta del punto de entrada a un programa ensamblador en el entorno de las prácticas 1 a 4 (GNU/as Linux x86) es:**

a) \_main

- b) \_start

P3 y P4 se redactan en C.

P1 y P2 sí son en ensamblador GNU/as Linux x86.

Incluso en P2 se llega a usar main para ensamblar con gcc ya que usamos printf.

En cualquier caso, las otras opciones son descabelladas

c) .L0

d) \_init

## **19.¿De qué tipo son los procesadores Intel que usamos en los laboratorios?**

- a) puede ajustarse mediante un bit de control en el registro CR0 que funcionen como little- o big-endian

- b) little-endian

Práct.1, tr.18

- c) el concepto de endian no es aplicable a estas máquinas, ya que un registro del procesador no cabe en una posición de memoria

d) big-endian

## Sesión 2

**1. Si ECX vale 0, la instrucción adc \$0,%ecx**

a) Pone CF=1

-> b) Pone CF=0

c) Cambia CF

d) No cambia CF

**2. Alguno de los siguientes no es un nombre de registro en una máquina x86-64 en modo 64 bits**

a) r8d

b) r12w

-> c) sih

%sil en todo caso, no %sih

Ver libro Hallaron Figura 3.35

d) spl

**3. Alguno de los siguientes no es un nombre de registro en una máquina IA32 en modo 32 bits**

a) ebp

b) ax

c) dh

-> d) sil

Sí lo sería en modo 64 bits

Ver libro Hallaron Figura 3.35

**4. En el contexto de una llamada a función cdecl: si los contenidos de ESP y EBP son, respectivamente, 0x0008d040 y 0x00000000 antes de ejecutar una instrucción call, tras ejecutar el ret correspondiente, los contenidos serán:**

a) 0x0008d044 y 0x00000000

b) 0x0008d03c y 0x0008d03c

c) 0x0008d040 y 0x0008d040

-> d) 0x0008d040 y 0x00000000

**5. ¿Qué valor contendrá %edx tras ejecutar las siguientes instrucciones?**

```
xor %eax, %eax  
sub $1, %eax  
cld  
idiv %eax
```

a) no puede saberse con los datos del enunciado  
b) -1  
c) 1  
-> d) 0

**6. ¿Cuál de las siguientes no es una sección de un fichero ELF?**

- a) .bss  
para datos sin inicialización, mencionada en P2 Apéndice 2 Tabla 9  
b) .data  
para datos inicializados como en la Práctica "media"  
-> c) .static  
inventada  
d) .text  
para el código

**7. En contraposición a un ejecutable Linux ELF, un fichero objeto (obtenido con gcc -c)**

- a) no reserva espacio para las direcciones de objetos desconocidos; p.ej., sólo al enlazar un CALL a printf se insertan los 4B de la dirección de printf entre el codop de CALL y el de la siguiente instrucción máquina  
el espacio reservado para símbolos reubicables/indefinidos aparece lleno con 0  
b) contiene tablas de símbolos, pero sólo para los símbolos locales; p.ej., sólo al enlazar un programa que llame a printf se añadirá una entrada a la tabla con la dirección de printf  
los símbolos no resueltos/no definidos aparecen etiquetados U / \*UND\*  
-> c) ubica el código (y los datos) a partir de la posición 0x0, las direcciones definitivas sólo se calculan tras enlazar  
d) no contiene tablas de símbolos, que sólo se añaden al ejecutable tras enlazar  
objdump -t muestra la tabla de símbolos, de ejecutables y/o objetos

**8. En X86-64, el registro contador de programa se denomina:**

- a) R15
- b) IP
- c) EIP
- > d) RIP

**9. ¿Cuántos operandos acompañan a la instrucción PUSH en IA32?**

- a) 0
- > b) 1
- c) 2
- d) 3

**10. Dada la siguiente definición de datos:**

```
lista: .int 0x10000000, 0x50000000,  
       0x10000000, 0x20000000  
longlista: .int (. -lista)/4  
resultado: .quad 0x123456789ABCDEF  
formato: .ascii "suma=%llu=%llx hex\n\0"
```

la instrucción movl longlista, %ecx copia el siguiente valor:

- a) 8
- > b) 4
- c) 16
- d) 32

**11. ¿Qué modificador (switch) de as hace falta para ensamblar una aplicación de 32bits en un sistema de 64bits en el que se ha instalado también el compilador de 32bits?**

- a) -m32
- > b) -32
- c) -m elf\_i386
- d) Ninguna de las anteriores respuestas es correcta

**12. En la práctica "media" se programa la suma de una lista de 32 enteros de 4 B para producir un resultado de 8 B, primero sin signo y luego con signo. Si la lista se rellena con el valor que se indica a continuación, ¿en qué caso ambos programas producen el mismo**

**resultado?**

a) 0x9999 9999

0x0000 0013 3333 3320 != 0xffff fff3 3333 3320

b) 0xAAAA AAAA

0x0000 0015 5555 5540 != 0xffff fff5 5555 5540

c) 0xFFFF FFFF

0x0000 001f ffff ffe0 != 0xffff ffff ffff ffe0

-> d) 0x1111 1111

resultado 0x0000 0002 2222 2220

porque es positivo incluso en complemento a 2

todos los demás valores se interpretan como negativos, lo primero que hace la suma con signo

es extenderlos a 64bit de manera que se activan los 32 bits superiores... resultado

radicalmente distinto

Recordar que multiplicar por 32 es desplazar 5 posiciones a la izquierda

**13. Compilar .s → ejecutable, usando sólo as y ld, sin gcc...**

a) No se puede

-> b) Se puede, usando en as y ld los modificadores (switches) que corresponda

c) Basta usar as, con los modificadores que corresponda

d) Basta usar ld, con los modificadores que corresponda

**14. Si ECX vale 0, la instrucción adc \$-1,%ecx**

-> a) No cambia CF

si CF=1 al sumar 0xff...ff(=4G-1) sale CF=1, si no, no

b) Cambia CF

c) Pone CF=0

d) Pone CF=1

**15. ¿Cuál de los siguientes grupos de instrucciones IA32 sólo modifican los indicadores de estado sin almacenar el resultado de la operación?**

a) IMUL, IDIV

-> b) CMP, TEST

c) ADC, SBB

d) AND, OR, XOR

**16. Los switches --32 y --64 para trabajar en 32bit/64bit corresponden a la herramienta...**

a) ld

sería -melf\_i386 y -melf\_x86\_64

-> b) as

c) gcc

sería -m32 y -m64

d) nm

no tiene 32/64 bits

**17. ¿Qué modificador (switch) de gcc hace falta para compilar .c→.s (de fuente C a fuente ASM)?**

-> a) gcc -S

b) gcc -s

c) gcc -c

d) Eso no se puede hacer con gcc

**18. En x86-64 se pueden referenciar los registros**

a) %r12q, %r12d, %r12w, %r12l

sería %r12 y %r12b, no %r12q ni %r12l

b) %r8, %r8d, %r8w, %r8l

sería %r8b, no %r8l

c) %rsi, %esi, %si, %sih, %sil

no existe %sih

-> d) %rax, %eax, %ax, %ah, %al

Ver libro Hallaron Figura 3.35.

%ah no sería compatible con los nuevos nombres x86-64 como %sib, %r8b.

**19. En IA32, tras dividir 0x640000 (64 bits) entre 0x8000 (32 bits), el resultado será:**

a) 0x0 en EAX y 0xC8 en EDX

mal: cociente 0 (saber que cociente es en EAX)

b) 0x0 en AX y 0xC8 en DX

mal: AX/DX

c) 0xC8 en AX y 0xC8 en DX

mal: resto no cero (0x640 es múltiplo de 0x8)

-> d) 0xC8 en EAX y 0x0 en EDX

por exclusión, y además  $0x640 \gg 3 = 0xC8$

**20. En la práctica "media" se pide calcular la media y resto de una lista de 32 enteros CON signo de 32bits en una plataforma de 32bits sin perder precisión, esto es, evitando desbordamiento. ¿Qué (media : resto) se debe obtener para una lista rellena a -1 salvo el primer elemento, que valiera -31?**

a) (-2 : 2)

b) (-2 : 1)

c) (-1 :-31)

-> d) (-1 :-30)

**22. En la práctica "media" se desea invocar desde lenguaje ensamblador la función printf() de libC. Eso implica que este programa, como todo programa que use libC,**

a) es ventajoso para ensamblarlo que contenga main, y entonces es conveniente enlazarlo usando gcc (aunque ambas cosas son opcionales)

No. Que exista main o \_start sólo se comprueba a la hora de linkar. Y lo apropiado es que un programa ensamblador empiece por \_start. Es el runtime de C quien define un \_start que invoca a main.

-> b) es ventajoso para enlazarlo usar gcc, y entonces es conveniente que contenga main (aunque ambas cosas son opcionales)

Sí. Ver comentarios para formato, main y call printf en P2 Figura 12 y P3 Figura 4

c) es obligatorio que contenga main (y entonces es más cómodo usar gcc para enlazar)

No. Se puede usar ld -lc -dynamic-linker, sin incluir runtime

d) es obligatorio enlazarlo usando gcc (y entonces es más cómodo que contenga main)

**21. El volcado mostrado abajo se ha obtenido con el comando...**

00000000 <main>:

0: 55 push %ebp

1: 89 e5 mov %esp,%ebp

3: 83 e4 f0 and \$-16,%esp

6: 83 ec 10 sub \$0x10,%esp

9: c7 44 24 movl \$3, 4(%esp)

c: 04 03 00 00 00  
11: c7 04 24 movl \$0x1,(%esp)  
14: 01 00 00 00  
18: e8 fc ff ff ff call <main+0x19>

1d: c9 leave

1e: c3 ret

a) objdump -t p2.o

-> b) objdump -d p2.o

-d porque es un desensamblado (no cabeceras -h ni tablas -t)

p2.o porque las direcciones empiezan en 0 (un ejecutable x86 empezaría en 0x080480xx)

c) objdump -d p

d) objdump -h p

**23. ¿Qué modificador (switch) de gcc hace falta para compilar .s → .o sin llamar al enlazador?**

a) Eso no se puede hacer con gcc

-> b) gcc -c

c) gcc -s

d) gcc -S

**24. ¿Qué hace gcc -O?**

-> a) Compilar con optimización suave

b) Compilar .c→.o (fuente C a objeto)

c) Compilar .s→.o (fuente ASM a objeto)

d) Ambas (b) y (c), según la extensión de los ficheros que se usen como argumentos

**25. Compilar .c→exe (de fuente C a ejecutable) usando sólo as y ld, sin gcc...**

-> a) No se puede

b) Se puede, repartiendo entre as y ld los modificadores (switches) que corresponda

c) Se puede, repartiendo modificadores entre as y ld, y añadiendo al comando ld el runtime de C

d) Basta usar ld, con los modificadores de gcc que corresponda, y añadiéndole el runtime de C

**26. ¿Qué modificador (switch) de gcc hace falta para compilar .c → .o (de fuente C a código objeto)?**

a) Eso no se puede hacer con gcc

-> b) gcc -c

c) gcc -o

d) gcc -O

**27. ¿Qué hace gcc -O1?**

a) Compilar .c→.o (fuente C a objeto)

b) Compilar .s→.o (fuente ASM a objeto)

c) Ambas (a) y (b), según la extensión de los ficheros que se usen como argumentos

-> d) Compilar con optimización

**28. En la práctica “media” se pide sumar una lista de 32 enteros \*con\* signo de 32bits en una plataforma de 32bits sin perder precisión, esto es, evitando overflow. ¿Cuál es el menor valor positivo que repetido en toda la lista causaría overflow con 32bits?**

a) 0x8000 0000

ya es el negativo más grande, en cuanto se le sumara -1 saldría overflow

b) 0x4000 0000

en cuanto se hiciera la primera suma saldría negativo

-> c) 0x0400 0000

$n * 32 == n << 5$  y el bit 26 (0x04...) pasaría a 31 (0x80...) y el resultado llegaría \*justo\* a convertirse en negativo. Algo más pequeño que 0x04... no llegaría.

d) 0x0800 0000

se perdería incluso el bit de signo

**29. ¿Qué hace gcc -O0?**

a) Compilar .c→.s (C→ASM sin generar objeto)

b) Compilar .s→.o (fuente ASM a objeto)

c) Compilar .c→.o (fuente C a objeto)

-> d) Compilar sin optimización

**30. Tras ejecutar las tres instrucciones que se muestran desensambladas a continuación, el registro EAX toma el valor**

08048074 <\_start>:

8048074: be 74 80 04 08 mov \$\_start, %esi

8048079: 46 inc %esi

804807a: 8b 06 mov (%esi), %eax

- > a) 0x08048074
- b) 0x08048075
- c) 0x08048079
- d) 0x0804807a

**31. ¿Qué modificador (switch) de gcc hace falta para compilar una aplicación de 32 bits en un sistema de 64 bits?**

- a) -64
- b) -32
- > c) -m32
- d) -m64

**32. En la práctica "media" se pide sumar una lista de 32 enteros SIN signo de 32bits en una plataforma de 32bits sin perder precisión, esto es, evitando perder acarreos. ¿Cuál es el mínimo valor entero que repetido en toda la lista causaría acarreo con 32bits (sin signo)?**

- a) 0xfc00 0000
- b) 0xfbff ffff
- c) 0x07ff ffff
- > d) 0x0800 0000

**33. Como parte del proceso de compilación de una aplicación en lenguaje C, enlazar .o → .exe (de objeto proveniente de fuente C a ejecutable) usando sólo as y ld, sin gcc...**

- a) Ninguna de las anteriores respuestas es correcta
- > b) Basta usar ld, con los modificadores de gcc que corresponda, y añadiéndole el runtime de C
- c) Se puede, repartiendo entre as y ld los modificadores (switches) que corresponda
- d) Se puede, repartiendo modificadores entre as y ld, y añadiendo al comando ld el runtime de C

**34. Los switches –m elf\_i386 y –m elf\_x86\_64 para trabajar en 32 bits / 64 bits corresponden a la herramienta...**

- a) nm
- > b) ld
- c) gcc
- d) as

**35. ¿Cuál de los siguientes es el orden correcto en el ciclo de compilación de un programa en lenguaje C? (el fichero sin extensión es un ejecutable):**

- a) file.s → file.c → file → file.o
- > b) file.c → file.s → file.o → file
- c) file.s → file.c → file.o → file
- d) file.c → file.s → file → file.o

**36. ¿Cuál de los siguientes contenidos no está incluido en un fichero ELF ejecutable?**

- a) código máquina
- sección .text
- b) variables globales
- secciones .data/.bss
- c) tabla de símbolos
- incluso símbolos de depuración con -g
- > d) pila del usuario

**37. La siguiente línea en la sección de datos de un programa en ensamblador de IA32**

result: .int 0,0

- a) Reserva espacio para un único entero, inicializado a 0, en la posición de memoria 0
- b) Reserva espacio para un único entero, inicializado a 0,0
- > c) Reserva espacio para dos enteros, inicializados ambos a 0
- d) Reserva espacio para un entero, inicializado a 0, seguido de un dato de tamaño indefinido, también inicializado a 0

**38. Considerar los siguientes dos bloques de código almacenados en dos ficheros distintos:**

```
/* main.c */
```

```
int i = 0;  
  
int main() {  
    func();  
    return 0;  
}
```

```
/* func.c */
```

```
int i = 1;  
  
void func() {
```

```
printf("%d", i);
}
```

¿Qué sucederá cuando se compile, enlace y ejecute este código?

- > a) Error al compilar o enlazar, no se obtiene ejecutable
- b) Escribe “0”
- c) A veces escribe “0” y a veces “1”
- d) Escribe “1”

**39. Dada la siguiente definición de datos:**

```
lista: .int 0x10000000, 0x50000000,
       0x10000000, 0x20000000
longlista: .int (. - lista)/4
resultado: .quad 0x123456789ABCDEF
formato: .ascii "suma=%llu=%llx hex\n\0"
```

La instrucción para copiar la dirección de memoria donde comienza lista en el registro EBX es:

- a) movl \$lista, (%ebx)
- b) movl lista, %ebx
- c) movl (lista), %ebx
- > d) movl \$lista, %ebx

**40. ¿Qué modificador (switch) de gcc hace falta para compilar una aplicación de 32bits (fuente C) en un sistema de 64bits en el que se ha instalado también el compilador de 32bits?**

- a) No se puede conseguir ese efecto con modificadores de gcc, porque el gcc por defecto (/usr/bin/gcc) es el de 64bits, se debe usar el gcc de la instalación alternativa (de 32bits)
- b) -m elf\_i386
- > c) -m32
- d) -32

**41. En la práctica “media” se pide sumar una lista de enteros \*con\* signo de 32 bits en una plataforma de 32 bits sin perder precisión, esto es, evitando overflow. ¿Cuál es el menor valor positivo que repetido en los \*dos\* primeros elementos de la lista causaría overflow con 32 bits al realizar la suma de \*esos dos\* primeros elementos de la lista?**

a) 0x8000 0000

es negativo, el enunciado pide menor valor positivo

-> b) 0x4000 0000

0x8000 0000 sería negativo en 32bit

c) 0x0400 0000

suma 0x0800 0000 sin problema

d) 0x0800 0000

suma 0x1000 0000 sin problema

**42. En la práctica "media" se pide sumar una lista de 32 enteros SIN signo de 32 bits en una plataforma de 32 bits sin perder precisión, esto es, evitando perder acarreos. Un estudiante entrega la siguiente versión**

```
# $lista en EBX, longlista en ECX
```

suma:

```
    mov $0, %eax
```

```
    mov $0, %edx
```

```
    mov $0, %esi
```

bucle:

```
    add (%ebx,%edx,4), %eax
```

```
    jnc seguir
```

```
    inc %edx
```

seguir:

```
    inc %esi
```

```
    cmp %esi,%ecx
```

```
    jne bucle
```

```
    ret
```

Esta función presenta una única diferencia frente a la solución recomendada en clase,

relativa al indexado en el array.

Esta función suma:

a) Produce siempre el resultado correcto

b) Fallaría con lista: .int 1,1,1,1, 1,1,1,1, ...

Coincide que sumar 32 veces lista[0]=1 es lo mismo que sumar los 32 elementos, porque todos

son iguales a lista[0]

-> c) Fallaría con lista: .int 1,2,3,4, 1,2,3,4, ...

Debería salir suma=80 y sale suma=32 porque suma 32 veces lista[0]=1.  $32 << 4G - 1$ , así que no hay acarreos ni por asomo.

d) No es correcta pero el error no se manifiesta en los ejemplos propuestos, o se manifiesta en ambos

**43. Dada la siguiente definición de datos:**

lista: .int 0x10000000, 0x50000000,

0x10000000, 0x20000000

longlista: .int (. - lista)/4

resultado: .quad 0x123456789ABCDEF

formato: .ascii "suma=%llu=%llx hex\n\0"

la llamada correcta a printf será:

a) push resultado

push resultado+4

push resultado

push resultado+4

push \$formato

call printf

add \$20, %esp

b) push resultado

push resultado+4

push \$formato

call printf

add \$12, %esp

c) push resultado+4

push resultado

push \$formato

call printf

add \$12, %esp

-> d) push resultado+4

```
push resultado  
push resultado+4  
push resultado  
push $formato  
call printf  
add $20, %esp
```

**44. ¿Cuál de los siguientes es el orden correcto en el ciclo de compilación de un programa en lenguaje C? (el fichero sin extensión es un ejecutable):**

- a) fich.c → fich.o → fich.s → fich
- b) fich → fich.s → fich.o → fich.c
- > c) fich.c → fich.s → fich.o → fich
- d) fich.c → fich.s → fich → fich.o

**45. ¿Qué modificador (switch) de ld hace falta para enlazar una aplicación de 32bits en un sistema de 64bits en el que se ha instalado también el compilador de 32bits?**

- a) No hace falta modificador, ld lo deduce del tipo de objeto a enlazar
- > b) -m elf\_i386
- c) -m32
- d) -32

**46. En la práctica "media" se pide sumar una lista de 32 enteros CON signo de 32 bits en una plataforma de 32 bits sin perder precisión, esto es, evitando desbordamiento. De entre los siguientes, ¿cuál es el valor negativo más pequeño (en valor absoluto) que repetido en toda la lista causaría desbordamiento con 32**

**bits (en complemento a 2)? Se usa notación decimal y espacios como separadores de millares/millones/etc.**

- a) -10 000 000

No llega,  $\text{abs}(-10 \text{ millones}) \leq 64M$

- > b) -100 000 000

Se pasa,  $100 \text{ millones} >> 64M$

- c) -1 000 000 000

- d) -10 000 000 000

**47. En la práctica “media” se pide sumar una lista de 32 enteros \*con\* signo de 32bits en una**

**plataforma de 32bits sin perder precisión, esto es, evitando overflow. ¿Cuál es el mayor valor negativo (menor en valor absoluto) que repetido en toda la lista causaría overflow con 32bits?**

- a) 0xffff ffff
- b) 0xfc00 0000

$nx32 == n << 5$  y quedaría 0x8000 0000 !!! justo para que no haya overflow

- c) 0xf000 0000

se pierde el signo con  $<<4$ , mucho más con  $<<5$

- > d) 0xfbff ffff

**48. En la práctica "media" un estudiante usa el siguiente bucle para acumular la suma en EBP:EDI antes de calcular la media y el resto**

bucle:

```
mov (%ebx,%esi,4), %eax
```

```
cltd
```

```
add %eax, %edi
```

```
adc %edx, %ebp
```

```
jnc nocarry
```

```
inc %edx
```

nocarry:

```
inc %esi
```

```
cmp %esi,%ecx
```

```
jne bucle
```

Este código es una mezcla inexplicada de las soluciones recomendadas para suma sin signo y para suma con signo. Estando bien programado todo lo demás, este código

- > a) produce siempre el resultado correcto

inc %edx se hace después de adc, cuando ya se ha usado el valor de %edx

Recordar que usamos cltd para extender a %edx el elemento leído en %eax

Igual podríamos haber hecho xor %edx,%edx y tampoco pasaría nada

- b) no siempre produce el resultado correcto, pero el error no se manifiesta en los ejemplos propuestos, o se manifiesta en ambos

- c) fallaría con lista: .int 0,1,2,3

d) fallaría con lista: .int -1,-2,-4,-8

**49. ¿Qué hace gcc -O?**

a) Compilar sin optimización, igual que -O0

b) Compilar .c → .o

-> c) Compilar con optimización, igual que -O1

d) Compilar .s → .o

**50. En la práctica "media" se suma una lista de 32 enteros de 4 B con signo para producir una media y un resto usando la instrucción IDIV. ¿Cuál de las siguientes afirmaciones es falsa?**

a) IDIV produce el mismo resto que el operador % en lenguaje C

división truncada ambos - resto del mismo signo que dividendo

b) IDIV produce el mismo cociente que el operador / en lenguaje C

división truncada ambos - resto del mismo signo que dividendo

c) El resto siempre tiene el mismo signo que la suma

división truncada IDIV - resto del mismo signo que dividendo

-> d) La media se redondea al entero más próximo

división truncada IDIV - no se redondea, se trunca

51. Para compilar un programa escrito en C en el entorno GNU/Linux se usa el programa:

a) gdb

b) objdump

-> c) gcc

d) ddd

**52. La(s) instrucción(es) necesaria(s) para cargar el dividendo 0xa30bf18a en la pareja**

**edx:eax como paso previo a una división sin signo son:**

a) movq \$0xa30bf18a,%rax

-> b) movl \$0xa30bf18a,%eax

xorl %edx,%edx

c) movl \$0xf18a,%eax

movl \$0xa30b,%edx

d) movl \$0xa30bf18a,%eax

cltd

53. En la práctica "media" se pide sumar una lista de 32 enteros CON signo de 32bits en una

plataforma de 32bits sin perder precisión, esto es, evitando desbordamiento. ¿Cuál es el valor negativo más pequeño (en valor absoluto) que repetido en toda la lista causaría desbordamiento con 32bits (en complemento a 2)?

- a) 0xf800 0000
- > b) 0xfbff ffff
- c) 0xfc00 0000
- d) 0xf800 0001

**54. Dada la siguiente definición de datos:**

lista: .int 0x10000000, 0x50000000,

0x10000000, 0x20000000

longlista: .int (.lista)/4

resultado: .quad 0x123456789ABCDEF

formato: .ascii "suma=%llu=%lx hex\n\0"

y suponiendo que hemos llamado a una función suma que devuelve un número de 64 bits en la pareja EDX:EAX, las instrucciones que copian ese número en resultado son:

- a) movl (%eax), resultado

movl (%edx), resultado+4

mov M-M imposible

- b) movl %eax, resultado+4

movl %edx, resultado

no es little-endian (tampoco es big-)

- c) movl (%eax), resultado+4

movl (%edx), resultado

mov M-M imposible

- > d) movl %eax, resultado

movl %edx, resultado+4

little-endian => primero el menos significativo

**55. El switch de gcc para que únicamente compile de lenguaje C a ensamblador, y no realice ningún paso adicional (ensamblar, enlazar, etc), es...**

- a) -o

para nombrar el ejecutable

-> b) -S

c) -c

de C/asm a objeto, .c/.s→.o

d) -g

para incorporar info depuración

**56. El switch -l para indicar librerías \*NO\* funciona con la herramienta...**

a) gcc

gcc pasa -L/-l a ld

-> b) as

c) ld

-L/-l son switches propios de ld

d) no se puede marcar una y solo una de las anteriores

**57. En la práctica "media" se programa la suma de una lista de 32 enteros de 4 B para producir un resultado de 8 B, primero sin signo y luego con signo. Si la lista se rellena con el valor 0x0400 0000, ¿en qué se diferencian los resultados de ambos programas?**

a) en uno ocupa 32 bits, en otro 64 bits

ambos resultados son de 64 bits

-> b) no se diferencian

0x0000 0000 8000 0000

el resultado cabría en 32bits sin signo

c) en uno se interpreta como negativo, en otro como positivo

ambos programas interpretan 0x0000 0000 8000 0000 como positivo

d) en uno los 32 bits superiores son 0xFFFF FFFF, en el otro no

en ambos los 32bits superiores son 0x0000 0000

**58. Haciendo definido en código fuente ensamblador longsal: .quad .-saludo justo detrás de un string saludo que ocupaba 28 bytes, si comparamos los comandos gdb siguientes: x/1xg &longsal frente a print (long) &longsal, se puede decir que:**

a. ambos nos muestran la longitud del string (que es/vale/equivale a 28)

-> b. el primero (x) nos muestra la longitud, y el segundo (print) nos muestra otro valor distinto

Práct.2, pág.7, Fig.11

c. el segundo (print) nos muestra la longitud, y el primero (x) nos muestra otro valor distinto

d. alguno o ambos contienen algún error gramatical (falta o sobra algún &, algún typecast (char\*) ó (long), etc)

**59. Habiendo definido en código fuente ensamblador saludo: .ascii "Hola mundo\n"), si comparamos los comandos gdb siguientes: x/s &saludo frente a print (char\*) &saludo, se puede decir que:**

- > a. ambos nos muestran el string (el mensaje "Hola mundo") Práct.2, pág.7, Fig.11
- b. el primero (x) nos muestra el string, y el segundo (print) nos muestra otro valor distinto
- c. el segundo (print) nos muestra el string, y el primero (x) nos muestra otro valor distinto
- d. alguno o ambos contienen algún error gramatical (falta o sobra algún &, algún typecast (char\*) ó (long), etc)

**60. En la práctica "media" se pide sumar una lista de 16 enteros \*sin\* signo de 32 bits (unsigned en C) evitando desbordamiento (sin signo), esto es, acarreo. ¿Cuál es el mayor valor que repetido en toda la lista \*no\* causaría acarreo en 32 bits (unsigned en C)?**

- a. 0xFFFF FFFF mayor que b.
- b. 0x7FFF FFFF mayor que c.
- c. 0x1000 0000 al desplazar 4 bits a la izquierda sale el 1 hacia el CF
- > d. 0x0FFF FFFF

**61. Habiendo definido en código fuente ensamblador saludo: .ascii "Hola a todos!\nHello World!\n"), para conseguir desde gdb cambiar el string para que la llamada al sistema WRITE escriba solo una línea en lugar de dos, se debería hacer...**

- a. nada, porque ya de por sí WRITE escribe saludo como una sola línea
- b. nada, desde gdb no se puede conseguir que WRITE escribe saludo como una sola línea
- c. p (char\*) &saludo+13 y también p (char\*) &saludo+14\_
- > d. set var \*((char\*)&saludo+13)=''

**62. En la práctica "media" se pide sumar una lista de 16 enteros \*sin\* signo de 32 bits (unsigned en C) evitando desbordamiento (sin signo), esto es, acarreo. ¿Cuál es el menor valor que repetido en toda la lista causaría acarreo en 32 bits (unsigned en C)?**

- a. 0xFFFF FFFF
  - b. 0x7FFF FFFF
  - > c. 0x1000 0000
- //al desplazar 4 bits a la izquierda sale el 1 hacia el CF

d. 0xFFFF FFFF

**63. En un sistema Linux x86-64, ¿cuál de las siguientes expresiones es equivalente a la expresión C  $(x[2] + 4)[3]$ ? Suponer que previamente se ha declarado int \*\*x. x[2] es int\*, y entonces da igual  $(X[2])[7] == (X[2]+7) == (X[2]+4)[3]$**

a.  $*((*x + 16)) + 28$

cierto si se hubiese hecho typecast (void\*)x para evitar aritmética punteros

b.  $*(((*x) + 2) + 7)$

$(*x)+2 == x[0]+2 != x[2]$

c.  $(*(*x + 2) + 4) + 3$

sería  $x[2][4]+3$ , la última suma sería suma entera, no aritmética punteros

d.  $*(((*x + 2) + 4) + 3)$

## Sesión 3

1. La práctica "parity" debía calcular la suma de paridades impar (XOR de todos los bits) de los elementos de un array. Un estudiante entrega la siguiente versión de parity3:

```
int parity3(unsigned* array, int len){  
    int i,res=0,val;  
    unsigned x;  
    for (i=0; i<len; i++){  
        x=array[i];  
        val=0;  
        do {  
            val += x;  
            x >>= 1;  
        } while (x);  
        val &= 0x1;  
        res+=val;  
    }  
    return res;  
}
```

Esta función parity3:

- > a) produce siempre el resultado correcto
- solemos escribir `res+=val&0x1`, en lugar de ponerlo en 2 sentencias C
- b) fallaría con `array={0,1,2,3}`
- c) fallaría con `array={1,2,4,8}`
- d) no siempre produce el resultado correcto, pero el error no se manifiesta en los ejemplos propuestos, o se manifiesta en ambos

2. ¿Cuál es el `popcount` (peso Hamming, nº de bits activados) de una lista de N números inicializada con los valores 0..N-1?

- a)  $(N-1)*N/2$

no, eso es la fórmula de Gauss para la suma de la progresión aritmética

- b)  $N*(N+1)/2$

no, eso sería la suma de la progresión aritmética de 1..N

c) si N es par,  $N^*(N/2)$

no, en todo caso  $N/2$  sería la suma de paridades

-> d) si N es potencia de 2,  $(\log_2(N)*N)/2$

todas las combinaciones de  $\log_2(N)$  bits, en cada posición de bit la mitad son 0 y la otra mitad son 1

p.ej.: LSB mitad par mitad impar, MSB mitad empieza por 0 mitad por 1, etc...

3. Suponga la siguiente sentencia asm en un programa:

```
asm(" add (%[a],%[i],4),%[r]"  
    :[r] "+r" (result)  
    :[i] "r" (i),  
    [a] "r" (array)  
);
```

¿Cuál de las siguientes afirmaciones es incorrecta?

a) se desea que el valor calculado por la instrucción ensamblador quede almacenado en la variable C result

correcto, add ...,%[r] almacena en result

b) r es un registro de entrada/salida

ok, "+r"

c) a es un registro de entrada

ok, "r"

-> d) i es un registro de salida

no, asm("sentence":output:input:clobber), [i] va en input

4. La práctica "popcount" debía calcular la suma de bits (peso Hamming) de los elementos de un array. Un estudiante entrega la siguiente versión de popcount4:

```
int popcount4(unsigned* array,
```

```
    int len){
```

```
    int i,j;
```

```
    unsigned x;
```

```
    int result = 0;
```

```
    for(i=0;i<len;i++){
```

```
        x=array[i];
```

```
for(j=0;j<8;j++){
    result += x & 0x01010101;
    x>>=1;
}
}

result += (result >> 16);
result += (result >> 8);
return result & 0xFF;
}
```

Esta función presenta varias diferencias con la versión "oficial" recomendada en clase, incluyendo la ausencia de una variable auxiliar val y la diferente posición de los desplazamientos  $>>$  y máscara 0xFF.

Esta función popcount4:

a) produce siempre el resultado correcto

No. Las sumas paralelas pueden llevar acarreos al siguiente byte. Y la suma total puede ser superior a 255, cosa que no permite el código mostrado.

b) fallaría con array={0,1,2,3}

No. La suma total es 4, no hay acarreo al byte superior ( $4 << 256$ ), quedan sin efecto las instrucciones erróneas o cambiadas de sitio ( $>>16$ ,  $>>8$ ,  $\&0xff$ )

c) fallaría con array={1,2,4,8}

No. La suma total es 4, no hay acarreo al byte superior ( $4 << 256$ ), quedan sin efecto las instrucciones erróneas o cambiadas de sitio ( $>>16$ ,  $>>8$ ,  $\&0xff$ )

-> d) no es correcta pero el error no se manifiesta en los ejemplos propuestos, o se manifiesta en ambos

Efectivamente.

El ejemplo más pequeño para llegar al error (suma paralela menos significativa  $>= 256$ ) sería lista[32]={0xff, 0xff, ... 0xff}

En ese caso, calcula result=256, le suma 1 (result=257) y lo enmascara: result=1

Las 4 sumas en paralelo de  $x \& 0x01010101$  deben hacerse sobre una variable inicialmente a cero, val +=  $x \& 0x01010101$ , y luego acumularse val += val  $>> 16$  y 8, antes de acumular la suma parcial val&0xFF a result.

5. La práctica "parity" debía calcular la suma de paridades impar (XOR de todos los bits) de los elementos de un array. Un estudiante entrega la siguiente versión de parity6:

```
int parity6(unsigned* array, int len){  
    int i,j,res=0;  
    unsigned x;  
    for (i=0; i<len; i++){  
        x=array[i];  
        asm("\n"  
            "mov %[x],%%edx \n\t"  
            "shr $16, %%edx \n\t"  
            "xor %%edx,%[x] \n\t"  
            "mov %[x],%%edx \n\t"  
            "mov %%dh, %%dl \n\t"  
            "xor %%edx, %[x]\n\t"  
            "setpo %%cl \n\t"  
            "movzx %%cl, %[x]"  
            :[x] "+r" (x)  
            :  
            :"edx","ecx"  
        );  
        res+=x;  
    }  
    return res;  
}
```

Esta función parity6:

-> a) produce siempre el resultado correcto

bastaba con un sobrescrito "edx", y con 6 instrucciones máquina (no 8)

en concreto, la 3<sup>a</sup> instrucción podría haber sido xor %[x],%%edx y ya sólo quedaría operar con %dh y %dl, y no usar %cl sino el propio %dl

b) fallaría con array={0,1,2,3}

c) fallaría con array={1,2,4,8}

d) no siempre produce el resultado correcto, pero el error no se manifiesta en los ejemplos propuestos, o se manifiesta en ambos

6. La práctica "popcount" debía calcular la suma de bits (peso Hamming) de los elementos de un array. Un estudiante entrega la siguiente versión de `popcount4`:

```
int popcount4(unsigned* array, int len)
```

```
{  
    int i, j, res = 0;  
  
    for(i = 0; i < len; ++i) {  
        unsigned x = array[i];  
  
        int n = 0;  
        do {  
            n += x & 0x01010101L;  
            x >>= 1;  
        } while(x);  
  
        for(j = 16; j == 1; j /= 2){  
            n ^= (n >>= j);  
        }  
        res += n & 0xff;  
    }  
  
    return res;  
}
```

Esta función `popcount4`:

-> a) produce el resultado correcto

Caso real, entregado en prácticas. La máscara está pensada para `for(j=0;j<8;j++)`. En lugar de eso, se hace `do...while(x)`, con lo cual no se ahoran iteraciones y todo el resultado queda acumulado en el LSB de `n`. El `for(j)` es absurdo, no itera ninguna vez. El resultado se extrae y acumula con `n&0xFF`. Es correcto, pero no mejora la eficiencia. `popcount2` es igual de eficiente y más elegante, porque no tiene código superfluo.

b) fallaría con `array={0,1,2,3}`

c) fallaría con `array={1,2,4,8}`

d) no es correcta pero el error no se manifiesta en los ejemplos propuestos, o se manifiesta en

ambos

7. ¿Cuál es la paridad (XOR "lateral" de todos los bits) del número 29?

-> a) 0

$29 = 0x1D = 16+8+4+1 = 0b0001\ 1101 \rightarrow$  par  $\rightarrow$  xor sale 0

b) 1

c) 2

d) 4

8. La práctica "parity" debía calcular la suma de paridades impar (XOR de todos los bits) de los elementos de un array. Un estudiante entrega la siguiente versión de parity4:

```
int parity4(unsigned* array, int len){  
    int val,i,res=0;  
    unsigned x;  
    for (i=0; i<len; i++){  
        x=array[i];  
        val=0;  
        asm("\n"  
            "ini3: \n\t"  
            "xor %[x],%[v] \n\t"  
            "shr %[x] \n\t"  
            "test %[x], %[x]\n\t"  
            "jne ini3 \n\t"  
            "[v]"+"r" (val)  
            "[x] "r" (x)  
        );  
        val = val & 0x1;  
        res+=val;  
    }  
    return res;  
}
```

La sentencia `asm()` del listado anterior tiene las siguientes restricciones

a) ninguna

b) arquitectura de 32 bits

-> c) dos entradas y una salida

[v] cuenta como salida y entrada, [x] como entrada

d) un registro y dos sobrescritos (clobber)

9. La práctica "parity" debía calcular la suma de paridades impar (XOR de todos los bits) de los elementos de un array. Un estudiante entrega la siguiente versión de parity5:

```
int paridad5(unsigned* array, int len) {  
    int i, k, result = 0;  
    unsigned x;  
    for (i = 0; i < len; i++) {  
        x = array[i];  
        for (k = 16; k == 1; k /= 2)  
            x ^= x >> k;  
        result += (x & 0x01);  
    }  
    return result;  
}
```

Esta función paridad5:

a) es correcta

b) falla para array={0,1,2,3}

-> c) falla para array={1,2,3,4}

d) no se puede marcar una y sólo una de las opciones anteriores

10. En la convención cdecl estándar para arquitecturas x86 de 32 bits, el resultado de una función se devuelve usualmente en el registro:

a) ESI

b) EBP

-> c) EAX

d) EBX

11. ¿Cuál de los siguientes registros tiene que ser salvaguardado (si va a modificarse) dentro de una subrutina según la convención cdecl para IA32?

a) eax

- b) ecx
- > c) ebx
- d) edx

12. ¿En qué registro se pasa el primer argumento a una función en Linux gcc x86-64?

- > a) edi
- b) edx
- c) esi
- d) ecx

13. ¿Cuál de las siguientes afirmaciones es falsa respecto al lenguaje C (en convención cdecl x86)?

- a) Pasar a una función un puntero a una variable se traduce en introducir en la pila el valor de la dirección de memoria donde está almacenada la variable exacto, puntero es dirección de...
- b) Al llamar a una subrutina o función se introducen los parámetros en la pila y después se realiza una llamada a la subrutina sí, suponiendo convención cdecl x86, porque x86-64 usa regs.
- > c) Antes de volver de la rutina llamada, el programa en C se encarga de quitar de la pila los parámetros de llamada realizando varios pop no, quita parámetros el invocante, tras retornar de la llamada
- d) Los parámetros se introducen en la pila en el orden inverso a como aparecen en la llamada de C, es decir, empezando por el último y acabando por el primero sí, de derecha a izquierda

14. La práctica "parity" debía calcular la suma de paridades impar (XOR de todos los bits) de los elementos de un array. Un estudiante entrega la siguiente versión de parity5:

```
int parity5(unsigned * array,
           int len){
    int i,j, result = 0;
    unsigned x;
    for(i = 0; i<len; i++){
        x=array[i];
        for(j=1; j<8*sizeof(int); j*=2)
```

```

x ^= x >> j;
result += x & 0x1;
}
return result;
}

```

Esta función sólo se diferencia de la versión "oficial" recomendada en clase, en las condiciones del bucle for interno.

Esta función parity5:

-> a) Produce siempre el resultado correcto

Efectivamente, con este for también se cumple que el LSB (bit menos significativo) sigue siendo el XOR lateral de todos los bits.

b) Fallaría con array={0,1,2,3}

c) Fallaría con array={1,2,4,8}

d) No es correcta pero el error no se manifiesta en los ejemplos propuestos, o se manifiesta en ambos

El bucle sugerido por el guión de prácticas hubiera sido:

```
for (j=8*sizeof(unsigned); j>0; j>=1)
```

15. Suponga la siguiente sentencia asm en un programa:

```

asm(" add (%[a],%[i],4),%[r]"
:[r] "+r" (result)
:[i] "r" (i),
[a] "r" (array)
);

```

¿Cuál de las siguientes afirmaciones es correcta?

- a) el código de retorno de la función asm se fuerza a que esté en la variable result
- b) a es una posición de memoria de entrada
- c) r es una posición de memoria de entrada/salida
- > d) i es un registro de entrada

16. La práctica "popcount" debía calcular la suma de bits (peso Hamming) de los elementos de un array. Un estudiante entrega la siguiente versión de `popcount3`:

```
int popcount3(unsigned* array,
```

```

int len){

int i, res = 0;

unsigned x;

for( i = 0; i < len; i++ ) {

x = array[i];

asm("ini3: \n"
"shr %[x] \n"
"adc $0, %[r] \n"
"add $0, %[x] \n"
"jne ini3 \n"
:[r] "+r" (res)
:[x] "r" (x );
}

return res;
}

```

Esta función sólo tiene una diferencia con la versión "oficial" recomendada en clase. En concreto, una instrucción máquina en la sección `asm()` es distinta.

Esta función `popcount3`:

-> a) produce siempre el resultado correcto

Si sumar  $x+0$  activa ZF sólo puede ser porque ya era  $x==0$ , así que la lógica es equivalente a la deseada.

b) fallaría con  $\text{array}=\{0,1,2,3\}$

c) fallaría con  $\text{array}=\{1,2,4,8\}$

d) no es correcta pero el error no se manifiesta en los ejemplos propuestos, o se manifiesta en ambos

Para acabar el bucle cuando  $x==0$  hubiéramos esperado que se usara `test %[x], %[x]`, que activa el flag ZF si el resultado es 0. Notar que `and(x,x)` sólo puede ser 0 si ya era  $x==0$

17. ¿Cuál de las siguientes afirmaciones es cierta respecto al lenguaje C?

a) Pasar a una función un puntero a una variable se traduce en introducir en la pila el valor de la variable

`ptr!=val`

-> b) En lenguaje C, al llamar a una subrutina o función se introducen los parámetros en la pila y después se realiza una llamada a la subrutina suponiendo convención cdecl x86, porque x86-64 usa regs.

c) Los parámetros se introducen en la pila en el orden en el que aparecen en la llamada de C, es decir, empezando por el primero y acabando por el último no, de derecha a izquierda

d) Antes de volver de la rutina llamada, el programa en C se encarga de quitar de la pila los parámetros de llamada realizando varios pop no, quita parámetros el invocante, tras retornar de la llamada

18. ¿Cuál es la suma de paridades (suma de los XOR "laterales" de los bits de cada número) de una lista de N números inicializada con los valores 0..N-1?

- a)  $(N-1)*N/2$  //no, eso es la fórmula de Gauss para la suma de la progresión aritmética
- b)  $N*(N+1)/2$  //no, eso sería la suma de la progresión aritmética de 1..N
- > c) si N es par,  $N/2$  //exacto, al cambiar el LSB cambia la paridad
- d) si N es potencia de 2,  $(\log_2(N)*N)/2$  //eso sería el popcorn

19. ¿Cuál es la paridad (XOR "lateral" de todos los bits) del número 19?

a) 0

-> b) 1

$19 = 0x13 = 16+3 = 0b0001\ 0011 \rightarrow$  impar -> xor sale 1

c) 2

d) 3

20. La práctica "popcount" debía calcular la suma de bits (peso Hamming) de los elementos de un array. Un estudiante entrega la siguiente versión de popcorn3:

```
int popcorn3(unsigned* array,
            int len){
    int i, res = 0;
    unsigned x;
    for( i=0; i<len; i++ ) {
        x = array[i];
        asm("ini3: \n"
            "shr %[x] \n"
            : [x] "r" (x)
            : : );
        res += x;
    }
    return res;
}
```

```

"adc $0, %[r] \n"
"add $0, %[x] \n"
"jne ini3 \n"
:[r] "+r" (res)
:[x] "r" (x );
}
return res;
}

```

Esta función produce siempre el resultado correcto, a pesar de que una instrucción máquina en la sección asm() es distinta a la que se esperaba después de haber estudiado pcount\_r en teoría. La instrucción distinta también se podría haber cambiado por...

a) cmp %[x], %[r] //no sirve

b) sbb \$0, %[r] //¿se pretende sustituir la segunda instrucción?

redacción original adc \$0, %[x] hubiera servido para sustituir la 3ª instrucción, así que cambiamos por esta otra que no sirve (usa %[r]).

c) sar %[x]

¿se pretende sustituir la primera instrucción?

-> d) test %[x], %[x]

así viene en pcount\_r (Tema 2.3 tr.38)

se esperaba que la 3ª instrucción, antes de jne, fuera test, para comprobar si x==0. La instrucción add hace lo mismo de una forma más difícil.

21. La práctica "parity" debía calcular la suma de paridades impar (XOR de todos los bits) de los elementos de un array. Un estudiante entrega la siguiente versión de parity5:

```

int parity5(unsigned* array, int len){
    int i,j,res=0;
    unsigned x;
    for (i=0; i<len; i++){
        x=array[i];
        for (j=sizeof(unsigned)*4;
             j>0; j=j/2){
            x^=x>>j;
        }
    }
    return res;
}

```

```

    }

    x = x & 0x1;

    res+=x;

}

return res;
}

```

Esta función parity5:

- > a) produce siempre el resultado correcto
- solemos escribir res+=x&0x1, en lugar de ponerlo en 2 sentencias C
- b) fallaría con array={0,1,2,3}
- c) fallaría con array={1,2,4,8}
- d) no siempre produce el resultado correcto, pero el error no se manifiesta en los ejemplos propuestos, o se manifiesta en ambos

22. Para averiguar la paridad de un número se puede usar la operación:

- a) NOT
- b) OR
- c) AND
- > d) XOR

23. Utilizando la sentencia asm(), las denominadas restricciones que se indican al final de dicha sentencia, involucran a:

- a) Solamente las entradas
- b) Solamente las salidas
- c) Solamente los sobrescritos
- > d) Ninguna de las anteriores es cierta

24. La práctica "parity" debía calcular la suma de paridades impar (XOR de todos los bits) de los elementos de un array. Un estudiante entrega la siguiente versión de parity6:

```

int parity6(unsigned * array, int len)

{
    int i, result = 0;
    unsigned x;
    for (i=0; i<len; i++){

```

```

x = array[i];

asm( "mov %[x], %%edx \n\t"
    "shr $16, %%edx \n\t"
    "shr $8, %%edx \n\t"
    "xor %%edx,%%edx \n\t"
    "setp %%dl \n\t"
    "movzx %%dl,%[x] \n\t"
    : [x] "+r" (x)
    :
    : "edx"
);

result += x;
}

return result;
}

```

Esta función parity6:

a) produce el resultado correcto

-> b) no es correcta; fallaría por ejemplo con array={0,1,2,3}

Caso real, entregado en prácticas. Las tres primeras instrucciones asm se pierden al poner edx a 0 usando xor. Consecuentemente, se activa PF para ajustar a impar, y termina siendo x=1. Es decir, todos los elementos del array contabilizan paridad=1. El array {1,2,4,8} pasa desapercibido, pero {0,1,2,3} debería producir resultado=2<>4.

c) no es correcta; fallaría por ejemplo con array={1,2,4,8}

d) no es correcta pero el error no se manifiesta en los ejemplos propuestos, o se manifiesta en ambos

25. En la práctica “popcount/paridad”, para cronometrar sistemáticamente las diversas versiones necesitamos una función crono() a la que se le pueda pasar como argumento cuál versión queremos cronometrar. En lenguaje C esto se puede hacer con punteros a funciones.

Sabiendo que todas las versiones devuelven un valor entero, el prototipo de la función crono() debería ser:

a) void crono( int \* func , char\* msg);

no compila, 1er argumento es puntero a entero

-> b) void crono( int (\* func)(), char\* msg);

ok, 1er argumento es el puntero a función devolviendo entero

c) void crono( int \* func (), char\* msg);

funciona pero 2warnings & 1note: tipos incompatibles int!=int\*

d) void crono((int \*)func (), char\* msg);

no compila, sobran paréntesis, sería prototip.func.devolv.ptr.int

26. La práctica "popcount" debía calcular la suma de bits (peso Hamming) de los elementos de un array. Un estudiante entrega la siguiente versión de popcount3:

```
int popcount3(int* array, int len){
```

```
    long val = 0;
```

```
    int i;
```

```
    unsigned x;
```

```
    for (i=0; i<len; i++){
```

```
        x= array[i];
```

```
        do{
```

```
            val += x & 0x1;
```

```
            x >>= 1;
```

```
        } while (x);
```

```
        val += (val >> 16);
```

```
        val += (val >> 8);
```

```
    }
```

```
    return val & 0xFF;
```

```
}
```

Esta función es una mezcla inexplicada de las versiones "oficiales" de popcount2 y popcount4, incluyendo diferencias en 2 tipos de datos, la ausencia de la variable res y la diferente posición de la máscara 0xFF.

Esta función popcount3:

a) produce siempre el resultado correcto

No. El ejemplo grande falla (NBITS=20), igual que fallarían otros ejemplos grandes más pequeños (desde NBITS=7 sale que  $7 \cdot 2^6 = 448 > 255$ ).

Fallaría con cualquier ejemplo en que el resultado fuera superior a 0xFF = 255, porque entonces val>>8 no sería cero y se sumaría erróneamente.

b) fallaría con array={0,1,2,3}

No, 4<<255, siempre val>>8 == 0, no se suma nada erróneamente

c) fallaría con array={1,2,4,8}

No, 4<<255, siempre val>>8 == 0, no se suma nada erróneamente

-> d) no es correcta pero el error no se manifiesta en los ejemplos propuestos, o se manifiesta en ambos

Efectivamente.

El ejemplo más pequeño para llegar al error (resultado>=256) sería

```
lista[] = {0xffffffff, 0xffffffff, 0xffffffff, 0xffffffff,  
          0xffffffff, 0xffffffff, 0xffffffff, 0xffffffff}
```

En ese caso, calcula val=256, le suma 1 (val=257) y lo enmascara: val=1

El array no es int, val no es long, queríamos haber sumado parcialmente en res el popcount de cada elemento... hasta el while(x) iba bien, sería similar a popcorn2 (popcount1 era con for, popcorn2 con while). Sobran los desplazamientos val+=val>>const y la máscara val&0xFF.

En Julio 2017 Problema 3 se comenta que esta función produce el resultado correcto para los tres ejemplos pequeños, y falla con el ejemplo grande ( $237 \ll 20 \cdot 2^{19} = 10M = 10\ 485\ 760$ ), sugiriendo que produce el resultado correcto siempre que queden sin efecto las "partes erróneas" del código: val>>16, val>>8 y val&0xFF

27. Suponga la siguiente sentencia asm en un programa:

```
asm(" add (%[a],%[i],4),%[r]"  
    :[r] "+r" (result)  
    :[i] "r" (i),  
    [a] "r" (array)  
);
```

¿Cuál de las siguientes afirmaciones es incorrecta?

- a) se desea que el valor calculado por la instrucción ensamblador quede almacenado en la variable C result  
correcto, add ..., %[r] almacena en result  
-> b) i es un registro de salida

no, `asm("sentence":output[input:clobber], [i] va en input`

c) r es un registro de entrada/salida

ok, "+r"

d) a es un registro de entrada

ok, "r"

28. ¿Cuál es el `popcount` (peso Hamming, nº de bits activados) del número 29?

a) 2

b) 3

-> c) 4

$29 = 0x1D = 16+8+4+1 = 0b0001\ 1101 \rightarrow \text{popcount } 4$

d) 5

29. En la convención `cdecl` estándar para arquitecturas x86 de 32 bits, cuál de las siguientes afirmaciones es cierta:

-> a) Los parámetros se pasan en pila, de derecha a izquierda; es decir, primero se pasa el último parámetro, después el penúltimo... y por fin el primero

b) Solamente es necesario salvar el registro EAX

c) Los registros EBX, ESI y EDI son salva-invocante

d) Ninguna de las anteriores es cierta

30. La práctica "parity" debía calcular la suma de paridades impar (XOR de todos los bits) de los elementos de un array. Un estudiante entrega la siguiente versión de parity4:

```
int parity4(unsigned* array, int len){
```

```
    int val,i,res=0;
```

```
    unsigned x;
```

```
    for (i=0; i<len; i++){
```

```
        x=array[i];
```

```
        val=0;
```

```
        asm("\n"
```

```
"ini3: \n\t"
```

```
"xor %[x],%[v] \n\t"
```

```
"shr %[x] \n\t"
```

```
"test %[x], %[x]\n\t"
```

```

"jne ini3 \n\t"
:[v]"+r" (val)
:[x] "r" (x)
);
val = val & 0x1;
res+=val;
}
return res;
}

```

Esta función parity4:

- > a) produce siempre el resultado correcto
- solemos escribir res+=val&0x1, en lugar de ponerlo en 2 sentencias C
- adicionalmente, shr afecta al flag ZF de manera que sobra test %[x],%[x]
- b) fallaría con array={0,1,2,3}
- c) fallaría con array={1,2,4,8}
- d) no siempre produce el resultado correcto, pero el error no se manifiesta en los ejemplos propuestos, o se manifiesta en ambos

31. ¿Cuál es el **popcount** (peso Hamming, nº de bits activados) del número 19?

a) 2

-> b) 3

$19 = 0x13 = 16 + 3 = 0b0001\ 0011 \rightarrow \text{popcount } 3$

c) 4

d) 5

32. La práctica "parity" debía calcular la suma de paridades impar (XOR de todos los bits) de los elementos de un array. Un estudiante entrega la siguiente versión de parity6:

```
int parity6(unsigned* array, int len){
```

```
    int i,j,res=0;
```

```
    unsigned x;
```

```
    for (i=0; i<len; i++){
```

```
        x=array[i];
```

```
        asm("\n"
```

```

"mov %[x],%%edx \n\t"
"shr $16, %%edx \n\t"
"xor %%edx,%[x] \n\t"
"mov %[x],%%edx \n\t"
"mov %%dh, %%dl \n\t"
"xor %%edx, %[x]\n\t"
"setpo %%cl \n\t"
"movzx %%cl, %[x]"
:[x] "+r" (x)
:
:"edx","ecx"
);
res+=x;
}
return res;
}

```

La sentencia `asm()` del listado anterior tiene las siguientes restricciones

- a) ninguna
  - b) arquitectura de 32 bits
  - c) dos entradas y una salida
- > d) un registro y dos sobrescritos (clobber) // el registro es `[x] "+r"` y sobrescritos son `"edx","ecx"`

33. En 80x86, los parámetros a las subrutinas se pueden pasar:

- a) a través de variables globales
  - b) a través de los registros
  - c) a través de la pila
- > d) todas las anteriores son ciertas

34. ¿En qué registro se pasa el primer argumento a una función según el estándar `_cdecl` en una arquitectura IA32?

- a) edi
- b) esi

c) eax

-> d) Las anteriores respuestas son erróneas

35. La práctica "parity" debía calcular la suma de paridades impar (XOR de todos los bits) de los elementos de un array. Un estudiante entrega la siguiente versión de parity4:

```
int parity4(unsigned* array, int len){  
    int i;  
    unsigned x;  
    int val=0, result=0;  
    for (i=0; i<len; i++){  
        x = array[i];  
        asm("\n"  
            "ini: \n\t"  
            "xor %[x], %[v] \n\t"  
            "shr %[x] \n\t"  
            "jnz ini \n\t"  
            : [v]" +r" (val)  
            : [x] "r" (x)  
            );  
        result += val & 0x1;  
    }  
    return result;  
}
```

36. Esta función contiene un único error (en realidad se han editado 2 líneas de código sobre la versión correcta) pero produce resultado correcto cuando se usa como test el array...

a) array={1, 16, 256, 1024}

debería salir 4 (todos impares)

sale 2 (1,0,1,0, cada otro impar deshace el anterior)

-> b) array={5, 4, 3, 2}

debería salir 2 (impares el 4 y el 2)

sale 2 (0,1,1,0, el 3 cuenta como impar, el 2 deshace el impar)

c) array={1, 2, 4, 8}

debería salir 4 (1,1,1,1, todos impares)

sale 2 (1,0,1,0, cada otro impar deshace el anterior)

d) array={0, 1, 2, 3}

debería salir 2 (0,1,1,0, impares el 1 y el 2)

sale 1 (0,1,0,0, el 2 deshace el impar del 1)

El error consiste en que val=0 se inicializa al principio, en lugar de tras cada  $x=array[i]$ , de manera que si el LSB de val se queda activado la siguiente paridad se calcula mal. En concreto, tras calcular un impar, todos los pares que sigan cuentan como impar hasta que llegue un impar que deshaga el impar (y entonces el nuevo impar cuenta como par).

37. Tras iniciar una sesión de depuración del Ejercicio 1 de la Práctica 3 con el comando gdb -tui --args suma\_01\_S\_SysV uno dos tres, arrancamos el programa con br \_start y run. Si tecleamos entonces `p *(char**)(\$rsp+40)`, ¿qué obtenemos?

a. tres sería \$rsp+32

-> b. 0x0 Práct.3, pág.4, Fig.3, NULL

c. una variable de entorno sería \$rsp+48, 56, 64...

d. un error de gdb

37. ¿Cuál es el `popcount` (peso Hamming, nº de bits activados) del número 42?

a. 2

-> b.  $3 // 42 = 0x2A = 0b10\ 1010 \Rightarrow 3$  bits ( $32+8+2$ )

c. 4

d. 5

## Sesión 4

1. En la práctica de la bomba, el segundo ejercicio consistía en crear un ejecutable sin "explosiones", para lo cual

se puede utilizar... (marcar la opción \*falsa\*)

- > a) objdump
- b) ddd
- c) hexedit
- d) gdb

2. Una de las "bombas" utiliza el siguiente bucle para cifrar la cadena con la contraseña introducida por el usuario:

80485bb: rolb \$0x4,(%eax)

80485be: add \$0x1,%eax

80485c1: cmp %edx,%eax

80485c3: jne 80485bb <encrypt+0x20>

La intrucción rolb rota el byte destino hacia la izquierda tantos bits como indica el operando fuente. Si

inicialmente eax apunta a la cadena del usuario, que se compara con otra cadena

"\x16\x26\x27\x16\x36\x16\x46\x16\x26\x27\x16", almacenada en el código, la contraseña es:

- > a) "\x61\x62\x72\x61\x63\x61\x64\x61\x62\x72\x61" ("abracadabra")
- b) "\x63\x61\x64\x61\x62\x72\x61\x61\x62\x72\x61" ("cadabraabra")
- c) "\x61\x62\x72\x61\x61\x62\x72\x61\x63\x61\x64" ("abraabracad")
- d) "\x61\x72\x62\x61\x64\x61\x63\x61\x72\x62\x61" ("arbadacarba")

3. En una bomba como las estudiadas en prácticas, del tipo...

0x0804873f <main+207>: call 0x8048504 <scanf>

0x08048744 <main+212>: mov 0x24(%esp),%edx

0x08048748 <main+216>: mov 0x804a044,%eax

0x0804874d <main+221>: cmp %eax,%edx

0x0804874f <main+223>: je 0x8048756 <main+230>

0x08048751 <main+225>: call 0x8048604 <boom>

0x08048756 <main+230>: ...

la contraseña es...

a) el entero 0x804a044

-> b) el entero almacenado a partir de la posición de memoria 0x804a044

c) el string almacenado a partir de la posición de memoria 0x24(%esp)

d) ninguna de las anteriores

4. En la práctica de la bomba necesitamos estudiar el código máquina de la bomba del compañero. A veces dicho

código no se visualiza directamente en el depurador ddd, y algunas de las técnicas que se pueden probar para

conseguir visualizarlo son... (marcar la opción \*falsa\*)

a) comprobar que está activado el panel View → Machine Code Window

-> b) recompilar con información de depuración, por si se nos había olvidado, ya que sin -g el ejecutable no contiene

información de depuración

c) asegurarse de que se ha escrito correctamente el nombre del ejecutable

d) escribir info line main en el panel de línea de comandos gdb

5. En la práctica de la bomba, el tercer ejercicio consistía en usar un editor hexadecimal para crear un ejecutable

sin “explosiones”. Para saber qué contenidos del fichero hay que modificar, se puede utilizar... (marcar la opción

\*falsa\*)

a) objdump

b) ddd

-> c) hexedit

d) gdb

6. En una bomba como las estudiadas en prácticas, del tipo...

0x0804873f <main+207>: call 0x8048504 <scanf>

0x08048744 <main+212>: mov 0x24(%esp),%edx

0x08048748 <main+216>: mov 0x804a044,%eax

0x0804874d <main+221>: cmp %eax,%edx

0x0804874f <main+223>: je 0x8048756 <main+230>

0x08048751 <main+225>: call 0x8048604 <boom>

0x08048756 <main+230>:

...el código numérico (pin) es...

- > a) el entero almacenado a partir de la posición de memoria 0x804a044
- b) el entero 0x804a044
- c) el entero cuya dirección está almacenada en la posición de memoria 0x804a044
- d) el entero almacenado a partir de la posición de memoria 0x24(%esp)

7. Respecto a las bombas estudiadas en la práctica "bomba digital", ¿en cuál de los siguientes tipos de bomba

sería más difícil descubrir la contraseña? Se distingue entre enteros definidos en el código fuente de la bomba, y

enteros solicitados al usuario mediante scanf(). Por "procesar" se entiende calcular el n-ésimo elemento de la

serie de Fibonacci.

- > a) 1 entero del usuario se procesa, y se compara con el entero del fuente
- b) 2 enteros del fuente se suman, se procesa la suma, y se compara el resultado con el entero del usuario
- c) 2 enteros del fuente se procesan, se suman los resultados, y se compara la suma con el entero del usuario
- d) Las opciones más difíciles son de la misma dificultad, así que no se puede marcar ninguna como la más difícil

8. Respecto a las bombas estudiadas en la práctica "bomba digital", ¿en cuál de los siguientes tipos de bomba

sería más difícil descubrir la(s) contraseña(s)? Se distingue entre enteros definidos en el código fuente de la

bomba, y enteros solicitados al usuario mediante scanf(). Por "procesar" podemos entender calcular el n-ésimo

elemento de la serie de Fibonacci, por ejemplo.

- a) 1 entero del fuente se procesa, y se compara con el entero del usuario
- b) 2 enteros del usuario se suman, se procesa la suma, y se compara con el entero del fuente
- > c) 2 enteros del usuario se procesan, se suman los resultados, y se compara con el entero del fuente
- d) Las 2 (o 3) opciones más difíciles son de la misma dificultad, así que no se puede marcar ninguna como la más difícil

9. Respecto a las bombas estudiadas en la práctica "bomba digital", ¿en cuál de los siguientes tipos de bomba

sería más difícil descubrir la contraseña? Se distingue entre strings definidos en el código fuente de la bomba, y

strings solicitados al usuario mediante `scanf()`. Por "cifrar" podemos entender la cifra del César, por ejemplo.

"Invertir" es darle la vuelta al string de manera que la primera letra se convierta en la última y viceversa.

-> a) 1 string del usuario se cifra, y se compara con el string del fuente

b) 2 strings del fuente se invierten, se concatenan, se cifra el resultado, y se compara con el string del usuario

c) 2 strings del fuente se concatenan, se invierte el resultado, se cifra, y se compara con el string del usuario

d) Las 2 (o 3) opciones más difíciles son de la misma dificultad, así que no se puede marcar ninguna como la más

difícil

10. En la práctica de la bomba, el primer ejercicio consistía en "saltarse" las "explosiones", para lo cual se puede

utilizar...

a) objdump o gdb

b) ddd o hexedit

c) hexedit u objdump

-> d) gdb o ddd

un depurador cualquiera sirve para forzar que se cumpla la condición para "saltarse" la bomba

12. Una de las "bombas" utiliza el siguiente código para cifrar la clave numérica introducida por el usuario y ahora

almacenada en eax:

804870d: xor \$0xffff,%eax

8048712: mov \$0x2,%ecx

8048717: cltd

8048718: idiv %ecx

804871a: cmp %eax,0x804a034

Si el entero almacenado a partir de 0x804a034 es 0x7ff, la clave numérica puede ser:

a) 0x1009 4F97 (269 045 655)

b) 0xffff (4095)

c) 0x7ff (2047)

-> d) 1

11. Para corregir la práctica “bomba digital”, un profesor dispone de 26 ejecutables (y la lista de claves

correspondientes) para asignar en el Laboratorio una bomba distinta a cada estudiante.

Cuando un estudiante

diga que la ha resuelto el profesor exigirá ver al estudiante ejecutando la bomba y tecleando la contraseña y el pin

correctos para comprobar que no explota, para así anotarla como resuelta y que le puntúe al estudiante.

Un estudiante dice que ha resuelto su bomba, y cuando el profesor pide que se tecleen las claves, la contraseña

coincide con la que tiene anotada el profesor, pero el pin no, y de todas formas la bomba no explota. Debería

hacerse lo siguiente:

a) la bomba tiene que puntuarle al estudiante porque no ha explotado, y el profesor estaría quedando en ridículo

(por injusto) si intenta entorpecer o dificultar de cualquier forma que el estudiante se lleve la merecida nota

-> b) el profesor puede pedirle que vuelva a descargar la bomba original e intente repetir con ese ejecutable las

claves que acaba de teclear: en realidad eso es sólo para asegurarse completamente de que no se trata de una

errata en la lista, aunque también es por la curiosidad de ver si el estudiante se da cuenta de que el profesor sabe lo

que ha hecho

c) hay que denunciar al profesor a los representantes de estudiantes

d) el enunciado es capcioso: si la lista no tiene erratas y el pin se escribe mal la bomba tiene que explotar, y si la

bomba no explota es porque las claves se escribieron bien y la lista está mal. En el primer caso no debe puntuar, en

el segundo caso sí debe puntuar, y en ningún caso el profesor tiene derecho a hacer comprobaciones distintas a las

dos mencionadas

13. ¿Para qué se utiliza la función gettimeofday en la práctica de la "bomba digital"?

a) Para cronometrar y poder comparar las duraciones de las distintas soluciones del programa

b) Para cifrar la contraseña en función de la hora actual

-> c) Para lanzar un error cuando el usuario tarde demasiado tiempo en introducir la contraseña o el PIN

d) Para imprimir la hora en la pantalla

14. En una bomba como las estudiadas en prácticas, del tipo...

0x080486e8 <main+120>: call 0x8048524 <strncpy>

0x080486ed <main+125>: test %eax,%eax

0x080486ef <main+127>: je 0x80486f6 <main+134>

0x080486f1 <main+129>: call 0x8048604 <boom>

0x080486f6 <main+134>: ...

la contraseña es...

a) el valor que tenga %eax

b) el string almacenado a partir de donde apunta %eax

c) el entero almacenado a partir de donde apunta %eax

-> d) ninguna de las anteriores

15. Para corregir la práctica “bomba digital”, un profesor dispone de 26 ejecutables (y la lista de claves

correspondientes) para asignar en el Laboratorio una bomba distinta a cada estudiante.

Cuando un estudiante

diga que la ha resuelto el profesor exigirá ver al estudiante ejecutando la bomba y tecleando la contraseña y el pin

correctos para comprobar que no explota, para así anotarla como resuelta y que le puntúe al estudiante.

Un estudiante (usando ordenador del Laboratorio con Ubuntu 10.04) dice que ha resuelto su bomba, y cuando el

profesor pide que se tecleen las claves, el ddd se “bloquea” y le empieza a “parpadear” al estudiante. Para poder

puntuar, el estudiante debería...

a) teclear las claves (contraseña y pin), aunque ddd esté “bloqueado” y “parpadeando”

b) probar en orden los remedios básicos: pulsar <Ctrl>-C varias veces, pulsar <Enter> repetidamente, comprobar

que está seleccionada “Machine Code Window”, teclear “info line main”, y si todo falla, ejecutar “rm -rf ~/.ddd”

-> c) matar la ventana ddd, abrir un terminal con un shell, ejecutar la bomba desde línea de comandos y teclear las

claves

d) reinstalar un paquete ddd más actualizado usando “sudo apt-get install”

16. En una bomba como las estudiadas en prácticas, del tipo...

0x08048705 <main+149>: call 0x80484c4 <gettimeofday>

...

0x08048718 <main+168>: cmp \$0x5,%eax

0x0804871b <main+171>: jle 0x8048722 <main+178>

0x0804871d <main+173>: call 0x8048604 <boom>

0x08048722 <main+178>: ...

ejecutada paso a paso con el depurador ddd, interesaría...

a) ejecutar hasta jle, ajustar %eax a 6, y continuar ejecutando paso a paso

b) ejecutar hasta jle, ajustar %eax a 4, y continuar ejecutando paso a paso

-> c) cambiar jle por jmp usando ddd o un editor hex, salvar el programa y reiniciar la depuración con el nuevo

ejecutable

d) Ninguna de las opciones anteriores es de interés

17. En una bomba como las estudiadas en prácticas, del tipo...

0x080486e8 <main+120>: call 0x8048524 <strncmp>

0x080486ed <main+125>: test %eax,%eax

0x080486ef <main+127>: je 0x80486f6 <main+134>

0x080486f1 <main+129>: call 0x8048604 <boom>

0x080486f6 <main+134>: ...

la contraseña es...

a) el string almacenado a partir de 0x8048524

b) el string almacenado a partir de 0x80486f6

c) el string almacenado a partir de 0x8048604

-> d) ninguna de las anteriores

18. La función gettimeofday() en la práctica de la "bomba digital" se utiliza para:

a) Para comparar las duraciones de las distintas soluciones del programa

b) Para imprimir la fecha y hora

c) Para cifrar la contraseña en función de la hora actual

-> d) Para cronometrar lo que tarda el usuario en introducir la contraseña

19. Un fragmento de una "bomba" desensamblada es:

```
0x0804873f: call 0x8048504 <scanf>
0x08048744: mov 0x24(%esp),%edx
0x08048748: mov 0x804a044,%eax
0x0804874d: cmp %eax,%edx
0x0804874f: je 0x8048756 <main+230>
0x08048751: call 0x8048604 <boom>
0x08048756: ...
```

La contraseña/clave en este caso es...

- a) el string almacenado a partir de la posición de memoria 0x804a044
- b) el string almacenado a partir de la posición de memoria 0x24(%esp)
- > c) el entero almacenado a partir de la posición de memoria 0x804a044
- d) el entero 0x804a044

20. En la realización de la práctica de la bomba digital, una parte del código máquina es el siguiente:

```
0x080486e8 <main+120>: call 0x8048524 <strncmp>
0x080486ed <main+125>: test %eax,%eax
0x080486ef <main+127>: je 0x80486f6 <main+134>
0x080486f1 <main+129>: call 0x8048604 <boom>
```

¿Cuál de los siguientes comandos cambiaría el salto condicional por un salto incondicional?

- a) set %0x080486ef=0xeb
- > b) set \*(char\*)0x080486ef=0xeb
- c) set \$0x080486ef=0xeb
- d) set \*(char\*)0x080486f6=jmp

21. Respecto a las bombas estudiadas en la práctica "bomba digital", ¿en cuál de los siguientes tipos de bomba

sería más fácil descubrir la(s) contraseña(s)? Se distingue entre strings definidos en el código fuente de la bomba,

y strings solicitados al usuario mediante scanf(). Por "cifrar" se entiende aplicar la cifra del César (sumar o restar

una constante fija a los códigos ASCII).

- > a) 1 string del fuente se cifra, y se compara con el string del usuario

- b) 2 strings del usuario se concatenan, se cifra el resultado y se compara con el string del fuente
- c) 2 strings del usuario se cifran, se concatenan los resultados, y se compara con el string del fuente
- d) Las opciones más fáciles son de la misma dificultad, así que no se puede marcar ninguna como la más fácil

22. En una bomba como las estudiadas en prácticas, del tipo...

0x080486e8 <main+120>: call 0x8048524 <strncmp>

0x080486ed <main+125>: test %eax,%eax

0x080486ef <main+127>: je 0x80486f6 <main+134>

0x080486f1 <main+129>: call 0x8048604 <boom>

0x080486f6 <main+134>:

...la contraseña es...

- a) el valor que tenga %eax
- b) el string almacenado a partir de 0x80486f6
- c) el entero almacenado a partir de donde apunta %eax
- > d) ninguna de las anteriores

23. ¿Para qué se utiliza la función scanf() en la práctica de la "bomba digital"?

- a. Para escanear el fichero ejecutable “bomba” y asegurarse de que no contenga virus
- b. Para leer la contraseña tecleada por el usuario
- > c. Para leer el PIN o código numérico //P4 pág.3 Fig.1
- d. Para lanzar un error cuando el usuario tarde demasiado tiempo en introducir la contraseña o el PIN

24. En una bomba como las estudiadas en prácticas, del tipo...

0x40079b <main+64> lea 0x30(%rsp),%rdi

0x4007a0 <main+69> mov 0x2008d9(%rip),%rdx # 0x601080

0x4007a7 <main+76> mov \$0x64,%esi

0x4007ac <main+81> callq 0x400600 <fgets@plt>

0x4007b1 <main+86> test %rax,%rax

0x4007b4 <main+89> je 0x400785 <main+42>

0x4007b6 <main+91> lea 0x30(%rsp),%rdi

0x4007bb <main+96> mov \$0xd,%edx

```
0x4007c0 <main+101> lea 0x2008a1(%rip),%rsi # 0x601068
0x4007c7 <main+108> callq 0x4005d0 <strncmp@plt>
0x4007cc <main+113> test %eax,%eax
0x4007ce <main+115> je 0x4007d5 <main+122>
0x4007d0 <main+117> callq 0x400727 <boom>
0x4007d5 <main+122> lea 0x20(%rsp),%rdi
...la contraseña (alfanumérica) es...
a. el string almacenado a partir de 0x601068
b. el string almacenado a partir de 0x2008a1+0x4007c7
c. el string almacenado a partir de 0x30(%rsp)
-> d. no se puede marcar una y sólo una de las respuestas anteriores
25. En una bomba como las estudiadas en prácticas, del tipo...
0x40080f <main+180> lea 0xc(%rsp),%rsi
0x400814 <main+185> lea 0x1dd(%rip),%rdi # 0x4009f8
0x40081b <main+192> mov $0x0,%eax
0x400820 <main+197> callq 0x400620 <__isoc99_scanf@plt>
0x400825 <main+202> mov %eax,%ebx
0x400827 <main+204> test %eax,%eax
0x400829 <main+206> jne 0x40083c <main+225>
0x40082b <main+208> lea 0x1c9(%rip),%rdi # 0x4009fb
0x400832 <main+215> mov $0x0,%eax
0x400837 <main+220> callq 0x400620 <__isoc99_scanf@plt>
0x40083c <main+225> cmp $0x1,%ebx
0x40083f <main+228> jne 0x4007f9 <main+158>
0x400841 <main+230> mov 0x200819(%rip),%eax # 0x601060
0x400847 <main+236> cmp %eax,0xc(%rsp)
0x40084b <main+240> je 0x400852 <main+247>
0x40084d <main+242> callq 0x400727 <boom>
0x400852 <main+247> lea 0x10(%rsp),%rdi
...el código numérico (pin) es...
a. el entero 0x601060
```

- b. el entero cuya dirección está almacenada en la posición de memoria 0x4009f8
- c. el entero almacenado a partir de la posición de memoria 0x4009fb
- > d. el entero almacenado a partir de la posición de memoria 0x200819+0x400847

## Sesión 5

1. En la práctica de E/S en Arduino, DDRB es

a) el segundo canal de memoria DDR

- b) el registro de dirección de datos del puerto B

el guión dice "Cada uno de los 8 bits del registro Port B Data Direction Register (DDRB) indica si el bit correspondiente del puerto B está configurado como entrada (0) o salida (1)."

c) el registro de datos y direcciones B

d) el registro buffer de datos D

2. En la práctica "blink" de Arduino, un estudiante muestra a los profesores lo que le han enseñado en algún otro lugar: conectando un led directamente entre las patillas Digital pin 13 (LED\_BUILTIN) y GND (justo al lado) también parpadea. Cabe esperar la siguiente respuesta de los profesores:

a) Exactamente esa solución viene en el guión, no hace falta aprenderla fuera de clase

- b) Nosotros conectamos los led en serie con una resistencia como dice el guión

c) Hay que revisar ese equipo en concreto, debe tener algún defecto, ese led no debe parpadear

d) Es cierto, es un error de diseño del guión, es más sencillo conectarlo así

3. El circuito que hay que montar en la práctica del zumbador pasivo con Arduino es un:

a) Buffer triestado

- b) Divisor de tensión

c) Seguidor de voltaje

d) Comparador

4. ¿Cuál de las siguientes patillas (pin) no se menciona (no se pide usar/no se debe usar) en la 1<sup>a</sup> práctica del Theremín de luz de Arduino? (Proyecto p06 / sólo fotocélula y zumbador)

a) Analog In A0

- b) AREF

c) ledPin = 13

d) GND

5. En la práctica de E/S en Arduino, la instrucción SBI del repertorio del microcontrolador realiza...

a) una suma de un valor inmediato

b) una resta de un valor inmediato

- c) una operación de bits

el guión dice "SBI P,b (Set Bit in I/O Register) pone a 1 el bit b del registro P [1]"

- d) un salto incondicional

6. En la placa del kit de Arduino, las patillas de tierra vienen etiquetadas con la leyenda:

- a) A0
- b) 5V
- c) GND

- d) 3.3V

7. Sobre el resultado devuelto por la función de Arduino map(sensorValue, sensorLow, sensorHigh, 50, 4000); usada en el programa del Theremín de luz:

- a) Si sensorValue vale 50, devuelve sensorLow
- b) Si sensorValue vale 50, devuelve sensorHigh
- c) Si sensorValue vale 2025, devuelve la mitad entre sensorLow y sensorHigh
- d) Si sensorValue vale la mitad entre sensorLow y sensorHigh, devuelve 2025  
sí, map traslada rango 2º-3er args al rango 4º-5º args

8. ¿Qué sentencia usamos en el programa blink (led intermitente) para encender el led integrado en la placa Elegoo Mega2560?

- a) analogWrite (LED\_BUILTIN, HIGH);
- b) pulseIn (LED\_BUILTIN, HIGH);
- c) pinMode (LED\_BUILTIN, OUTPUT);
- d) digitalWrite (LED\_BUILTIN, HIGH);

9. En la práctica de E/S en Arduino, la instrucción SBIW del repertorio del microcontrolador realiza...

- a) una suma de un valor inmediato
- b) una resta de un valor inmediato  
el guión dice "La instrucción SBIW RdI,K (Subtract Immediate from Word) resta la constante K del registro de 16 bits Rdh:Rdl, es decir, hace Rdh:Rdl  $\leftarrow$  Rdh:Rdl - K."
- c) una operación de bits
- d) un salto incondicional

10. Al colocar un led en la placa de prototipado de Arduino, ¿cómo se sabe cuáles son el ánodo y el cátodo?

- a) Mirando las leyendas A+ o K- impresas en las patillas del led
- b) No es necesario saberlo, la polaridad es indiferente a la hora de colocar un led en un circuito

c) Mirando los códigos de color dibujados sobre el encapsulado

- d) Mirando la longitud de las patillas o qué lado del encapsulado es plano

11. En la práctica de E/S en Arduino, la instrucción CBI del repertorio del microcontrolador realiza...

a) una comparación

b) un complemento a uno

- c) una operación de bits

el guión dice "hay que usar la instrucción máquina CBI P,b (Clear Bit in I/O Register), que pone a 0 el bit b del registro P [1]:"

d) una llamada a subrutina

12. Las resistencias utilizadas en la práctica de Arduino

- a) Son de color azul claro y tienen 5 bandas de color: las 3 primeras indican un valor, la 4<sup>a</sup> banda es un multiplicador y la 5<sup>a</sup> banda es la tolerancia
- b) Son de color beige y tienen 4 bandas de color: las 2 primeras indican un valor, la 3<sup>a</sup> banda es un multiplicador y la 4<sup>a</sup> banda es la tolerancia
- c) Tienen polaridad y el cátodo (polo negativo) es el extremo de la banda de color con una separación mayor respecto a las otras.
- d) Tienen polaridad y el ánodo (polo positivo) es el extremo de la banda de color con una separación mayor respecto a las otras.

13. La función setup() de Arduino es llamada:

- a) Al principio de cada iteración de la función loop()
- b) Cuando se sube desde el entorno de desarrollo o se pulsa el botón de reset, pero no cuando se conecta la alimentación
- c) Cuando se conecta la alimentación a la placa, se pulsa el botón de reset, pero no cuando se sube desde el entorno de desarrollo
- d) Cuando se conecta la alimentación a la placa, se pulsa el botón de reset, o se sube desde el entorno de desarrollo

## Sesión 6

1. En la práctica de la cache, el código de “size.cc” accede al vector saltando de 64 en 64.

¿Por qué?

a) Para recorrer el vector más rápidamente

- b) Porque con un salto menor que 64 habría aciertos por localidad espacial y haría menos clara la gráfica

sí, intentamos meter lo más rápidamente posible el vector en cache, para lo cual tocamos sólo un byte de cada línea.

c) Porque cada elemento del vector ocupa 64 B

d) Para evitar aciertos por localidad temporal y que sólo haya aciertos por localidad espacial no, al revés, sólo localidad temporal

2. ¿Cuál de las siguientes afirmaciones sobre las caches es \*FALSA\*?

a) La cache de nivel 3 no contiene toda la memoria que maneja el programa

b) Las direcciones a las que accede un programa no son completamente aleatorias, sino que se rigen por ciertos patrones de localidad

c) Un procesador actual tiene varias caches de nivel 1

-> d) Casi ningún procesador actual tiene memoria cache L2

3. ¿Cuál de las siguientes afirmaciones sobre el programa size.cc de la práctica 5 es cierta?

a) La diferencia de velocidades entre L2 y L3 es mayor que la diferencia de velocidades entre L1 y L2.

puede que haya sistemas con esa característica, pero desde luego los que usamos nosotros en prácticas no.

b) Si continuáramos multiplicando por 2 el tamaño del vector en el eje X obteniendo más puntos de la gráfica, esta continuaría horizontal para cualquier valor más allá de 64 MB.

-> c) La gráfica tiene escalones hacia arriba porque en cada punto del eje X accedemos al mismo número de elementos del vector y el número de aciertos por localidad temporal disminuye bruscamente en ciertos puntos al aumentar el tamaño del vector.

si cabe el vector en cache, los millones de accesos (siempre el mismo número de accesos) son todo aciertos por localidad temporal

d) La gráfica tiene tramos horizontales porque el hecho de realizar la mitad de accesos al vector en cada punto de un tramo horizontal respecto al anterior punto de ese mismo tramo

horizontal es compensado por el número de fallos creciente en ese mismo tramo horizontal.

4. El servidor de SWAD tiene dos procesadores Xeon E5540 con 4 núcleos cada uno. Cada procesador tiene 4 caches L1 de instrucciones de 32 KB, 4 caches L1 de datos de 32 KB, 4 caches unificadas L2 de 256 KB y una cache unificada L3 de 8MB. Suponga que un proceso swad, que se ejecuta en un núcleo, tiene que ordenar un vector de estudiantes accediendo repetidamente a sus elementos. Cada elemento es una estructura de datos de un estudiante y tiene un tamaño de 4KB. Si representamos en una gráfica las prestaciones en función del número de estudiantes a ordenar, ¿para qué límites teóricos en el número de estudiantes se observarán saltos en las prestaciones debidos a accesos a la jerarquía de memoria?

- a) 32 / 256 / 8192 estudiantes
- b) 16 / 32 / 64 estudiantes
- > c) 8 / 64 / 2048 estudiantes
- d) 4 / 32 / 512 estudiantes

5. En el programa "size" de la práctica de la cache, si el primer escalón pasa de tiempo = 1 para todos los tamaños de vector menores o iguales que 32 KB a tiempo = 3 para los tamaños 64 KB y 128 KB, podemos asegurar que:

(Nota: en las opciones de respuesta, "rapidez" se refiere a Ti, el tiempo de acceso efectivo al nivel i, no a ti, el tiempo de acceso propio al nivel i. Si dice "La cache L1 es 3x más rápida que L2", se refiere a que  $T1=T2/3$ , no a que  $t1=t2/3$ . Ver T6 tr.26)

- 
- a) la cache L2 es como mucho el doble de rápida que la memoria principal
  - b) la cache L2 es al menos el doble de rápida que la memoria principal
  - c) la cache L1 es como mucho tres veces más rápida que la cache L2
  - > d) la cache L1 es al menos tres veces más rápida que la cache L2
- sólo sería exactamente 3 si el tiempo de cómputo adicional a los accesos de memoria fuera despreciable. Como en general no lo es, los accesos deben ser aún más rápidos para que el incremento sea 3x.

Matemáticamente, el enunciado dice que  $c+m1=1$ ,  $c+m2=3$ , entonces  $(c+m2)/(c+m1)=3$ , y despejando sale  $m1=(m2)/3-2c/3$ , es decir, sólo si  $c=0$  entonces  $m1=(m2)/3$ , y si  $c>0$  entonces  $m1<(m2)/3$

6. Suponer una memoria cache con las siguientes propiedades: Tamaño: 512 bytes. Política

de reemplazo: LRU. Estado inicial: vacía (todas las líneas inválidas). Suponer que para la siguiente secuencia de direcciones enviadas a la cache: 0, 2, 4, 8, 16, 32, la tasa de acierto es 0.33. ¿Cuál es el tamaño de bloque de la cache?

- a) 4 bytes
- > b) 8 bytes
- c) 16 bytes
- d) Ninguno de los anteriores

7. En la práctica de la cache, el código de line.cc incluye la sentencia

```
for (unsigned long long line=1;  
line<=LINE; line<<=1) { ... }
```

¿Qué objetivo tiene la expresión line<<=1?

- a) sacar un uno (1) por el stream line
- b) volver al principio del vector cuando el índice exceda la longitud del vector
- > c) duplicar el tamaño del salto en los accesos al vector respecto a la iteración anterior
- d) salir del bucle si el tamaño de línea se volviera menor o igual que 1 para algún elemento del vector

8. En la práctica de cache hemos hecho una gráfica con el código size.cc ¿Qué forma tiene la gráfica que se debe obtener?

- a) Forma de U (o V) con un tramo descendente y otro ascendente
- b) Forma de U (o V) invertida, con un tramo ascendente y otro descendente
- c) Forma de /, una gráfica siempre creciente y sin escalones
- > d) Una escalera con varios tramos horizontales

9. Abajo se ofrece el listado de una función para multiplicar matrices  $C = A \times B$ .

```
void mult_matr(float A[N][N], float B[N][N], float C[N][N]){\n/* Se asume valor inicial C = {0,0...} */\n\nint i,j,k;\n\nfor (i=0; i<N; i++)\n    for (j=0; j<N; j++)\n        for (k=0; k<N; k++)\n            C[i][j] += A[i][k] * B[k][j];\n}
```

Suponer que:

- El computador tiene una cache de datos de 8 MB, 16-vías, líneas de 64 bytes.
- N es grande, una fila o columna no cabe completa en cache.
- El tamaño de los tipos de datos es como en IA32.
- El compilador optimiza el acceso a  $C[i][j]$  en un registro.

Imaginar que se modifica la última sentencia (el cuerpo anidado) por esta otra

$C[i][j] += A[i][k] * B[j][k];$

de manera que se calcule  $C = A \times B'$  ( $A$  por traspuesta de  $B$ ). Aproximadamente, ¿qué tasa de fallos se podría esperar de esta nueva función para valores grandes de  $N$ ?

- a) 1/8
- b) 1/4
- c) 1/2
- > d) 1/16

uno de cada 16  $A[i][k]$  y otro de cada 16  $B[i][k]$

10. En el programa "size" de la práctica de la cache, si el primer escalón pasa de tiempo=2 para un tamaño de vector menor que 32KB a tiempo=8 para un tamaño mayor que 32KB, podemos asegurar que:

---

(Nota: en las opciones de respuesta, "rapidez" se refiere a  $T_i$ , el tiempo de acceso efectivo al nivel  $i$ , no a ti, el tiempo de acceso propio al nivel  $i$ . Si dice "La cache L1 es 4x más rápida que L2", se refiere a que  $T_1=T_2/4$ , no a que  $t_1=t_2/4$ . Ver T6 tr.26)

- 
- a) La cache L1 es como mucho cuatro veces más rápida que la cache L2
  - b) La cache L1 es seis veces más rápida que la cache L2
  - > c) La cache L1 es al menos cuatro veces más de rápida que la cache L2
- sólo sería exactamente 4 si el tiempo de cómputo adicional a los accesos de memoria fuera despreciable. Como en general no lo es, los accesos deben ser aún más rápidos para que el incremento sea 4x.

Matemáticamente, el enunciado dice que  $c+m_1=2$ ,  $c+m_2=8$ , entonces  $(c+m_2)/(c+m_1)=4$ , y despejando sale  $m_1=(m_2)/4-3c/4$ , es decir, sólo si  $c=0$  entonces  $m_1=(m_2)/4$ , y si  $c>0$  entonces  $m_1<(m_2)/4$

d) La cache L1 es cuatro veces más rápida que la cache L2

11. En la práctica de la cache, el código de size.cc accede al vector saltando de 64 en 64. ¿Por qué?

- a) Porque cada elemento del vector ocupa 64 bytes
- b) Para recorrer el vector más rápidamente
- c) Porque el tamaño de cache L1 de todos los procesadores actuales es de 64KB
- > d) Para anular los aciertos por localidad espacial, esto es, que sólo pueda haber aciertos por localidad temporal

12. Sea un computador de 32 bits con una memoria cache L1 para datos de 32 KB y líneas de 64 bytes asociativa por conjuntos de 2 vías. Dado el siguiente fragmento de código:

```
int v[262144];  
for (i = 0; i < 262144; i += 8)  
    v[i] = 9;
```

¿Cuál será la tasa de fallos aproximada que se obtiene en la ejecución del bucle anterior?

- a) 1 (todo son fallos)
- b) 1/8 (un fallo por cada 8 accesos)
- c) 0 (ningún fallo)
- d) 1/2 (mitad aciertos, mitad fallos)

es un array de ints (4B) y se salta  $i+=8$ , los accesos están separados 32B y las líneas son de 64B, los accesos en iteraciones par son fallos, en iteraciones impar son aciertos.

13. En la práctica de la cache, el código de “line.cc” incluye la sentencia

```
for (unsigned long long line=1; line<=LINE; line<<=1) { ... }
```

¿Qué objetivo tiene la expresión  $line<<=1$ ?

- a) sacar un uno (1) por el stream line
- b) volver al principio del vector cuando el índice exceda la longitud del vector
- c) salir del bucle si el tamaño de línea se volviera menor o igual que 1 para algún elemento del vector
- > d) duplicar el tamaño del salto en los accesos al vector respecto a la iteración anterior

15. Abajo se ofrece el listado de una función para multiplicar matrices  $C = A \times B$ .

```
void mult_matr(float A[N][N], float B[N][N], float C[N][N]) {  
    /* Se asume valor inicial C = {0,0,...} */
```

```

int i,j,k;
for (i=0; i<N; i++)
for (j=0; j<N; j++)
for (k=0; k<N; k++)
C[i][j] += A[i][k] * B[k][j];
}

```

Suponer que:

- El computador tiene una cache de datos de 8 MB, 16-vías, líneas de 64 bytes.
- N es grande, una fila o columna no cabe completa en cache.
- El tamaño de los tipos de datos es como en IA32.
- El compilador optimiza el acceso a C[i][j] en un registro.

Aproximadamente, ¿qué tasa de fallos se podría esperar de esta función para valores grandes de N?

-> a) 1/2

1/2 por cada B[k][j] + 1/32 por cada 16-ésimo A[i][k]

b) 1/4

c) 1/16

d) 1/8

16. El código del programa "size" de la práctica de la cache accede al vector saltando...

a) de byte en byte

-> b) de 64 en 64 bytes

es el tamaño de línea deducido en la práctica "line"

c) de 1 KB en 1 KB

d) de 64 KB en 64 KB

18. En el programa line.cc de la práctica de cache, si para cada tamaño de línea (line)recorremos una única vez el vector, la gráfica resultante es decreciente porque:

a) Cada vez que line aumenta al doble, el número de aciertos por localidad temporal aumenta, porque ya habíamos accedido a cada posición i del vector cuando lo recorrimos en el punto anterior del eje X.

b) Cada vez que line aumenta al doble, el número de aciertos por localidad espacial aumenta, porque ya habíamos accedido a cada posición i-1 del vector cuando lo recorrimos en el punto

anterior del eje X.

c) Cada vez que line aumenta al doble, se accede con éxito a más posiciones del vector en niveles de la jerarquía de memoria más rápidos.

-> d) Cada vez que line aumenta al doble, realizamos la mitad de accesos al vector que para el valor anterior.

19. Sea un computador de 32 bits con una memoria cache L1 para datos de 32 KB y líneas de 64 bytes asociativa por conjuntos de 2 vías. Dado el siguiente fragmento de código:

```
int v[262144];
for (i = 0; i < 262144; i += 2)
    v[i] = 9;
```

¿Cuál será la tasa de fallos aproximada que se obtiene en la primera ejecución del bucle anterior?

a) 0 (ningún fallo)

b) 1/2 (mitad aciertos, mitad fallos)

-> c) 1/8 (un fallo por cada 8 accesos)

d) 1 (todo son fallos)

20. En el programa line.cc de la práctica de cache, si para cada tamaño de línea (line) recorremos una única vez el vector, la gráfica resultante es decreciente porque:

a) hay un mayor historial de accesos y es más probable que un nuevo acceso sea un acierto

b) cada vez los tamaños de línea escogidos van decreciendo y se tarda menos en leerlos

c) cada vez los tamaños de línea escogidos van decreciendo y hay menor localidad espacial

• d) el vector se indexa con la variable de control del bucle, con un incremento o paso de line

el bucle es for (unsigned i = 0; i < bytes.size(); i += line), si line se duplica se realizan la mitad de accesos

21. ¿En qué unidades se suelen medir las capacidades de almacenamiento de los niveles de cache L1, L2 y L3 de un microprocesador actual (2017-2018)?

a) L1 en MB, L2 en MB, L3 en GB.

• b) L1 en KB, L2 en KB o MB, L3 en MB.

valores típicos en laboratorio: L1 4x 32KB, L2 4x 256KB, L3 6MB

c) L1 en MB, L2 en GB, L3 en GB o TB.

d) L1 en KB, L2 en MB, L3 en GB.

22.Un servidor tiene dos procesadores Intel Xeon E5-2620 v3@ 2.40GHz (2,4 Ghz, 6 núcleos, 12 hebras, 15MiB de caché L3, reloj DDR4 a 1866 MHz). Suponga que un proceso, que se ejecuta en un único núcleo, tiene que ordenar un vector de datos de usuarios accediendo repetidamente a sus elementos. Cada elemento es una estructura de datos y tiene un tamaño de 4 KB. Segundo <http://www.cpu-world.com/> los tamaños de cache son:

Cache: L1 data    L1 instr.    L2    L3

Size: 6 x 32KB 6 x 32KB 6 x 256KB 15MB

Si representamos en una gráfica las prestaciones en función del número de usuarios a ordenar, ¿para qué límites teóricos en el número de usuarios se observarán saltos en las prestaciones debidos a accesos a la jerarquía de memoria?

- a) 4 / 32 / 512 usuarios
- b) 8 / 64 / 3840 usuarios
- c) 32 / 256 / 8192 usuarios
- d) 48 / 384 / 23040 usuarios

23.En la práctica de la cache, en size.cc se realiza un número fijo y grande de accesos, pero podríamos haber recorrido una única vez el vector (saltando también de 64 en 64). Al dibujar la gráfica del tiempo de bucle en función del tamaño del vector...

- a) de ambas formas sale gráfica creciente
- b) recorriendo una vez sale gráfica creciente

sí, cada vez hay más fallos porque el vector es más grande

en clase hemos repetido muchas más veces para que haya localidad temporal si el vector cabe en un nivel dado de cache, baje el tiempo de acceso y quede marcado el escalón en la gráfica

- c) con num. fijo accesos sale gráfica creciente
- sí, lo hemos hecho en clase para comparar tiempos acceso distintos niveles
- d) de ninguna de las dos formas sale gr. Creciente

24.Suponga una memoria cache con las siguientes propiedades: Tamaño: 512 bytes. Política de reemplazo: LRU. Estado inicial: vacía (todas las líneas inválidas). Suponga que para la siguiente secuencia de direcciones enviadas a la cache: 1, 2, 4, 8, 16, 32, la tasa de acierto es 0,333.

¿Cuál es el tamaño de línea de la cache?

- a) 4 bytes
- b) 8 bytes
- c) 16 bytes
- d) 22 bytes

25. Suponga una memoria cache con las siguientes propiedades: Tamaño: 512 bytes. Política de reemplazo: LRU. Estado inicial: vacía (todas las líneas inválidas). Suponga que para la siguiente secuencia de direcciones enviadas a la cache: 0, 10, 16, 20, 30, 32, 40, 50, 60, 64, la tasa de acierto es 0,50. ¿Cuál es el tamaño de línea de la cache?

- a) 4 bytes
- b) 8 bytes
- c) 16 bytes
- d) Ninguno de los anteriores

26. En la práctica de cache hemos hecho una gráfica con el código line.cc ¿Qué forma tiene la gráfica que se puede obtener?

- a) Forma de U (o V), con un tramo descendente y otro ascendente
- b) Forma de /, una gráfica siempre creciente
- c) Forma de media U seguida de \, es decir, un tramo descendente suave, un pequeño tramo horizontal, y un tramo descendente lineal
- d) Una escalera con varios tramos horizontales

27. En el programa line.cc de la práctica "cache", se accede al vector saltando...

- a) de byte en byte
  - b) de 64 en 64 bytes
  - c) de 1 KB en 1 KB
  - d) de line en line bytes, donde line barre los valores desde 1 hasta 1K en un bucle for
28. ¿Qué forma tiene la gráfica que se debe obtener con el código size.cc?
- a) Forma de U, con un tramo descendente y otro ascendente.
  - b) Forma de  $\cap$ , con un tramo ascendente y otro descendente.
  - c) Una escalera decreciente con varios tramos horizontales.
  - d) Una escalera creciente con varios tramos horizontales.

**Nombre:**
**DNI:**
**Grupo:**

### Test de Teoría (3.0p)

Todas las preguntas son de elección simple sobre 4 alternativas.

Cada respuesta vale 3/30 si es correcta, 0 si está en blanco o claramente tachada, -1/30 si es errónea.

Anotar las respuestas (a, b, c o d) en la siguiente tabla.

|   |   |   |   |   |   |   |   |   |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |
|---|---|---|---|---|---|---|---|---|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|
| 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 | 11 | 12 | 13 | 14 | 15 | 16 | 17 | 18 | 19 | 20 | 21 | 22 | 23 | 24 | 25 | 26 | 27 | 28 | 29 | 30 |
|   |   |   |   |   |   |   |   |   |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |

1. Si almacenamos según el criterio little-endian la palabra de 64 bits 0xFACEB00C a partir de la dirección 0xCAFEBAE, el byte 0xCE quedará almacenado en la dirección:  
 a. 0xCAFEBAC1  
 b. 0xCAFEBAC0  
 c. 0xCAFEBABF  
 d. 0xCAFEBABE

---

2. ¿Qué novedad se desarrolló en la tercera generación de computadores?  
 a. Los circuitos integrados  
 b. Los primeros lenguajes de programación de alto nivel  
 c. Los microprocesadores CISC  
 d. Los microprocesadores RISC

---

3. En X86-64, el registro contador de programa se denomina:  
 a. RIP  
 b. EIP  
 c. IP  
 d. R15

---

4. La primera letra (l) de la instrucción lea:  
 a. indica que la instrucción usa ordenación de bytes little-endian  
 b. indica que la instrucción trabaja con un operando destino de 32 bits (long word)  
 c. indica que la instrucción afecta a los 16 bits menos significativos del operando destino (low word)  
 d. forma parte del nemotécnico de la instrucción

---

5. ¿Cuál es la diferencia entre las instrucciones mov y lea?  
 a. lea accede a la posición indicada, mientras que mov no lo hace  
 b. mov accede a la posición indicada, mientras que lea no lo hace  
 c. lea puede usarse para copiar un registro a otro, mientras que mov no  
 d. mov puede usarse para copiar un registro a otro, mientras que lea no

---

6. La instrucción not:  
 a. realiza el complemento a dos  
 b. realiza el complemento a uno (cambiar unos por ceros y ceros por unos)  
 c. realiza la operación no-or (or negada)  
 d. realiza un salto condicional si negativo

---

7. La instrucción JGE / JNL provoca un salto si...  
 a. SF = 1  
 b. CF = 1  
 c. SF = 0  
 d. OF = SF

---

8. Un overflow nunca puede ocurrir cuando:  
 a. se suman dos números positivos  
 b. se suman dos números negativos  
 c. se suma un número positivo a un número negativo  
 d. se resta un número positivo de un número negativo

---

9. Despues de ejecutar una instrucción de suma sobre dos números con signo de la que

sabemos que no provocará overflow (los dos números son pequeños en valor absoluto), queremos comprobar si el resultado de la suma es menor que 0. ¿Qué flag necesita comprobar la instrucción de salto condicional equivalente a “if (resultado<0) then goto label”?

- a. CF
- b. OF
- c. SF
- d. ZF

**10.** La instrucción cmovb %edx, %eax

- a. copia en %eax el contenido de %edx si el indicador de acarreo es 1
- b. copia el byte bajo de %edx en el byte bajo de %eax
- c. copia en %eax el byte de memoria apuntado por la dirección contenida en %edx
- d. copia en %eax el contenido de %edx si %eax es menor que %edx

**11.** ¿Cuál de las siguientes afirmaciones sobre la instrucción leave es **cierta**?

- a. Se ejecuta justo después de retornar de un procedimiento
- b. Equivale a pop %ebp seguida de mov %ebp,%esp
- c. Equivale a mov %esp,%ebp seguida de pop %ebp
- d. No es obligatorio usarla. En su lugar puede realizarse una secuencia explícita de operaciones mov y pop

**12.** Para crear espacio en la pila para variables locales sin inicializar suele realizarse la siguiente operación:

- a. Restar una cantidad positiva a EBP
- b. Sumar una cantidad positiva a EBP
- c. Restar una cantidad positiva a ESP
- d. Sumar una cantidad positiva a ESP

**13.** ¿Cuál de los siguientes lenguajes **no** permite el paso de parámetros por referencia?

- a. Pascal
- b. C
- c. C++
- d. FORTRAN

**14.** En la secuencia de programa siguiente:

```
804854e:e8 3d 06 00 00  call 8048b90 <main>
8048553:50      pushl %eax
```

¿cuál es el valor que introduce en la pila la instrucción call?

- a. 804854e
- b. 804854f
- c. 8048b90
- d. 8048553

**15.** En el fragmento de código

```
804854e:e8 3d 06 00 00  call 8048b90 <main>
8048553:50          pushl %eax
```

la instrucción call suma al contador de programa la cantidad:

- a. 0000063d
- b. 08048553
- c. 0804854e
- d. 50

**16.** Es responsabilidad del procedimiento llamado salvaguardar los registros:

- a. %ebx, %esi, %edi
- b. %eax, %edx, %ecx
- c. %eax, %ebx, %ecx, %edx
- d. %esi, %edi

**17.** Al llamar a una función de 2 argumentos foo(arg1, arg2) , ¿cuál es el orden correcto en el que se ejecutan las instrucciones? (suponiendo convención de llamada x86 cdecl, y que foo requiere ajustar marco de pila, esto es, salvar %ebp)

- a. push arg1, push arg2, call foo, push %ebp
- b. push arg1, push arg2, push %ebp , call foo
- c. push arg2, push arg1, call foo, push %ebp
- d. push arg2, push arg1, push %ebp, call foo

**18.** Cuando se ejecuta la instrucción ret al final de una subrutina:

- a. la dirección de comienzo de la pila se transfiere al puntero de pila
- b. la dirección de memoria de la instrucción ret se transfiere al contador de programa
- c. la dirección almacenada en la cima de la pila se transfiere al contador de programa
- d. la dirección almacenada en la cima de la pila se transfiere al puntero de pila

**19.** En IA-32 la pila es:

- a. un registro de 16 bits en el microprocesador
- b. un registro de 32 bits en el microprocesador
- c. una dirección de memoria de 32 bits almacenada en el contador de programa

d. un conjunto de posiciones de memoria usadas para almacenar información temporal durante la ejecución del programa

20. En IA-32 el puntero de pila es:

- a. un registro de 16 bits en el microprocesador
- b. un registro de 32 bits en el microprocesador
- c. una dirección de memoria de 32 bits almacenada en el contador de programa
- d. un conjunto de posiciones de memoria usadas para almacenar información temporal durante la ejecución del programa

21. ¿Cuál de las siguientes afirmaciones \*NO\* es cierta? (entender que x86=IA-32)

- a. x86-64 proporciona un espacio de memoria virtual mayor que x86
- b. Las disciplinas de pila para x86 y x86-64 son diferentes
- c. x86 usa %ebp como puntero base para el marco de pila
- d. x86-64 usa %rbp como puntero base para el marco de pila

22. Alguna de las siguientes \*NO\* es una operación básica de la unidad de control:

- a. Transferir un registro a otro
- b. (Leer / escribir) un registro (de / a) memoria
- c. (Guardar / recuperar) registro (en/de) la pila
- d. Realizar una operación de la ALU y guardar el resultado en un registro

23. Un computador tiene una memoria de control de 16000 palabras de 250 bits, de las que 447 son diferentes. ¿Cuántos bits ahorraremos usando nanoprogramación en lugar de micropogramación?

- a. 3744250
- b. 259206
- c. 287935
- d. Ninguno de los resultados anteriores es exacto

24. Un sistema no segmentado tarda 20 ns en procesar una tarea. La misma tarea puede ser procesada en un cauce (pipeline) de 4 segmentos con un ciclo de reloj de 5 ns. Cuando se procesan muchas tareas, la ganancia máxima de velocidad que se obtiene se aproxima a:

- a. 5
- b. 4
- c. 0,25

d. 20

25. En la técnica de salto retardado:

- a. el compilador puede reorganizar el código para llenar los huecos de retardo con instrucciones útiles
- b. el compilador no puede insertar operaciones NOP en los huecos de retardo
- c. el salto se realiza varios ciclos antes de la instrucción de salto
- d. las instrucciones en los huecos de retardo se ejecutan unas veces y otras no

26. Al método de interacción con los periféricos, en los que el procesador vigila periódicamente el estado de los dispositivos mediante una encuesta activa se le denomina:

- a. daisy-chain
- b. interrupción
- c. polling
- d. DMA

27. ¿Cuántas señales de control se necesitan como mínimo para implementar un sistema de gestión de interrupciones?

- a. 1
- b. 2
- c. 3
- d. 4

28. Se dispone de un procesador con una frecuencia de reloj de 1 GHz. Se le conecta un dispositivo que genera 100.000 interrupciones por segundo. La rutina de servicio de interrupción ejecuta 500 instrucciones. El número medio de ciclos por instrucción es 2. ¿Qué porcentaje del tiempo dedica el procesador al dispositivo?

- a. 1%
- b. 10%
- c. 50%
- d. 90%

29. ¿Cuál de las siguientes afirmaciones acerca de la memoria es \*FALSA\*?

- a. La memoria dinámica usa señales de control RAS# y CAS#
- b. Las celdas de memoria dinámica están constituidas por un transistor y un condensador

c. Las celdas de memoria estática tienen que ser constantemente refrescadas

d. La memoria estática se emplea en las cachés L1 y L2

---

30. ¿Cuál de las siguientes afirmaciones acerca de la jerarquía de memoria es **\*FALSA\***?

- a. Acceder a los discos es órdenes de magnitud más lento que acceder a la RAM
  - b. Una memoria principal constituida por la tecnología más rápida es órdenes de magnitud más cara que la DRAM
  - c. La velocidad de acceso a la memoria principal ha crecido proporcionalmente a la velocidad del procesador
  - d. Un computador puede tener una pequeña cantidad de memoria rápida además de una gran cantidad de memoria más lenta
-

**Nombre:**
**DNI:**
**Grupo:**

## Test de Prácticas (4.0p)

**Todas las preguntas son de elección simple sobre 4 alternativas.**

**Cada respuesta vale 4/20 si es correcta, 0 si está en blanco o claramente tachada, -4/60 si es errónea.**

**Anotar las respuestas (a, b, c o d) en la siguiente tabla.**

| 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 | 11 | 12 | 13 | 14 | 15 | 16 | 17 | 18 | 19 | 20 |
|---|---|---|---|---|---|---|---|---|----|----|----|----|----|----|----|----|----|----|----|
|   |   |   |   |   |   |   |   |   |    |    |    |    |    |    |    |    |    |    |    |

1. La dirección efectiva del primer parámetro de llamada a una función suele calcularse desde el código de la función como:
- a. EBP+8
  - b. EBP-8
  - c. EBP+4
  - d. EBP-4
- 
2. El comienzo de un procedimiento que siga la convención cdecl es:
- a. mov %ebp,%esp; push %ebp
  - b. mov %esp,%ebp; push %ebp
  - c. push %ebp; mov %ebp,%esp
  - d. push %ebp; mov %esp,%ebp
- 
3. Considere una función C declarada así:
- ```
void fun4arg (int a,int b,int c,int d);
```
- Suponiendo que fun4arg se ha compilado para una máquina x86 IA-32 con enteros de 4 bytes, ¿cuál sería la dirección del argumento b relativa a %ebp, en el marco de pila de fun4arg?
- a. %ebp + 8
  - b. %ebp + 12
  - c. %ebp + 16
  - d. %ebp + 20
- 
4. ¿Cuál de las siguientes afirmaciones sobre las caches es **\*FALSA\***?
- a. Casi ningún procesador actual tiene memoria cache L2
- b. Las direcciones a las que accede un programa no son completamente aleatorias, sino que se rigen por ciertos patrones de localidad
- c. Un procesador actual tiene varias cachés de nivel 1
- d. La caché de nivel 3 no contiene toda la memoria que maneja el programa
- 
5. En un sistema Linux x86-64, ¿cuál de las siguientes variables ocupa más bytes en memoria?
- a. char a[7]
  - b. short b[3]
  - c. int \*c
  - d. float d
- 
6. En la práctica "suma" se pide sumar una lista de 32 enteros SIN signo de 32bits en una plataforma de 32bits sin perder precisión, esto es, evitando acarreos. ¿Cuál es el mínimo valor entero que repetido en toda la lista causaría acarreo con 32bits (sin signo)?
- a. 0xfc00 0000
  - b. 0xfbff ffff
  - c. 0x0800 0000
  - d. 0x07ff ffff
- 
7. En la práctica "suma" se pide sumar una lista de 32 enteros CON signo de 32bits en una plataforma de 32bits sin perder precisión, esto es, evitando

desbordamiento. ¿Cuál es el valor negativo más pequeño (en valor absoluto) que repetido en toda la lista causaría desbordamiento con 32bits (en complemento a 2)?

- a. 0xfc00 0000
  - b. 0xfbff ffff**
  - c. 0xf800 0000
  - d. 0xf800 0001
- 

8. ¿Qué valor contendrá edx tras ejecutar las siguientes instrucciones?

```
xor %eax, %eax
sub $1, %eax
cltd
idiv %eax
```

- a. 0**
  - b. 1
  - c. -1
  - d. No puede saberse con los datos del enunciado
- 

9. La práctica "popcount" debía calcular la suma de bits de los elementos de un array. Un estudiante entrega lo siguiente:

```
int popcount4(unsigned* array,
              int len) {
    int i, j, res = 0;
    for(i = 0; i < len; ++i) {
        unsigned x = array[i];
        int n = 0;
        do {
            n += x & 0x01010101L;
            x >>= 1;
        } while(x);
        for(j = 16; j == 1; j /= 2){
            n ^= (n >>= j);
        }
        res += n & 0xff;
    }
    return res;
}
```

Esta función popcount4:

- a. produce el resultado correcto**
  - b. fallaría con **array={0,1,2,3}**
  - c. fallaría con **array={1,2,4,8}**
  - d. no es correcta pero el error no se manifiesta en los ejemplos propuestos, o se manifiesta en ambos
- 

10. La práctica "paridad" debía calcular la suma de paridades impar (XOR de todos los bits) de los elementos de un array. Un estudiante entrega la siguiente versión de parity6:

```
int parity6(unsigned * array,
            int len) {
    int i, result = 0;
    unsigned x;
    for (i=0; i<len; i++) {
        x = array[i];
        asm("mov %[x], %%edx \n\t"
            "shr $16, %%edx \n\t"
            "shr $8, %%edx \n\t"
            "xor %%edx,%%edx \n\t"
            "setp %%dl \n\t"
            "movzxx %%dl, %[x] \n\t"
            : [x] "+r" (x)
            :
            : "edx"
        );
        result += x;
    }
    return result;
}
```

Esta función parity6:

- a. produce el resultado correcto
  - b. fallaría con **array={0,1,2,3}****
  - c. fallaría con **array={1,2,4,8}**
  - d. no es correcta pero el error no se manifiesta en los ejemplos propuestos, o se manifiesta en ambos
- 

11. En la práctica "paridad" se pide calcular la suma de paridades de una lista de enteros sin signo. Suponer que un estudiante entrega la siguiente versión:

```
int paridad5(unsigned* array,
              int len) {
    int i, k, result = 0;
    unsigned x;
    for (i = 0; i < len; i++) {
        x = array[i];
        for (k = 16; k == 1; k /= 2)
            x ^= x >> k;
        result += (x & 0x01);
    }
    return result;
}
```

Esta función:

- a. es correcta

- b. falla para `array={0,1,2,3}`
  - c. falla para `array={1,2,3,4}`
  - d. no se puede marcar una y sólo una de las opciones anteriores
- 

12. Utilizando la sentencia asm, las denominadas restricciones que se indican al final de dicha sentencia, involucran a:

- a. solamente las entradas
  - b. solamente las salidas
  - c. solamente los sobrescritos
  - d. Ninguna de las anteriores es cierta
- 

13. En la realización de la práctica de la bomba digital, una parte del código máquina es el siguiente:

```
0x080486e8 <main+120>: call 0x8048524 <strcmp>
0x080486ed <main+125>: test %eax,%eax
0x080486ef <main+127>: je 0x80486f6<main+134>
0x080486f1 <main+129>: call 0x8048604 <boom>
```

¿Cuál de los siguientes comandos cambiaría el salto condicional por un salto incondicional?

- a. set \$0x080486ef=0xeb
  - b. set \*(char\*)0x080486ef=0xeb
  - c. set \*(char\*)0x080486f6=jmp
  - d. set %0x080486ef=0xeb
- 

14. En una bomba como las estudiadas en prácticas, del tipo...

```
0x0804873f <main+207>: call 0x8048504 <scanf>
0x08048744 <main+212>: mov 0x24(%esp),%edx
0x08048748 <main+216>: mov 0x804a044,%eax
0x0804874d <main+221>: cmp %eax,%edx
0x0804874f <main+223>: je 0x8048756<main+230>
0x08048751 <main+225>: call 0x8048604 <boom>
0x08048756 <main+230>: ...
```

la contraseña es...

- a. el entero 0x804a044
  - b. el entero almacenado a partir de la posición de memoria 0x804a044
  - c. el string almacenado a partir de la posición de memoria 0x24(%esp)
  - d. ninguna de las anteriores
- 

15. En una bomba como las estudiadas en prácticas, del tipo...

```
0x080486e8 <main+120>: call 0x8048524 <strcmp>
0x080486ed <main+125>: test %eax,%eax
0x080486ef <main+127>: je 0x80486f6 <main+134>
0x080486f1 <main+129>: call 0x8048604 <boom>
0x080486f6 <main+134>: ...
```

la contraseña es...

- a. el valor que tenga %eax
  - b. el string almacenado a partir de donde apunta %eax
  - c. el entero almacenado a partir de donde apunta %eax
  - d. ninguna de las anteriores
- 

16. El servidor de SWAD tiene dos procesadores Xeon E5540 con 4 núcleos cada uno. Cada procesador tiene 4 caches L1 de instrucciones de 32 KB, 4 caches L1 de datos de 32 KB, 4 caches unificadas L2 de 256 KB y una cache unificada L3 de 8MB. Suponga que un proceso swad, que se ejecuta en un núcleo, tiene que ordenar un vector de estudiantes accediendo repetidamente a sus elementos. Cada elemento es una estructura de datos de un estudiante y tiene un tamaño de 4KB. Si representamos en una gráfica las prestaciones en función del número de estudiantes a ordenar, ¿para qué límites teóricos en el número de estudiantes se observarán saltos en las prestaciones debidos a accesos a la jerarquía de memoria?

- a. 4 / 32 / 512 estudiantes
  - b. 8 / 64 / 2048 estudiantes
  - c. 16 / 32 / 64 estudiantes
  - d. 32 / 256 / 8192 estudiantes
- 

17. En la práctica de la cache, el código de line.cc incluye la sentencia

```
for (unsigned line=1;line<=MAXLINE;
     line<<=1) { ... }
```

¿Qué objetivo tiene la expresión line<<=1?

- a. Salir del bucle si el tamaño de línea se volviera menor o igual que 1 para algún elemento del vector
- b. Duplicar el tamaño del salto en los accesos al vector respecto a la iteración anterior
- c. Volver al principio del vector cuando el índice exceda la longitud del vector
- d. Sacar un uno (1) por el stream line

18. Sea un computador de 32 bits con una memoria caché L1 para datos de 32 KB y líneas de 64 bytes asociativa por conjuntos de 2 vías. Dado el siguiente fragmento de código:

```
int v[262144];  
for (i = 0; i < 262144; i += 2)  
    v[i] = 9;
```

¿Cuál será la tasa de fallos aproximada que se obtiene en la primera ejecución del bucle anterior?

- a. 0 (ningún fallo)
- b. 1/2 (mitad aciertos, mitad fallos)
- c. 1/8 (un fallo por cada 8 accesos)
- d. 1 (todo son fallos)

19. Abajo se ofrece el listado de una función para multiplicar matrices  $C = A \times B$ .

```
void mult_matr(    float A[N][N],  
                  float B[N][N],  float C[N][N]) {  
    /* Se asume valor inicial C = {0,0...} */  
    int i,j,k;  
    for (i=0; i<N; i++)  
        for (j=0; j<N; j++)  
            for (k=0; k<N; k++)  
                C[i][j] += A[i][k] * B[k][j];  
}
```

Suponer que:

- El computador tiene una cache de datos de 8 MB, 16-vías, líneas de 64 bytes.
- N es grande, una fila o columna no cabe completa en cache.
- El tamaño de los tipos de datos es como en IA32.
- El compilador optimiza el acceso a  $C[i][j]$  en un registro.

Aproximadamente, ¿qué tasa de fallos se podría esperar de esta función para valores grandes de N?

- a. 1/16
- b. 1/8
- c. 1/4
- d. 1/2

20. Con los mismos supuestos, imaginar que se modifica la última sentencia (el cuerpo anidado) por esta otra

```
C[i][j] += A[i][k] * B[j][k];
```

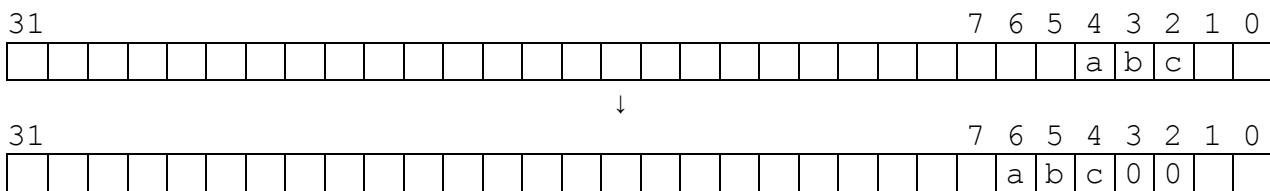
de manera que se calcule  $C = A \times B'$  (A por traspuesta de B). Aproximadamente, ¿qué tasa de fallos se podría esperar de esta nueva función para valores grandes de N?

- a. 1/16
- b. 1/8
- c. 1/4
- d. 1/2

Nombre:	
DNI:	Grupo:

### Examen de Problemas (3,0 p)

1. **Ensamblador** (0.6 puntos). Tenemos un número almacenado en una variable x de tipo unsigned int. Queremos colocar los bits 4,3,2 en los bits 6,5,4, poniendo los bits 3 y 2 a 0, sobrescribiendo los bits 6 y 5 (cuyo valor se pierde), y dejando el resto de bits intactos.



Escriba una expresión en C para realizar esta operación, y un pequeño fragmento en ensamblador de IA32 que realice la misma operación.

2. **Ensamblador**. (0.2 puntos). Al ejecutar el fragmento de código:

```
leal -48(%eax), %edx
cmpb $9, %edx
ja .L2
```

¿Para qué valores de %eax se salta a .L2?.

3. **Disposición de estructuras** (0.6 puntos). Considerar las siguientes declaraciones de estructuras en C:

```
struct a {
    float* f;
    char c;
    int i;
    char z[4];
    double d;
    short s;
};

struct b {
    struct a a1;
    int j;
    struct a a2;
};
```

- Mostrar la disposición de **struct a** en memoria, en una máquina Linux en modo IA32 (es decir, con direcciones de 32bits), mediante un dibujo como los estudiados en clase, donde se indiquen los desplazamientos y tamaños de cada campo. Marcar los bytes de relleno (si los hubiera) tachándolos o sombreándolos. ¿Cuántos bytes usa **struct a** en total en este caso?
- ¿Cuántos bytes usa **struct b** en una máquina Linux en modo IA32 (con direcciones de 32bits)?

4. **Entrada/Salida y Memoria** (0.5 puntos). Enunciado.

- Dibuje un esquema de conexiónado detallado para un procesador con E/S mapeada en memoria con un sistema de memoria de  $48K \times 8$  formado a partir de chips de memoria SRAM de  $16K \times 8$ . En el mapa de memoria y E/S toda la memoria debe situarse en direcciones consecutivas a partir de la dirección 0x4000.

Las señales del procesador son:

- bus de direcciones de 16 bits
- bus de datos de 8 bits
- señal R/W#

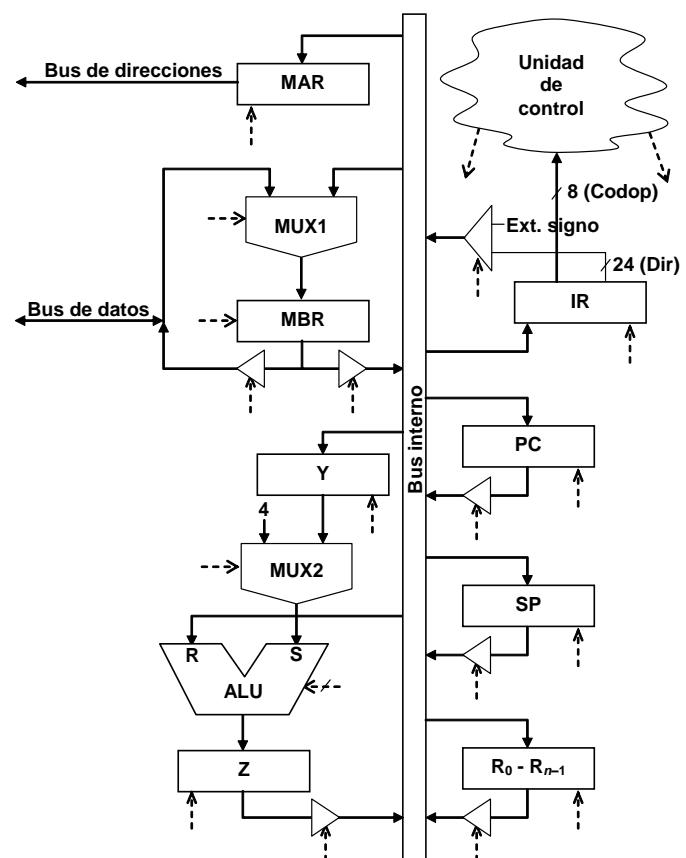
Las señales de cada chip de memoria son:

- bus de direcciones de 14 bits
- señal CS
- señal R/W#

b) ¿Cuántas direcciones están disponibles para puertos de E/S?

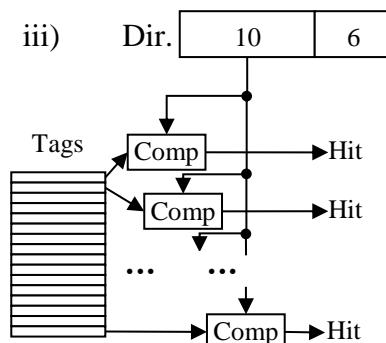
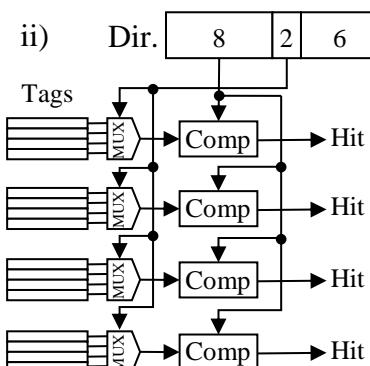
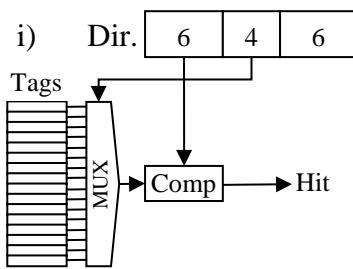
5. **Unidad de control** (0.5 puntos). La figura muestra el camino de datos de un procesador de 32 bits que direcciona la memoria por bytes y en el que cada instrucción ocupa una palabra completa (4 bytes). El multiplexor MUX2 selecciona o bien la salida del registro Y o un valor constante igual a 4. Las operaciones de lectura o escritura en memoria consumen un ciclo de reloj. La ALU puede realizar varias operaciones, entre ellas R+S y R-S.

Escriba (en lenguaje de transferencia de registros o de alto nivel) los fragmentos de microprograma que para implementar las instrucciones CALL X (X es un desplazamiento de 24 bits relativo a PC, almacenado en IR) y RET, sin incluir la fase de captación de instrucción.



## 6. Memoria cache (0.6 puntos).

- Indique qué tipo de correspondencia de cache se usa en cada uno de los tres siguientes casos.
- Indique un (posible) nombre para cada campo de la dirección en cada caso.
- ¿Cuál es el tamaño de línea?
- ¿Cuántas direcciones de datos tiene la cache?
- ¿Cuántas direcciones tiene la memoria principal?



### Test de Teoría (3.0p)

<b>1</b>	<b>2</b>	<b>3</b>	<b>4</b>	<b>5</b>	<b>6</b>	<b>7</b>	<b>8</b>	<b>9</b>	<b>10</b>	<b>11</b>	<b>12</b>	<b>13</b>	<b>14</b>	<b>15</b>	<b>16</b>	<b>17</b>	<b>18</b>	<b>19</b>	<b>20</b>	<b>21</b>	<b>22</b>	<b>23</b>	<b>24</b>	<b>25</b>	<b>26</b>	<b>27</b>	<b>28</b>	<b>29</b>	<b>30</b>
b	a	a	d	b	b	d	c	c	a	d	c	b	d	a	a	c	c	d	b	d	c	a	b	c	b	b	c	c	

### Test de Prácticas (4.0p)

<b>1</b>	<b>2</b>	<b>3</b>	<b>4</b>	<b>5</b>	<b>6</b>	<b>7</b>	<b>8</b>	<b>9</b>	<b>10</b>	<b>11</b>	<b>12</b>	<b>13</b>	<b>14</b>	<b>15</b>	<b>16</b>	<b>17</b>	<b>18</b>	<b>19</b>	<b>20</b>
a	d	b	a	c	c	b	a	a	b	c	d	b	b	d	b	b	c	d	a

### Examen de Problemas (3.0p)

#### 1. Ensamblador (0.6 puntos).

Otras soluciones son posibles, siempre que produzcan el resultado correcto y no añadan complejidad innecesaria.  
 El enunciado no requería comentar el código, el comentario se ofrece sencillamente como explicación.

##### Expresión C:

Alternativa:

Braindamage:

Comentario: el primer & elimina los bits 6-2 (conserva el resto), el | con el segundo & recupera los bits 4-2 en posiciones 6-4  
 Versión braindamage es como el propio nombre indica <http://onlineslangdictionary.com/meaning-definition-of/brain-damaged>

##### Pequeño fragmento en ensamblador IA-32:

```
movb    x, %al      ; ó movl x, %eax
movb    %al, %dl      ; copia bits 7-0
andb    $0x83, %dl      ; bits 7,1,0 en DL
andb    $0x1C, %al      ; abc (4,3,2)en AL
shlb    $2, %al      ; pasarlos a 6,5,4
orb     %dl, %al      ; recuperar resto bits
movb    %al, x      ; ó movl %eax, x
```

##### Alternativa:

```
movl    x, %eax
movl    %eax, %edx
andl    $0xFFFFF83,%edx
andl    $0x1C, %eax
shll    $2, %eax
orl     %edx, %eax
movl    %eax, x
```

##### Alternativa GCC:

```
movl    x, %eax
leal    (,%eax,4), %edx
andl    $0xFFFFF83,%eax
andl    $0x70, %edx
orl     %edx, %eax
movl    %eax, x
```

#### 2. Ensamblador (0.2 puntos).

El enunciado no requería comentar el código, el comentario se ofrece sencillamente como explicación.

```
leal -48(%eax), %edx      ; edx = eax-48
cmpl $9 , %edx;      ; > 9?
ja .L2                  ; above (sin signo)
```

EAX podría contener un código ASCII, en cuyo caso LEA lleva el char '0' a valor 0, ninguno de los caracteres desde '0' a '9' cumplen la condición ('9' se queda justo en el valor 9), y el resto de caracteres la cumplen

Respuesta: se salta a .L2 para valores de **EAX = [0..47] ∪ [58..0xffffffff]**

Alternativa: \*NO\* se salta para valores de EAX = [48..57]

#### 3. Disposición de estructuras (0.6 puntos).

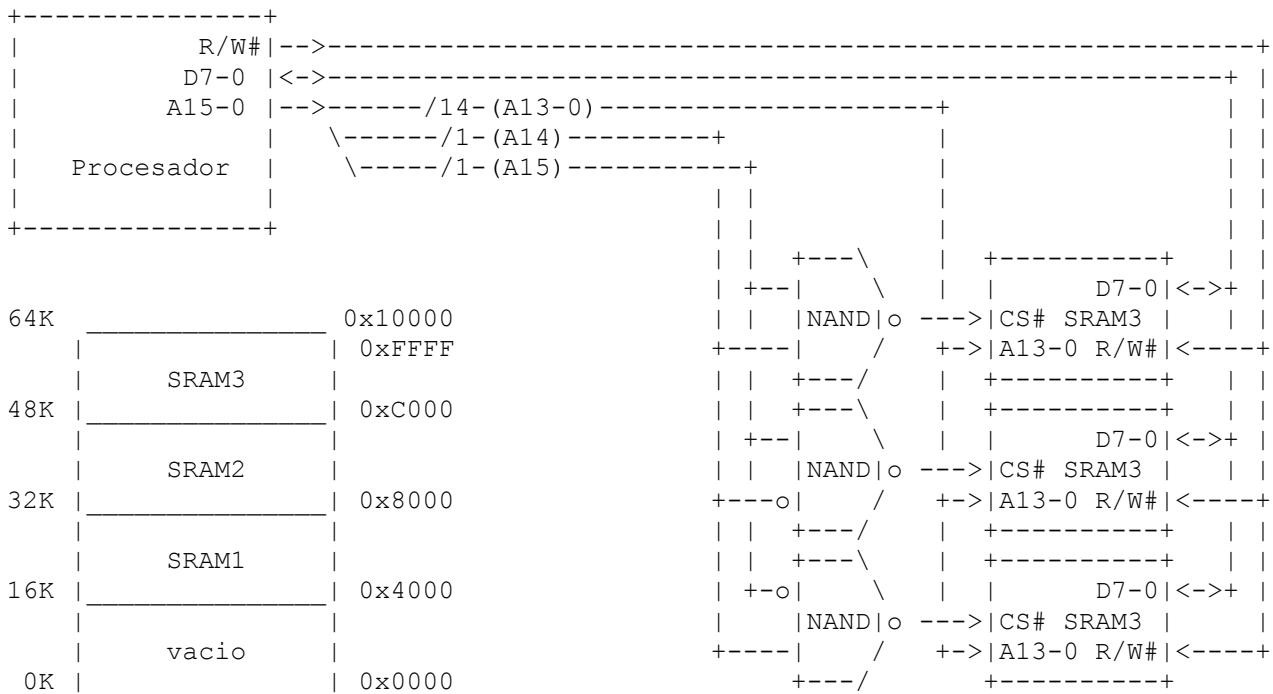
##### a) 28 bytes

0	4	5	8	12	16	24	2627
-----	-----	-----	-----	-----	-----	-----	-----
f	c   XXX   i	z0-3   d		s   XX			
-----	-----	-----	-----	-----	-----	-----	-----

##### b) 60 bytes

#### 4. Entrada/Salida y Memoria (0.5 puntos).

- a) Esquema de conexionado. El dibujo del mapa no se solicitaba, se añade como explicación.



- b) **16K** direcciones disponibles para E/S (cada posición 1B). Etiquetado “vacío” en el mapa de memoria.

#### 5. Unidad de Control (0.5 puntos).

**CALL:** Z = SP - 4  
 SP = Z ; MAR = Z  
 MBR = PC ; Y = PC  
 M[MAR] = MBR ; Z = IR + Y  
 PC = Z ; goto FETCH

**RET:** MAR = SP ; Z = SP + 4  
 MBR = M[MAR] ; SP = Z  
 PC = MBR ; goto FETCH

#### 6. Memoria cache (0.6 puntos).

- a) Correspondencia **directa** (i), asociativa por **conjuntos de 4** vías (ii), totalmente **asociativa** (iii)
- b) i) **etiqueta, marco, palabra**  
 ii) **etiqueta, conjunto, palabra**  
 iii) **etiqueta, palabra**  
 alternativas: etiqueta = (marca, tag), marco = (línea, bloque), palabra = (desplazamiento, offset)
- c) línea =  $2^6$  palabras = **64 pal**  
 d) cache =  $2^4$  líneas x  $2^6$  palabras/línea =  $2^{10}$  pal = **1K pal**  
 e)  $2^{(6+4+6)} = 2^{(8+2+6)} = 2^{(10+6)} = 2^{16} = \text{64K pal}$

#### 2. Ensamblador (0.2 puntos). Comentarios adicionales

El enunciado está basado en un caso real de una necesidad que se dio en una base de datos (SWAD), cuyo lenguaje de consultas permite expresiones similares a las de lenguaje C.

La versión “pobremente diseñada” se ha calificado como tal porque utiliza 2 AND (&) para realizar el trabajo que se puede hacer con uno solo (0xFFFFFFFF83), e invierte el orden “lógico humano” máscara(0x1C)-desplazamiento resultando en desplazamiento-máscara(0x70), haciendo la máscara más difícil de entender para un programador humano.

Curiosamente, GCC con bitfields produce esa misma máscara **0x70**: en un esfuerzo por ahorrar operaciones usa LEAL para conseguir el efecto combinado de las instrucciones 2 y 5 (MOV y SHL), con lo cual es obligatorio retrasar la máscara.

En clase no se han explicado bitfields de lenguaje C. El lenguaje C se diseñó inicialmente como lenguaje de programación de sistemas, en donde se anticipa que las operaciones a nivel de bits serán muy frecuentes. Un programa C completo para realizar la operación deseada sobre un número cualquiera podría ser:

```
#include <stdio.h>

int main (void)
{
    unsigned int x;
    union
    {
        struct {unsigned int :2, central:3;} original;
        struct {unsigned int :2, cero:2, central:3;} nueva;
        unsigned int x;
    } dato;

    printf("x? ");
    if (scanf ("%x",&x) !=1) return 1;
    printf("x original = %#08X\n",x);

    dato.x      = x;
    dato.nueva.central = dato.original.central;
    dato.nueva.cero   = 0;
    x               = dato.x;

    printf("x nuevo     = %#08X\n",x);

    return 0;
}
```

Al compilar con distintos niveles de optimización se obtiene el siguiente código ensamblador:

#### -O0

```
movzbl 24(%esp), %eax          ; EAX= 8 bits menos significativos x
shrb    $2, %al                ; eliminar bits 1-0
andl    $7, %eax              ; dejar sólo bits 4-2 en pos.2-0
andl    $7, %eax              ; gcc -O0 es así
movl    %eax, %edx             ; EDX=bits "abc"
sall    $4, %edx              ; ...en posiciones 6-4
movzbl 24(%esp), %eax          ; EAX= 8 bits menos significativos x
andl    $-113, %eax            ; ...& FFFFFFF8F ->hueco en pos.6-4
orl    %edx, %eax              ; ...|EDX -> pegarle "abc" pos.6-4
movb    %al, 24(%esp)           ; gcc -O0 es así, machaca x
movzbl 24(%esp), %eax          ; pero se lo vuelve a traer porque
andl    $-13, %eax              ; ...& FFFFFFF3 ->hueco en pos.3-2
movb    %al, 24(%esp)           ; ahora sí, guardar x
```

#### -O1

```
movl    28(%esp), %edx          ; EDX=x
leal    0(,%edx,4), %eax        ; EAX=x<<2
andl    $112, %eax              ; EAX=(x<<2) & 0x70
andl    $-125, %edx             ; EDX=x & FFFFFFF83
orl    %edx, %eax              ; 
movl    %eax, 28(%esp)           ; x= EAX | EDX
```

#### -O2

```
movl    28(%esp), %eax          ; EAX=x
leal    0(,%eax,4), %edx        ; EDX=x<<2
andl    $-125, %eax              ; EAX=x & FFFFFFF83
andl    $112, %edx              ; EDX=(x<<2) & 0x70
orl    %edx, %eax              ; 
movl    %eax, 28(%esp)           ; x= EAX | EDX
```

Obsérvese que GCC deduce de las posiciones de bits especificadas en los bitfields las máscaras necesarias. Con -O0 usa \$7 para extraer los bits 2-0 (“abc” ya desplazados a la derecha), \$-113 (0xFFFFFFF8F) para hacer hueco en bits 6-4, y muy aparatosamente \$-13 (0xFFFFFFFF3) para hacer hueco en posiciones 3-2. Con -O1 usa \$112 (**el mencionado 0x70**) para extraer los bits 6-4 y \$-125 (0xFFFFFFFF83) para hacer hueco en bits 6-2. Con -O2 usa las mismas máscaras.

**Nombre:**
**DNI:**
**Grupo:**

### Test de Teoría (3.0p)

Todas las preguntas son de elección simple sobre 4 alternativas.

Cada respuesta vale 3/30 si es correcta, 0 si está en blanco o claramente tachada, -1/30 si es errónea.

Anotar las respuestas (a, b, c o d) en la siguiente tabla.

1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30

1. El conjunto de todos los atributos de un sistema que son visibles para el programador y son necesarios para programar en lenguaje máquina se denomina:

- a. arquitectura del computador
- b. conjunto de componentes físicos del computador
- c. organización del computador
- d. repertorio de instrucciones máquina

2. ¿Cuál de las siguientes afirmaciones es cierta?

- a. la arquitectura Von Neumann de los computadores tradicionales consiste en tener almacenados los datos separados de las instrucciones en memorias distintas
- b. el registro de estado (flags) es un registro de propósito específico cuyo contenido puede ser visto directa o indirectamente por el usuario mediante el uso de ciertas instrucciones específicas
- c. la unidad de control necesita como entrada el registro contador de programa para saber cuál es la instrucción que debe ejecutar a continuación
- d. el registro de direcciones de memoria es un registro de propósito general que puede contener tanto direcciones como datos

3. En una máquina little-endian con memoria de bytes y representación en complemento a dos que permite accesos a memoria de tamaño byte (1 B), media palabra (2 B) y palabra (4 B), se almacenan a partir de la posición 0xCAFEBA0 cuatro palabras con valores -1, -2, -3, -4. ¿Qué se obtendría al

consultar la media palabra de la posición 0xCAFEBAE?

- a. -1
  - b. -4
  - c. no se puede saber, faltan datos
  - d. ninguna de las anteriores
- 
- 4. Se pretende almacenar una palabra de 4 B en una memoria de bytes a partir de una dirección determinada. ¿Cuál de las siguientes es válida, si la palabra debe quedar alineada?
- a. 0xFACEB00C
**0xCAFEBAE**
- b. 0xDEADBEEF
0xABADDF00D
- c. 0xCAFEBABE
0xABADDF00D
- d. 0xABADDF00D
0xFACEB00C
- 
- 5. En una arquitectura de acumulador, la instrucción LOAD X:
- a. transfiere el contenido del registro X a la memoria
0xFACEB00C
- b. suma M(X) al acumulador
0xCAFEBAE
- c. transfiere el contenido del acumulador a la posición de memoria X
0xABADDF00D
- d. transfiere el contenido de la posición de memoria X al acumulador
0xFACEB00C
- 
- 6. Una instrucción máquina del tipo "Add M,R" podría formar parte del repertorio de
- a. una máquina pila
0xFACEB00C
- b. una máquina de acumulador
0xCAFEBAE
- c. una máquina con arquitectura R/R
0xABADDF00D
- d. una máquina con arquitectura M/M
0xFACEB00C
-

7. ¿Cuál de los siguientes **no** es un modo de direccionamiento IA-32?

- a. Registro
  - b. Memoria
  - c. Cache
  - d. Inmediato
- 

8. Un bus se compone de:

- a. líneas de datos y líneas de dirección
  - b. líneas de alimentación
  - c. líneas de estado y líneas de control
  - d. líneas de control/estado, líneas de dirección y líneas de datos
- 

9. ¿Cuál de los siguientes **no** es un tipo de bus?

- a. Secuencial
  - b. Paralelo
  - c. E/S
  - d. Sistema
- 

10. Si en un bus de direcciones de 32 bits se decodifica parcialmente la dirección de un dispositivo de 32 posiciones usando 22 bits, ¿cuántas veces aparecerá repetido en el mapa de memoria?

- a. 10
  - b. 16
  - c. 32
  - d. 1024
- 

11. Para obtener una única velocidad comparativa final, el benchmark SPEC CPU combina las velocidades de ejecución de una serie de tests, respecto a un ordenador de referencia, usando la media...

- a. aritmética
  - b. geométrica
  - c. armónica
  - d. ponderada
- 

12. El primer computador electrónico basaba su funcionamiento en:

- a. tubos de vacío
  - b. circuitos integrados LSI
  - c. amplificadores operacionales
  - d. núcleos de ferrita
- 

13. En Linux IA-32, si gcc usa la instrucción leave se puede asegurar que en ese punto del programa

a. correspondería emitir la secuencia de salida pop/ret, pero leave hace lo mismo y ocupa menos espacio

b. ya no hay registros salva-invocado que recuperar

c. ya no hay variables locales que destruir

d. ya no se hacen llamadas anidadas y por tanto no hay parámetros que ocupen espacio en pila

---

14. Usando el repertorio IA-32, para intercambiar el valor de 2 variables (por ejemplo A: .int 1 y B: .int 2) se pueden usar...

- a. dos instrucciones mov
  - b. una instrucción mov y una instrucción lea
  - c. 3 mov, no menos (se le llama "intercambio circular")
  - d. 4 mov, no menos (debido a la arquitectura R/M)
- 

15. Respecto a registros base e índice en IA-32, la excepción es que

- a. EBP no puede ser registro base
  - b. EBP no puede ser registro índice
  - c. ESP no puede ser registro base
  - d. ESP no puede ser registro índice
- 

16. El registro SP / ESP / RSP...

- a. es un registro transparente al usuario y contiene la instrucción que se está ejecutando
  - b. es un registro de propósito específico y contiene la dirección de la cima de la pila
  - c. es un registro transparente al usuario y contiene la dirección de memoria a la que se está accediendo
  - d. es un registro de propósito específico y contiene la dirección de la siguiente instrucción a ejecutar
- 

17. Diferencias gcc Linux IA-32/x86-64: marcar la respuesta **falsa**

- a. los enteros largos (long) pasan de 32 a 64 bits
  - b. los punteros (void\*) pasan de 32 a 64 bits
  - c. el tipo double pasa de 4 B a 8 B
  - d. long double pasa de 10/12 B a 16 B
- 

18. ¿Cuál de los siguientes fragmentos de código deja en %eax un resultado distinto a los otros tres fragmentos?

- a. mov \$-1, %edx  
sub %eax, %edx  
mov %edx, %eax
- b. not %eax  
add \$1, %eax
- c. xor %edx, %edx  
sub %eax, %edx  
mov %edx, %eax
- d. neg %eax

19. Si A y B son dos enteros almacenados respectivamente en %eax y %ebx, ¿cuál de las siguientes implementaciones de  
if (!A && !B) { ...then part... }  
es incorrecta?

- a. or %ebx, %eax  
jne not\_true  
...then part...  
not\_true:  
...  
b. cmp \$0, %eax  
jne not\_true  
cmp \$0, %ebx  
jne not\_true  
...then part...  
not\_true:  
...  
c. test %ebx, %eax  
jne not\_true  
...then part...  
not\_true:  
...  
d. test %eax, %eax  
jne not\_true  
test %ebx, %ebx  
jne not\_true  
...then part...  
not\_true:  
...

20. Dada la siguiente declaración en lenguaje C, una estructura de este tipo podría ocupar en un sistema Linux IA-32 o bien en uno x86-64 un total de...

```
struct a{  
    int i;  
    double d;  
    char c;  
    short s; };
```

- a. 18B  
b. 20B  
c. 22B  
d. 24B

21. En un sistema Linux x86-64, ¿cuál de las siguientes expresiones es equivalente a la expresión C  $(x[2] + 4)[3]$ ? Suponer que previamente se ha declarado int \*\*x.

- a.  $\ast((\ast(x + 16)) + 28)$   
b.  $\ast((\ast x) + 2) + 7)$   
c.  $(\ast(\ast(x + 2) + 4) + 3)$   
d.  $\ast((\ast(x + 2) + 4) + 3)$

22. Una unidad de control microprogramada se denomina "con secuenciamiento de microinstrucciones explícito" según tenga o no tenga

- a. ROM/PLA para traducir el codop en dirección de inicio de microprograma (goto f(IR))  
b. un multiplexor para seleccionar la fuente de la dirección de la memoria de control  
c. micro-contador de programa atacando a las líneas de dirección de la memoria de control  
d. microcódigo de decodificación que analice el codop bit a bit de izquierda a derecha

23. Dado un camino de datos concreto, un posible formato de microprogramación se caracteriza como horizontal o vertical según tenga más o menos (señalar la respuesta falsa)

- a. codificación  
b. solapamiento  
c. microbifurcaciones  
d. longitud relativa de microinstrucción

24. Motivos que impiden que la ganancia (aceleración) de un cauce segmentado sea ideal (señalar la respuesta falsa)

- a. registros de acople (coste de la segmentación)  
b. fragmentación desigual (duración desigual de etapas)  
c. riesgos (hazards)  
d. cola de instrucciones (precaptación)

25. La técnica de "adelanto de registros" (register forwarding) en un cauce segmentado se usa para limitar el impacto de los riesgos...

- a. estructurales  
b. organizativos  
c. de control  
d. (por dependencias) de datos

**26.** Las técnicas principales de E/S son (señalar la respuesta falsa)

- a. DMA (por acceso directo)
  - b. E/S programada
  - c. E/S cableada (hardwired)
  - d. IRQ (por interrupciones)
- 

**27.** Para determinar la causa de una interrupción se pueden usar las siguientes técnicas:  
(señalar la respuesta falsa)

- a. múltiples líneas de interrupción INT1#, INT2#...
  - b. línea de reconocimiento INTA#
  - c. consulta de estado, o polling
  - d. interrupciones vectorizadas
- 

**28.** Utilizar una cache en el mismo chip del procesador:

- a. aumenta el tamaño de los bloques enviados entre cache y procesador
  - b. reduce los tiempos de ejecución
  - c. reduce el tamaño del bus
  - d. aumenta la tasa de aciertos
- 

**29.** En un sistema Linux IA-32, ¿cuántos enteros se podrían almacenar en una línea de cache, si la cache del procesador fuera de 4 KB, asociativa por conjuntos de 4-vías, y contuviera 4 conjuntos?

- a. 16
  - b. 32
  - c. 64
  - d. 128
- 

**30.** La cache con correspondencia directa se puede considerar como un caso límite de la asociativa por conjuntos, en donde...

- a. solo hay 1 línea por conjunto
  - b. solo hay 1 palabra por bloque
  - c. solo hay 1 conjunto por cache
  - d. ninguna de las anteriores
-

**Nombre:**
**DNI:**
**Grupo:**

## Test de Prácticas (4.0p)

**Todas las preguntas son de elección simple sobre 4 alternativas.**

**Cada respuesta vale 4/20 si es correcta, 0 si está en blanco o claramente tachada, -4/60 si es errónea.**

**Anotar las respuestas (a, b, c o d) en la siguiente tabla.**

1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20

1. El switch de gcc para que únicamente compile de lenguaje C a ensamblador, y no realice ningún paso adicional (ensamblar, enlazar, etc), es...
  - a. -c
  - b. -S
  - c. -o
  - d. -g

---

2. Los switches --32 y --64 para trabajar en 32bit/64bit corresponden a la herramienta...
  - a. gcc
  - b. as
  - c. ld
  - d. nm

---

3. El switch -l para indicar librerías **\*NO\*** funciona con la herramienta...
  - a. gcc
  - b. as
  - c. ld
  - d. no se puede marcar una y solo una de las anteriores

---

4. ¿Cuál de las siguientes no es una sección de un fichero ELF?
  - a. .text
  - b. .static
  - c. .data
  - d. .bss

---

5. ¿Cuál de los siguientes contenidos no está incluido en un fichero ELF ejecutable?
  - a. código máquina
  - b. variables globales
  - c. pila del usuario
  - d. tabla de símbolos

---

6. En la práctica "media" se programa la suma de una lista de 32 enteros de 4 B para producir un resultado de 8 B, primero sin signo y luego con signo. Si la lista se rellena con el valor que se indica a continuación, ¿en qué caso ambos programas producen el mismo resultado?
  - a. 0x1111 1111
  - b. 0x9999 9999
  - c. 0xAAAA AAAA
  - d. 0xFFFF FFFF

---

7. En la práctica "media" se programa la suma de una lista de 32 enteros de 4 B para producir un resultado de 8 B, primero sin signo y luego con signo. Si la lista se rellena con el valor 0x0400 0000, ¿en qué se diferencian los resultados de ambos programas?
  - a. no se diferencian
  - b. en uno ocupa 32 bits, en otro 64 bits
  - c. en uno se interpreta como negativo, en otro como positivo
  - d. en uno los 32 bits superiores son 0xFFFF FFFF, en el otro no

8. En la práctica "media" se suma una lista de 32 enteros de 4 B con signo para producir una media y un resto usando la instrucción IDIV. ¿Cuál de las siguientes afirmaciones es falsa?

- a. IDIV produce el mismo cociente que el operador / en lenguaje C
- b. IDIV produce el mismo resto que el operador % en lenguaje C
- c. La media se redondea al entero más próximo
- d. El resto siempre tiene el mismo signo que la suma

9. En la práctica "media" un estudiante usa el siguiente bucle para acumular la suma en EBP:EDI antes de calcular la media y el resto

bucle:

```
    mov (%ebx,%esi,4), %eax
    cltd
    add %eax, %edi
    adc %edx, %ebp
    jnc nocarry
    inc %edx
nocarry:
    inc %esi
    cmp %esi,%ecx
    jne bucle
```

Estando bien programado todo lo demás, este código

- a. produce siempre el resultado correcto
- b. fallaría con lista: .int 0,1,2,3
- c. fallaría con lista: .int -1,-2,-4,-8
- d. no siempre produce el resultado correcto, pero el error no se manifiesta en los ejemplos propuestos, o se manifiesta en ambos

10. Alguno de los siguientes no es un nombre de registro en una máquina IA-32 en modo 32 bits

- a. ebp
- b. ax
- c. dh
- d. sil

11. Alguno de los siguientes no es un nombre de registro en una máquina x86-64 en modo 64 bits

- a. r8d
- b. r12w
- c. sih
- d. spl

12. Para comprobar si el entero almacenado en EAX es cero (y posiblemente saltar a continuación usando JZ/JNZ), gcc genera el código

- a. cmp %eax, \$0
- b. test %eax
- c. cmp %eax
- d. test %eax, %eax

13. La práctica "paridad" debía calcular la suma de paridades impar (XOR de todos los bits) de los elementos de un array. Un estudiante entrega la siguiente versión de parity3:

```
int parity3(unsigned* array,
            int len){
    int i,res=0,val;
    unsigned x;
    for (i=0; i<len; i++){
        x=array[i];
        val=0;
        do {
            val += x;
            x >>= 1;
        } while (x);
        val &= 0x1;
        res+=val;
    }
    return res;
}
```

Esta función parity3:

- a. produce siempre el resultado correcto
- b. fallaría con array={0,1,2,3}
- c. fallaría con array={1,2,4,8}
- d. no siempre produce el resultado correcto, pero el error no se manifiesta en los ejemplos propuestos, o se manifiesta en ambos

14. Un estudiante entrega la siguiente versión de parity4:

```
int parity4(unsigned* array,
            int len){
    int val,i,res=0;
    unsigned x;
    for (i=0; i<len; i++){
        x=array[i];
        val=0;
        asm("\n"
"ini3:           \n\t"
"    xor  %[x],%[v] \n\t"
"    shr  %[x]         \n\t"
"    test %[x], %[x]\n\t"
"    jne   ini3      \n\t"
":[v]"+"r" (val)
:[x] "r" (x)
);
        val = val & 0x1;
        res+=val;
    }
    return res;
}
```

Esta función parity4:

- a. produce siempre el resultado correcto
- b. fallaría con array={0,1,2,3}
- c. fallaría con array={1,2,4,8}
- d. no siempre produce el resultado correcto, pero el error no se manifiesta en los ejemplos propuestos, o se manifiesta en ambos

15. La sentencia asm() del listado anterior tiene las siguientes restricciones

- a. ninguna
- b. arquitectura de 32 bits
- c. dos entradas y una salida
- d. un registro y dos sobreescritos (clobber)

16. Un estudiante entrega la siguiente versión de parity5:

```
int parity5(unsigned* array,
            int len){
    int i,j,res=0;
    unsigned x;
    for (i=0; i<len; i++){
        x=array[i];
        for (j=sizeof(unsigned)*4;
```

```
                j>0; j=j/2){
            x^=x>>j;
        }
        x = x & 0x1;
        res+=x;
    }
    return res;
}
```

Esta función parity5:

- a. produce siempre el resultado correcto
- b. fallaría con array={0,1,2,3}
- c. fallaría con array={1,2,4,8}
- d. no siempre produce el resultado correcto, pero el error no se manifiesta en los ejemplos propuestos, o se manifiesta en ambos

17. Un estudiante entrega la siguiente versión de parity6:

```
int parity6(unsigned* array,
            int len){
    int i,j,res=0;
    unsigned x;
    for (i=0; i<len; i++){
        x=array[i];
        asm("\n"
"    mov  %[x],%%edx \n\t"
"    shr $16, %%edx \n\t"
"    xor %%edx,%[x] \n\t"
"    mov  %[x],%%edx \n\t"
"    mov  %%dh, %%dl \n\t"
"    xor %%edx, %[x]\n\t"
"    setpo %%cl      \n\t"
"    movzx %%cl, %[x]"
:[x] "+r" (x)
:
:"edx","ecx"
);
        res+=x;
    }
    return res;
}
```

Esta función parity6:

- a. produce siempre el resultado correcto
- b. fallaría con array={0,1,2,3}
- c. fallaría con array={1,2,4,8}

- d. no siempre produce el resultado correcto, pero el error no se manifiesta en los ejemplos propuestos, o se manifiesta en ambos
- 

**18.** La sentencia asm() del listado anterior tiene las siguientes restricciones

- a. ninguna
  - b. arquitectura de 32 bits
  - c. dos entradas y una salida
  - d. un registro y dos sobreescritos (clobber)**
- 

**19.** En el programa "size" de la práctica de la cache, si el primer escalón pasa de tiempo = 1 para todos los tamaños de vector menores o iguales que 32 KB a tiempo = 3 para los tamaños 64 KB y 128 KB, podemos asegurar que:

- a. la cache L1 es al menos tres veces más rápida que la cache L2.**
  - b. la cache L1 es como mucho tres veces más rápida que la cache L2.
  - c. la cache L2 es al menos el doble de rápida que la memoria principal.
  - d. la cache L2 es como mucho el doble de rápida que la memoria principal.
- 

**20.** El código del programa "size" de la práctica de la cache accede al vector saltando...

- a. de byte en byte.
  - b. de 64 en 64 bytes.**
  - c. de 1 KB en 1 KB.
  - d. de 64 KB en 64 KB.
-

Nombre:	
DNI:	Grupo:

### Examen de Problemas (3.0 p)

1. **Ensamblador a C** (0.6 puntos). Dada la siguiente función en ensamblador x86-64:

```

loop:
# a la entrada: a en %rdi, n en %esi
    movl    $0, %r8d
    movl    $0, %ecx
    testl  %esi, %esi
    jle .L3
.L6:
    movl    (%rdi,%rcx,4), %edx
    leal    3(%rdx), %eax
    testl  %edx, %edx
    cmovns %edx, %eax
    sarl    $2, %eax
    addl    %eax, %r8d
    addq    $1, %rcx
    cmpl    %ecx, %esi
    jg     .L6
.L3:
    movl    %r8d, %eax
    ret

```

Rellenar los huecos en el código C correspondiente.

Se debe usar sintaxis de indexación en arrays para acceder a los elementos de **a**.

Naturalmente, no se deben usar nombres de registros x86-64 en código C.

Si se duda sobre la equivalencia aritmética de la operación de desplazamiento, expresarla también como desplazamiento en lenguaje C.

```

int loop(int a[], int n)
{
    int i, sum;
    sum = ____;
    for (i = _____; _____; _____) {
        sum += _____;
    }
    return _____;
}

```

2. **Acceso a estructuras** (0.5 puntos). Sean las siguientes declaraciones de estructuras en C:

```

struct data {
    long x;
    char str[16];
};

struct node {
    struct data d;
    struct node *next;
};

```

Abajo se ofrecen cinco funciones C y cinco bloques de código x86-64. Anotar en cada bloque x86-64 el nombre de la función C que implementa.

<code>int alpha(struct node *ptr){     return ptr-&gt;d.x; }</code>	_____	<code>movsb 15(%rdi),%eax ret</code>
<code>char *beta(struct node *ptr){     ptr = ptr-&gt;next;     return ptr-&gt;d.str; }</code>	_____	<code>movq (%rdi), %rax ret</code>
<code>char gamma(struct node *ptr){     return ptr-&gt;d.str[7]; }</code>	_____	<code>movq 24(%rdi), %rax addq \$8, %rax ret</code>
<code>long *delta(struct node *ptr){     struct data *dp =         (struct data *) ptr;     return &amp;dp-&gt;x; }</code>	_____	<code>movq %rdi, %rax ret</code>
<code>char *epsilon(struct node *ptr){     return &amp;ptr-&gt;d.str[2]; }</code>	_____	<code>leaq 10(%rdi), %rax ret</code>

3. **Acceso a matrices.** (0.5 puntos). Hemos escrito un archivo **f.c** que contiene una función **f** escrita en lenguaje C, a la que se le pasa la dirección de una matriz cuadrada de **N×N** números. La función **f** suma los elementos de la diagonal de la matriz.

```
#define N ...
typedef ... number;
int f (number v[N][N]) {
    ...
}
```

Compilamos dicha función mediante la siguiente orden:  
`gcc f.c -m32 -O1 -S -fno-omit-frame-pointer`  
obteniendo un archivo **f.s** con el siguiente contenido:

```
f:
    pushl %ebp
    movl %esp, %ebp
    pushl %ebx
    movl 8(%ebp), %ebx
    movl %ebx, %edx
    addl $10100, %ebx
    movl $0, %eax
```

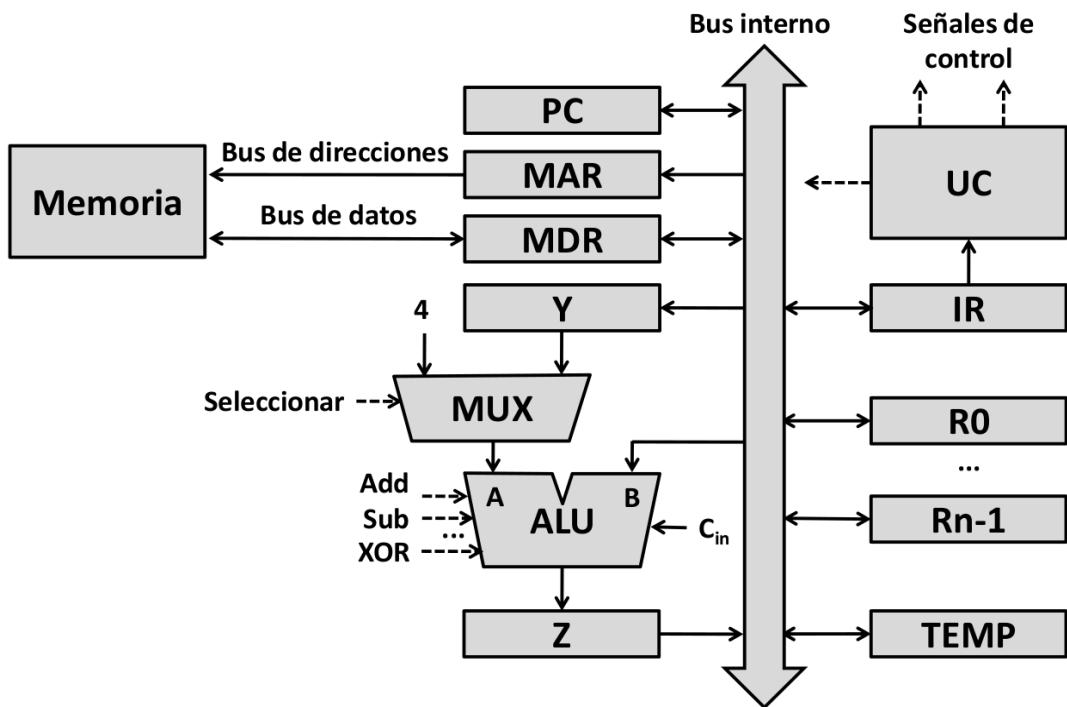
.L2:

```
movsb    (%edx), %ecx
addl    %ecx, %eax
addl    $101, %edx
cmpl    %ebx, %edx
jne     .L2
popl    %ebx
popl    %ebp
ret
```

- a) ¿Qué registros son modificados en el cuerpo de la función f?
- b) ¿Qué registros modificados se guardan en la pila y cuáles no, y por qué?
- c) ¿Cuánto vale N?
- d) ¿De qué tipo son los elementos de la matriz?
- e) ¿Dónde devuelve el resultado la función?

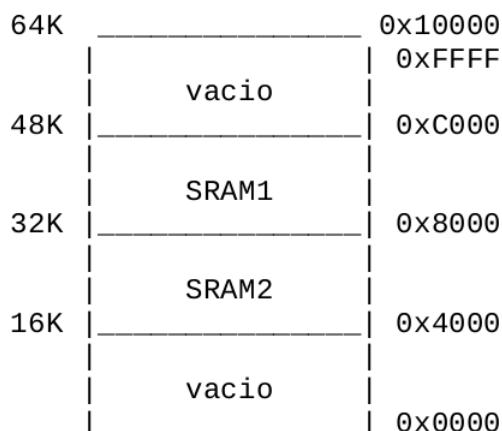
Escriba la función f completa en C (contenido del archivo f.c)

4. **Unidad de control** (0.5 puntos). Exprese en lenguaje RTL (transferencia entre registros) o pseudocódigo las operaciones elementales que se producen en cada uno de los ciclos de la ejecución de la instrucción de una palabra BRANCH desplazamiento, incluyendo la fase de captura de instrucción (*fetch*), para la siguiente microarquitectura:



La lectura de memoria se realiza en un solo ciclo. El bus interno y todos los registros tienen un tamaño de  $n$  bits. El desplazamiento es un número con signo relativo al contador de programa, ocupará los  $m$  bits menos significativos ( $m < n$ ) del registro IR cuando la instrucción pase a dicho registro, y la salida del registro IR hacia el bus interno es ese campo desplazamiento con el signo extendido a  $n$  bits.

5. **Diseño de memoria** (0.5 puntos). Un sistema basado en un pequeño microprocesador (con un bus de datos de 8 bits, un bus de direcciones de 16 bits y una patilla R/W#) dispone del siguiente mapa de memoria:



Dibuje la decodificación y conexionado de los dos módulos de memoria SRAM1 y SRAM2 al microprocesador.

6. **Memoria cache** (0.4 puntos). Sea un computador de 32 bits con una memoria cache de datos asociativa por conjuntos de 2 vías con líneas de 64 bytes y un tamaño total de 32 KB, y que emplea la política de reemplazo LRU.
- Indique el número de líneas y de conjuntos de la cache.
  - Dado el fragmento de código:

```
int v[1048576];
for (i = 0; i < 1048576; i += 4)
    v[i] = i;
```

y considerando exclusivamente los accesos al vector, calcule y razoné la tasa de fallos que se obtiene en dicha cache de datos en la ejecución del bucle anterior.

### Test de Teoría (3.0p)

<b>1</b>	<b>2</b>	<b>3</b>	<b>4</b>	<b>5</b>	<b>6</b>	<b>7</b>	<b>8</b>	<b>9</b>	<b>10</b>	<b>11</b>	<b>12</b>	<b>13</b>	<b>14</b>	<b>15</b>	<b>16</b>	<b>17</b>	<b>18</b>	<b>19</b>	<b>20</b>	<b>21</b>	<b>22</b>	<b>23</b>	<b>24</b>	<b>25</b>	<b>26</b>	<b>27</b>	<b>28</b>	<b>29</b>	<b>30</b>
a	b	a	a	d	d	c	d	a	c	b	a	b	d	d	b	c	a	c	d	d	c	c	d	d	c	b	b	c	a

### Test de Prácticas (4.0p)

<b>1</b>	<b>2</b>	<b>3</b>	<b>4</b>	<b>5</b>	<b>6</b>	<b>7</b>	<b>8</b>	<b>9</b>	<b>10</b>	<b>11</b>	<b>12</b>	<b>13</b>	<b>14</b>	<b>15</b>	<b>16</b>	<b>17</b>	<b>18</b>	<b>19</b>	<b>20</b>
b	b	b	b	c	a	a	c	a	d	c	d	a	a	c	a	a	d	a	b

### Examen de Problemas (3.0p)

1. Ensamblador a C (0.6 puntos).

```
int loop(int a[], int n)
{
    int i, sum;
    sum = 0;
    for (i = 0; i<n; i++) {
        sum += a[i]/4;
    //alter.sum += (a[i]>0? a[i] : a[i]+3) >> 2;
    }
    return sum;
}
```

2. Acceso a estructuras (0.5 puntos).

En orden: **gamma, alpha, beta, delta, epsilon**

3. Acceso a matrices (0.5 puntos).

- a) **%ebp, %ebx, %edx, %eax, %ecx** (%ebp en ajuste marco pila, %esp, %eip implícitamente)
- b) **%ebp, %ebx**
- c) **100**
- d) **char**
- e) **%eax**

f.c

```
#define N 100
typedef char number;

int f(number v[N][N]){
    int i, sum=0;
    for (i=0; i<N; i++) sum += v[i][i];
    return sum;
}
```

#### 4. Unidad de control (0.5 puntos).

Solución 1:

```

fetch: MAR:=PC; Z:=PC+4;
PC:=Z; Y:=Z; MDR:=M[MAR];
IR:=MDR;
goto f(IR);

branch: Z:=Y+IR;
PC:=Z; goto fetch;

```

Solución 2:

```

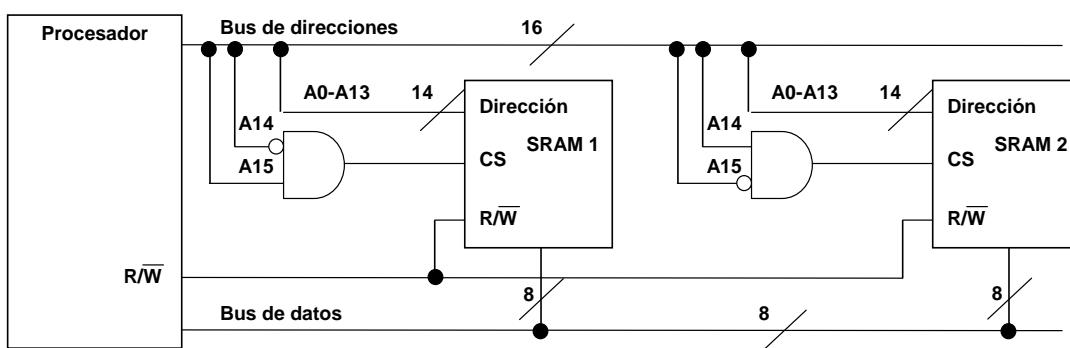
fetch: MAR:=PC; Z:=PC+4;
PC:=Z; MDR:=M[MAR];
IR:=MDR;
goto f(IR);

branch: Y:=PC;
Z:=Y+IR;
PC:=Z; goto fetch;

```

#### 5. Diseño de memoria (0.5 puntos).

La solución es el enunciado del problema 5 del examen de febrero de 2015



#### 6. Memoria cache (0.4 puntos).

- a)  $32 \text{ KB} = 2^{15}$  bytes.

Como cada línea tiene  $2^6$  bytes, el nº de líneas es  $(2^{15} \text{ bytes}) / (2^6 \text{ bytes/línea}) = 2^9$  líneas = **512 líneas**.

Como la cache es asociativa por conjuntos de 2 vías, cada conjunto tiene 2 líneas, por tanto el número de conjuntos es  $512 / 2 = 256$  conjuntos.

- b) **F=1/4**

64 bytes/línea:  $2^6 / 2^2 = 2^4$  ints/línea, y como se avanza de 4 en 4 ints, hay **un fallo cada  $2^4 / 2^2 = 4$  accesos**

| F | \_ | \_ | A | \_ | \_ | A | \_ | \_ | A | \_ | \_ | F ...

**Nombre:**
**DNI:**
**Grupo:**

### Test de Teoría (3.0p)

**Todas las preguntas son de elección simple sobre 4 alternativas.**

**Cada respuesta vale 0.1p si es correcta, 0p si está en blanco o claramente tachada, -0.03p si es errónea.**

**Anotar las respuestas (a, b, c ó d) en la siguiente tabla.**

1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30

1. ¿Cuál de las siguientes afirmaciones sobre el benchmark SPEC CPU es falsa?
  - a. La última versión es SPEC CPU2006 V1.2 de 2011
  - b. Se cronometran unos 12 tests de enteros (CINT2006) y unos 17 tests de punto flotante (CFP2006)
  - c. Se usa como referencia un computador UltraSPARC II 300MHz, y para cada test se calcula el cociente entre el tiempo de ejecución en el computador a testear y en el de referencia
  - d. El resultado final es la media aritmética de las (12 ó 17) velocidades, bien sea de enteros ó de punto flotante (SPECint2006 ó SPECfp2006)

---

2. ¿En qué generación, dentro de la historia de los computadores digitales, aparecieron la microprogramación, la segmentación de cauce, la memoria cache, los S.O. multiusuario y la memoria virtual?
  - a. 2ª generación (1955-65)
  - b. 3ª generación (1965-75)
  - c. 4ª generación (1975-85)
  - d. esas innovaciones se repartieron a lo largo de varias generaciones, no sólo una

---

3. Respecto a tamaños de tipos integrales en x86 y x86-64, la excepción es que
  - a. int pasa de 4 B (x86) a 8 B (x86-64)
  - b. long int pasa de 4 B a 8 B
  - c. long long int pasa de 4 B a 8 B
  - d. ninguna de las anteriores

---

4. Con el repertorio IA32, para sumar %eax y %ebx dejando el resultado en %ecx se podría hacer lo siguiente:
  - a. lea %eax, %ebx, %ecx
  - b. lea (%eax, %ebx, 1), %ecx
  - c. lea %ecx, [%eax, %ebx]
  - d. lea %ecx, %ebx, %eax

---

5. Cuál de las instrucciones máquina siguientes es incorrecta en x86-64:
  - a. testl %edx, %edx
  - b. movl %r8, %eax
  - c. movl (%rdi,%rcx,4), %edx
  - d. addq \$1, %rcx

---

6. Si la variable val está almacenada en ebx y la variable x está almacenada en eax, la sentencia **val ^= x;** se puede traducir a ensamblador como:
  - a. xorl %ebx,%eax
  - b. xorl %eax,%ebx
  - c. andl %ebx,%eax
  - d. testl %eax,%ebx

---

7. Para poner a 1 el bit 5 del registro %edx sin cambiar el resto de bits podemos usar la instrucción máquina:
  - a. and \$32, %edx
  - b. and \$0x5, %edx
  - c. or \$0b101, %edx
  - d. or \$0x20, %edx

---

8. Si tenemos un número  $n$ , de 64 bits, almacenado en la pareja de registros EDX:EAX (EDX contiene los 32 bits más

significativos y EAX los 32 bits menos significativos) y queremos realizar la división  $n / 2^{32}$  entonces:

- a. Podemos quedarnos con EDX, pero sólo en el caso de que  $n$  sea un número sin signo.
- b. Podemos quedarnos con EDX tanto si  $n$  es un número con signo como sin signo.
- c. Podemos usar las instrucciones siguientes, pero sólo en el caso de que  $n$  sea un número sin signo:

```
mov $0x10000000,%ecx  
div %ecx
```

- d. Podemos usar las instrucciones siguientes tanto si  $n$  es un número con signo como sin signo:

```
mov $0x10000000,%ecx  
div %ecx
```

- 
9. ¿Dónde está ubicado el primer argumento a una función (suponer código ensamblador cdecl generado por gcc para Linux/x86) inmediatamente después de ejecutar la instrucción call?

- a. %ebp + 0x4
- b. %ebp - 0x4
- c. %esp + 0x4
- d. %esp - 0x4

- 
10. Dado el código C siguiente:

```
struct data {  
    char str[16];  
};  
char *f(struct data *ptr) {  
    return &(ptr->str[2]);  
}
```

la función se traducirá a ensamblador de x86-64 como:

- a. leaq 2(%rdi), %rax  
ret
- b. movq (%rdi,2), %rax  
ret
- c. movq 2(%rdi), %rax  
ret
- d. leaq (%rdi,2), %rax  
ret

- 
11. Respecto a requisitos de alineamiento de structs en gcc/IA32 x86 y x86-64, alguna de las siguientes afirmaciones es falsa

- a. en x86 Linux alinea double a 4x (Windows no)
- b. en x86 Linux alinea long double a 4x (Windows también)
- c. en x86-64 Linux alinea double a 8x (Windows también)
- d. en x86-64 Linux alinea float a 8x (Windows también)

- 
12. Si la estructura **struct a** ocupa un espacio de 28 bytes en memoria, ¿cuántos bytes ocupa la siguiente estructura **struct b** cuando se compila en 64 bits?

```
struct b {  
    struct a a1;  
    int i;  
    struct a a2;  
};
```

- a. 24 bytes
- b. 60 bytes
- c. 64 bytes
- d. 84 bytes

- 
13. Respecto a los términos microinstrucción y microcódigo:

- a. Son equivalentes, llamamos microcódigo o microinstrucción a una palabra de la memoria de control
- b. Una microinstrucción está programada en microcódigo, que es un lenguaje para programar señales de control
- c. Un microcódigo controla una serie de señales de control relacionadas (por ejemplo, el código 000 para que la ALU realice la suma), y varios microcódigos juntos forman una microinstrucción
- d. Microcódigo es el contenido de la memoria de control, y una microinstrucción es una palabra de dicha memoria

- 
14. ¿Cuál de las siguientes afirmaciones es verdadera?

- a. La unidad de control necesita como entrada el registro de estado para poder controlar la ejecución de las instrucciones de salto condicional.
- b. El registro de instrucción es un registro de propósito específico que contiene la dirección de la siguiente instrucción a ejecutar.

- c. Las únicas instrucciones en las que algunas de sus fases de ejecución conllevan un acceso a memoria son las instrucciones load y store.
- d. El registro puntero de pila es un registro de propósito general que suele contener tanto direcciones como datos.

15. Un procesador con una unidad de control micropogramada tiene una memoria de control de 300 palabras de 100 bits, de las que 200 son diferentes. Si se rediseñara como unidad de control nanoprogramada, ¿qué tamaño ocuparía la nanomemoria que contiene las microinstrucciones completas sin repeticiones?

- a. 20000 bits
- b. 21600 bits
- c. 22400 bits
- d. 30000 bits

16. En el pseudocódigo usado para representar las microinstrucciones, la expresión “goto f(IR)”:

- a. Se utiliza para realizar un microsalto condicional en función del registro de estado.
- b. Realiza una llamada a una microsubrutina.
- c. Salta a una dirección de memoria de control que depende de la instrucción máquina actual.
- d. Permite saltar a la dirección de memoria de control del principio de un microbucle.

17. Respecto a la predicción de saltos, alguna de las siguientes afirmaciones es falsa

- a. si se toma la misma decisión para cada tipo de instrucción, se trata de "predicción estática"
- b. si la predicción cambia según la historia de ejecución del programa, se trata de "predicción dinámica"
- c. para predicción estática, es conveniente decidir que los saltos hacia adelante siempre se cumplen, y hacia atrás no
- d. para predicción dinámica, existen, entre otros, algoritmos de dos o cuatro estados, que requieren 1 o 2 bits por instrucción

18. Respecto a los conceptos de procesamiento segmentado y superescalar, una de las siguientes afirmaciones es falsa

- a. idealmente, con el segmentado se intenta ejecutar una instrucción por ciclo, y con el superescalar más de una por ciclo (al combinarlo con segmentado)
- b. en cualquier procesador resulta ventajoso usar una cola de instrucciones, pero es más importante para uno segmentado (fundamental) que para uno superescalar (conveniente)
- c. por definición, un procesador superescalar debe tener varias unidades funcionales (más de una)
- d. implícitamente, se presupone que un procesador superescalar emitirá más de una instrucción por ciclo

19. Respecto a los conceptos de interfaz de dispositivo, controlador(a), puerto de E/S:

- a. La controladora o interfaz contiene los puertos necesarios para utilizar el dispositivo
- b. Cada puerto o interfaz es una línea de comunicación con el procesador. El conjunto de ellos forma el controlador.
- c. El puerto, o interfaz, contiene los controladores necesarios para comunicar el dispositivo con el procesador
- d. El interfaz contiene las controladoras necesarias para conectar los puertos con el procesador

20. Respecto a los conceptos de procesador de E/S, canal de E/S, dispositivos de E/S:

- a. Un procesador o canal tiene un repertorio de instrucciones específico para manejar los dispositivos E/S
- b. Cada canal es una línea de comunicación entre el procesador y un dispositivo de E/S.
- c. Al conjunto de conexiones entre el procesador y los dispositivos se le denomina canal de E/S (de ese ordenador)
- d. La pregunta es capciosa, el procesador no es E/S, son otros dos componentes von Neumann distintos (ALU+UC)

21. La E/S programada:

- a. Mejora las prestaciones globales del sistema respecto a la E/S por interrupciones porque la CPU tiene el control de toda la operación.
- b. Mejora las prestaciones globales del sistema respecto a la E/S por interrupciones porque la CPU es más rápida que el controlador de interrupciones y la interfaz del periférico.
- c. Empeora las prestaciones globales del sistema respecto a la E/S por interrupciones porque una

instrucción de transferencia individual de datos con la interfaz del periférico (por ej. IN, OUT) es más lenta en E/S programada que en E/S por interrupciones.

- d. Empeora las prestaciones globales del sistema respecto a la E/S por interrupciones porque la CPU debe encargarse de la sincronización con la interfaz del periférico haciendo una espera activa.

22. Una puerta AND con 16 entradas conectada a un bus de direcciones de 16 bits, con todos los bits negados excepto A10 y A6, permite seleccionar un dispositivo (con CS activa en alta) en la dirección:

- a. 0xFDDF  
b. 0xFBFF  
c. 0x0220  
d. 0x0440

23. Un computador con 15 líneas de direcciones tiene 3 módulos de memoria de  $2^{13}$  palabras y utiliza E/S mapeada en memoria. ¿Cuál es el número máximo de periféricos que pueden conectarse, si cada uno de ellos utiliza 8 direcciones?

- a.  $2^{10}$   
b.  $2^{12}$   
c.  $2^{11}$   
d.  $2^{13}$

24. Un procesador accede en el instante de tiempo t a una posición de memoria d(t). Poco tiempo después (en el instante de tiempo t+k) accede a la posición anterior d(t)-1. Esos dos accesos son un ejemplo de...

- a. Localidad espacial  
b. Localidad temporal  
c. No tiene nombre, ese tipo de localidad con incremento negativo (d(t)-1) no se ha estudiado en clase  
d. No es una localidad, esa condición no guarda relación con el concepto de localidad

25. Una jerarquía de memoria consta de una cache de con una tasa de aciertos del 92% y 4 ns de tiempo de acceso y una memoria principal con una tasa de aciertos del 100% y 100 ns de tiempo de acceso. ¿Cuál es el tiempo promedio estimado de acceso a memoria?

- a. 6 ns

- b. 8 ns  
c. 10 ns  
d. 12 ns

26. Una SRAM de 1Mx4bit (4Mbit) puede venir organizada en 2048 filas, dedicando por tanto al decodificador de columnas...

- a. 6 bits  
b. 7 bits  
c. 8 bits  
d. 9 bits

27. Un sistema basado en un microprocesador con un bus de datos de n bits y un bus de direcciones de 16 bits direcciona la memoria por palabras de n bits y dispone de una memoria SRAM formada por dos módulos de 16 K x n cada uno. ¿Qué porcentaje del mapa de memoria está ocupado por la SRAM?

- a. 12,5%  
b. 25%  
c. 50%  
d. 100%

28. Un módulo de memoria de 16 GB está formado por varios chips DRAM de 1024Mx4. ¿Cuántos chips DRAM necesita el módulo?

- a. 4  
b. 8  
c. 16  
d. 32

29. Una cache de 256 B asociativa por conjuntos de 4-vías con líneas de 16 B tendría

- a. 4 conjuntos  
b. 16 conjuntos  
c. 64 conjuntos  
d. ningún conjunto

30. En un sistema con memoria de bytes, ¿cuál sería el tamaño de una línea de cache, si la cache del procesador fuera de 4MB, asociativa por conjuntos de 16-vías, y contuviera 4096 conjuntos?

- a. 16 B  
b. 32 B  
c. 64 B  
d. 128 B

Nombre:	
DNI:	Grupo:

## Test de Prácticas (4.0p)

**Todas las preguntas son de elección simple sobre 4 alternativas.**

**Cada respuesta vale 0.2p si es correcta, 0 si está en blanco o claramente tachada, -0.06p si es errónea.**  
**Anotar las respuestas (a, b, c ó d) en la siguiente tabla.**

1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20

1. ¿Qué hace gcc -O0?
- a. Compilar sin optimización
  - b. Compilar .c→.o (fuente C a objeto)
  - c. Compilar .s→.o (fuente ASM a objeto)
  - d. Compilar .c→.s (C→ASM sin generar objeto)
- 
2. ¿Qué modificador (switch) de gcc hace falta para compilar una aplicación de 32 bits en un sistema de 64 bits?
- a. -m32
  - b. -m64
  - c. -march32
  - d. -march64
- 
3. La etiqueta del punto de entrada a un programa ensamblador en el entorno de las prácticas 1 a 4 (GNU/as Linux x86) es:
- a. \_main
  - b. .L0
  - c. \_start
  - d. \_init
- 
4. La siguiente línea en la sección de datos de un programa en ensamblador de IA32
- ```
result: .int 0,0
```
- a. Reserva espacio para un único entero, inicializado a 0,0
  - b. Reserva espacio para un entero, inicializado a 0, seguido de un dato de tamaño indefinido, también inicializado a 0
  - c. Reserva espacio para dos enteros, inicializados ambos a 0
- 
- d. Reserva espacio para un único entero, inicializado a 0, en la posición de memoria 0
5. El volcado mostrado abajo se ha obtenido con el comando...
- ```
00000000 <main>:
 0: 55                      push %ebp
 1: 89 e5                   mov %esp,%ebp
 3: 83 e4 f0                and $-16,%esp
 6: 83 ec 10                sub $0x10,%esp
 9: c7 44 24 04 03          movl $3, 4(%esp)
 e: 00 00 00
 11: c7 04 24 01 00         movl $0x1,(%esp)
 16: 00 00
 18: e8 fc ff ff ff        call <main+0x19>
 1d: c9                     leave
 1e: c3                     ret
```
- a. objdump -h p
  - b. objdump -d p
  - c. objdump -d p2.o
  - d. objdump -t p2.o
- 
6. En la práctica "media" se desea invocar desde lenguaje ensamblador la función printf() de libC. Eso implica que este programa, como todo programa que use libC,
- a. es obligatorio que contenga main (y entonces es más cómodo usar gcc para enlazar)
  - b. es obligatorio enlazarlo usando gcc (y entonces es más cómodo que contenga main)
  - c. es ventajoso para ensamblarlo que contenga main, y entonces es conveniente enlazarlo usando gcc (aunque ambas cosas son opcionales)

- d. es ventajoso para enlazarlo usar gcc, y entonces es conveniente que contenga main (aunque ambas cosas son opcionales)

7. En la práctica "media" se pide sumar una lista de 32 enteros SIN signo de 32 bits en una plataforma de 32 bits sin perder precisión, esto es, evitando perder acarreos. Un estudiante entrega la siguiente versión

```
# $lista en EBX, longlista en ECX
suma:
```

```
    mov $0, %eax
    mov $0, %edx
    mov $0, %esi

bucle:
    add (%ebx,%edx,4), %eax
    jnc seguir
    inc %edx

seguir:
    inc %esi
    cmp %esi,%ecx
    jne bucle
    ret
```

Esta función presenta una única diferencia frente a la solución recomendada en clase, relativa al indexado en el array.

Esta función suma:

- a. Produce siempre el resultado correcto  
b. Fallaría con lista: .int 1,1,1,1, 1,1,1,1, ...  
c. Fallaría con lista: .int 1,2,3,4, 1,2,3,4, ...  
d. No es correcta pero el error no se manifiesta en los ejemplos propuestos, o se manifiesta en ambos

8. En la práctica "media" se pide sumar una lista de 32 enteros SIN signo de 32 bits en una plataforma de 32 bits sin perder precisión, esto es, evitando perder acarreos. De entre los siguientes, ¿cuál es el mínimo valor entero que repetido en toda la lista causaría acarreo con 32 bits (sin signo)? Se usa notación decimal y espacios como separadores de millares/millones/etc.

- a. 10 000 000  
b. 100 000 000  
c. 1 000 000 000  
d. 10 000 000 000

9. En la práctica "media" se pide sumar una lista de 32 enteros CON signo de 32 bits en una plataforma de 32 bits sin perder precisión, esto es, evitando desbordamiento. De entre los

siguientes, ¿cuál es el valor negativo más pequeño (en valor absoluto) que repetido en toda la lista causaría desbordamiento con 32 bits (en complemento a 2)? Se usa notación decimal y espacios como separadores de millares/millones/etc.

- a. -10 000 000
b. -100 000 000
c. -1 000 000 000
d. -10 000 000 000

10. ¿Cuál es el popcorn (peso Hamming, nº de bits activados) del número 19?

- a. 2
b. 3
c. 4
d. 5

11. La práctica "popcount" debía calcular la suma de bits (peso Hamming) de los elementos de un array. Un estudiante entrega la siguiente versión de popcorn3:

```
int popcorn3(unsigned* array,
             int len){
    int i, res = 0;
    unsigned x;
    for( i = 0; i < len; i++ ) {
        x = array[i];
        asm("ini3:          \n"
            "shr %[x]         \n"
            "adc $0, %[r]      \n"
            "add $0, %[x]      \n"
            "jne ini3          \n"
            : [r] "+r" (res)
            : [x] "r" (x) );
    }
    return res;
}
```

Esta función sólo tiene una diferencia con la versión "oficial" recomendada en clase. En concreto, una instrucción máquina en la sección **asm()** es distinta.

Esta función popcorn3:

- a. produce siempre el resultado correcto  
b. fallaría con array={0,1,2,3}  
c. fallaría con array={1,2,4,8}  
d. no es correcta pero el error no se manifiesta en los ejemplos propuestos, o se manifiesta en ambos

12. La práctica "popcount" debía calcular la suma de bits (peso Hamming) de los elementos de un array. Un estudiante entrega la siguiente versión de `popcount3`:

```
int popcount3(int* array, int len){  
    long val = 0;  
    int i;  
    unsigned x;  
    for (i=0; i<len; i++){  
        x= array[i];  
        do{  
            val += x & 0x1;  
            x >>= 1;  
        } while (x);  
        val += (val >> 16);  
        val += (val >> 8);  
    }  
    return val & 0xFF;  
}
```

Esta función es una mezcla inexplicada de las versiones "oficiales" de `popcount2` y `popcount4`, incluyendo diferencias en 2 tipos de datos, la ausencia de la variable `res` y la diferente posición de la máscara `0xFF`.

Esta función `popcount3`:

- a. produce siempre el resultado correcto
- b. fallaría con `array={0,1,2,3}`
- c. fallaría con `array={1,2,4,8}`
- d. no es correcta pero el error no se manifiesta en los ejemplos propuestos, o se manifiesta en ambos

13. La práctica "popcount" debía calcular la suma de bits (peso Hamming) de los elementos de un array. Un estudiante entrega la siguiente versión de `popcount4`:

```
int popcount4(unsigned* array,  
              int len){  
    int i,j;  
    unsigned x;  
    int result = 0;  
    for(i=0;i<len;i++){  
        x=array[i];  
        for(j=0;j<8;j++){  
            result += x & 0x01010101;  
            x>>=1;  
        }  
        result += (result >> 16);  
        result += (result >> 8);  
    }  
    return result & 0xFF;
```

}

Esta función presenta varias diferencias con la versión "oficial" recomendada en clase, incluyendo la ausencia de una variable auxiliar `val` y la diferente posición de los desplazamientos `>>` y máscara `0xFF`.

Esta función `popcount4`:

- a. produce siempre el resultado correcto
- b. fallaría con `array={0,1,2,3}`
- c. fallaría con `array={1,2,4,8}`
- d. no es correcta pero el error no se manifiesta en los ejemplos propuestos, o se manifiesta en ambos

14. La práctica "parity" debía calcular la suma de paridades impar (XOR de todos los bits) de los elementos de un array. Un estudiante entrega la siguiente versión de `parity5`:

```
int parity5(unsigned * array,  
            int len){  
    int i,j, result = 0;  
    unsigned x;  
    for(i = 0; i<len; i++){  
        x=array[i];  
        for(j=1; j<8*sizeof(int); j*=2)  
            x ^= x >> j;  
        result += x & 0x1;  
    }  
    return result;
```

Esta función sólo se diferencia de la versión "oficial" recomendada en clase, en las condiciones del bucle `for` interno.

Esta función `parity5`:

- a. Produce siempre el resultado correcto
- b. Fallaría con `array={0,1,2,3}`
- c. Fallaría con `array={1,2,4,8}`
- d. No es correcta pero el error no se manifiesta en los ejemplos propuestos, o se manifiesta en ambos

15. La función `gettimeofday()` en la práctica de la "bomba digital" se utiliza para

- a. Para comparar las duraciones de las distintas soluciones del programa
- b. Para imprimir la fecha y hora
- c. Para cifrar la contraseña en función de la hora actual
- d. Para cronometrar lo que tarda el usuario en introducir la contraseña

16. Un fragmento de una “bomba” desensamblada es:

```
0x0804873f: call 0x8048504 <scanf>
0x08048744: mov 0x24(%esp),%edx
0x08048748: mov 0x804a044,%eax
0x0804874d: cmp %eax,%edx
0x0804874f: je 0x8048756 <main+230>
0x08048751: call 0x8048604 <boom>
0x08048756: ...
```

La contraseña/clave en este caso es...

- a. el string almacenado a partir de la posición de memoria 0x804a044
- b. el string almacenado a partir de la posición de memoria 0x24(%esp)
- c. el entero almacenado a partir de la posición de memoria 0x804a044
- d. el entero 0x804a044

17. Una de las “bombas” utiliza el siguiente bucle para cifrar la cadena con la contraseña introducida por el usuario:

```
80485bb: rolb $0x4,(%eax)
80485be: add $0x1,%eax
80485c1: cmp %edx,%eax
80485c3: jne 80485bb <encrypt+0x20>
```

La instrucción **rolb** rota el byte destino hacia la izquierda tantos bits como indica el operando fuente. Si inicialmente eax apunta a la cadena del usuario, que se compara con otra cadena “\x16\x26\x27\x16\x36\x16\x46\x16\x26\x27\x16”, almacenada en el código, la contraseña es:

- a. “\x61\x62\x72\x61\x63\x61\x64\x61\x62\x72\x61” (“abracadabra”)
- b. “\x61\x72\x62\x61\x64\x61\x63\x61\x72\x62\x61” (“arbadacarba”)
- c. “\x63\x61\x64\x61\x62\x72\x61\x61\x62\x72\x61” (“cadabraabra”)
- d. “\x61\x62\x72\x61\x61\x62\x72\x61\x63\x61\x64” (“abraabracad”)

18. Una de las “bombas” utiliza el siguiente código para cifrar la clave numérica introducida por el usuario y ahora almacenada en eax:

```
804870d: xor    $0xffff,%eax
8048712: mov    $0x2,%ecx
8048717: cltd
8048718: idiv   %ecx
804871a: cmp    %eax,0x804a034
```

Si el entero almacenado a partir de 0x804a034 es 0x7ff, la clave numérica puede ser:

- a. 0x10094F97 (269045655)
- b. 0xffff (4095)
- c. 0x7ff (2047)
- d. 1

19. En el programa line.cc de la práctica 5, si para cada tamaño de línea (line) recorremos una única vez el vector, la gráfica resultante es decreciente porque:

- a. Cada vez que line aumenta al doble, el número de aciertos por localidad temporal aumenta, porque ya habíamos accedido a cada posición i del vector cuando lo recorrimos en el punto anterior del eje X.
- b. Cada vez que line aumenta al doble, el número de aciertos por localidad espacial aumenta, porque ya habíamos accedido a cada posición i-1 del vector cuando lo recorrimos en el punto anterior del eje X.
- c. Cada vez que line aumenta al doble, se accede con éxito a más posiciones del vector en niveles de la jerarquía de memoria más rápidos.
- d. Cada vez que line aumenta al doble, realizamos la mitad de accesos al vector que para el valor anterior.

20. ¿Cuál de las siguientes afirmaciones sobre el programa size.cc de la práctica 5 es cierta?

- a. La diferencia de velocidades entre L2 y L3 es mayor que la diferencia de velocidades entre L1 y L2.
- b. Si continuáramos multiplicando por 2 el tamaño del vector en el eje X obteniendo más puntos de la gráfica, esta continuaría horizontal para cualquier valor más allá de 64 MB.
- c. La gráfica tiene escalones hacia arriba porque en cada punto del eje X accedemos al mismo número de elementos del vector y el número de aciertos por localidad temporal disminuye bruscamente en ciertos puntos al aumentar el tamaño del vector.
- d. La gráfica tiene tramos horizontales porque el hecho de realizar la mitad de accesos al vector en cada punto de un tramo horizontal respecto al anterior punto de ese mismo tramo horizontal es compensado por el número de fallos creciente en ese mismo tramo horizontal.

Nombre:

DNI:

Grupo:

**Examen de Problemas (3,0 p)****1. Ensamblador (0.7 puntos).** Dado el siguiente programa escrito en C:

```
#include <stdio.h>

unsigned int x;
unsigned short y;
signed int z;

void main (void) {
    if (x > (signed short) y)
        printf ("Hello ");
    if (x > z)
        printf ("world");
}
```

a) Escriba la función **main()** en ensamblador de IA32. Para ello, tenga en cuenta estas aclaraciones:

1. En comparaciones con tamaños diferentes, se pasa el dato de menor tamaño al tamaño del mayor.
2. En comparaciones que implican **unsigned** y **signed**, se pasa el **signed** a **unsigned**.
3. Si hay que realizar un cambio de tamaño y de “signedness” (de **signed** a **unsigned** o viceversa), primero se realiza el cambio de tamaño manteniendo el signo y luego el de “signedness”.

Ayuda:

- La instrucción **movzwl src,dst** pasa de 16 a 32 bits añadiendo ceros por la izquierda
- La instrucción **movswl src,dst** pasa de 16 a 32 bits extendiendo el signo
- Las instrucciones de salto condicional para **unsigned** usan los sufijos *a* (above) y *b* (below)

b) Suponiendo que las variables *x*, *y* y *z* estuvieran inicializadas así:

```
unsigned int x = 0xFFFFFFFF;
unsigned short y = 0xDEAD;
signed int z = -1;
```

¿qué imprimiría el programa?

**2. Ensamblador.** (0.4 puntos). La práctica "popcount" debía calcular la suma de bits (peso Hamming) de los elementos de un array. Un estudiante entrega la siguiente versión de `popcount4`:

```
int popcount4(unsigned* array, int len) {
    int i,j;
    unsigned x;
    int result = 0;

    for(i=0;i<len;i++){
        x=array[i];
        for(j=0;j<8;j++){
            result += x & 0x01010101;
            x>>=1;
        }
    }
    result += (result >> 16);
    result += (result >> 8);

    return result & 0xFF;
}
```

Esta función presenta varias diferencias con la versión "oficial" recomendada en clase, incluyendo la ausencia de una variable auxiliar **val** y la diferente posición de los desplazamientos `>>` y máscara **0xFF**.

Esta función `popcount4` produce resultado 128 con el último ejemplo "grande" de la batería de tests, que inicializa a `lista[i]=i` un array de  $2^{20}$  elementos pasado como argumento (`popcount4(lista, SIZE)`). Explicar por qué.

(Ayuda: calcular las sumas de bits verticalmente, en lugar de horizontalmente. Si el tamaño del array es potencia de dos, la mitad de los bits de cada columna son 0, y la mitad son 1).

**3. Unidad de control** (0.5 puntos). Suponga una unidad de procesamiento con un bus interno que está conectado a los registros PC, MAR, MDR, Y, Z, IR y R0...R7. También contiene una ALU con una entrada conectada a "4" o al registro Y por medio de un multiplexor, la otra entrada conectada directamente al bus, y la salida conectada al registro Z. El bus interno y todos los registros tienen un tamaño de 32 bits. El desplazamiento es un número con signo relativo al contador de programa que ocupa los 24 bits menos significativos del registro IR cuando la instrucción está en dicho registro, y la salida del registro IR hacia el bus interno es ese campo desplazamiento con el signo extendido a 32 bits. El código que implementa la ejecución de la instrucción de una palabra **BRANCH desplazamiento**, incluyendo la fase de captura de instrucción es el siguiente:

```

fetch:   MAR:=PC; Z:=PC+4;
          PC:=Z; Y:=Z; MDR:=M[MAR];
          IR:=MDR;
          goto f(IR);
branch:  Z:=Y+IR;
          PC:=Z; goto fetch;

```

Dibuje el camino de datos de esta unidad de procesamiento, incluyendo registros, ALU, multiplexor, buses, posibles buffers triestado, y señales de control etiquetadas.

**4. Diseño de E/S y memoria** (0.5 puntos). El microcontrolador ATmega2560 (utilizado en algunas placas Arduino Mega) tiene una arquitectura Harvard (espacios de memoria de programa y de datos separados). El espacio de memoria de datos y E/S mapeada en memoria es direccionable por bytes y se conecta al microcontrolador a través de un bus de direcciones de 16 bits y un bus de datos de 8 bits. El mapa de memoria-E/S incluye las siguientes zonas, en este orden:

- Registros de trabajo de propósito general: 32 registros de 8 bits accesibles a través de las 32 primeras direcciones de memoria
- E/S: 64 direcciones de puertos de E/S, accesibles por instrucciones in/out y load/store
- E/S ampliada: 416 direcciones de puertos de E/S, accesibles por instrucciones load/store
- SRAM interna: 8 KB
- Espacio de memoria SRAM externa opcional: resto del mapa de memoria (55.5 KB)

a. (0.25 puntos) Dibuje el mapa de memoria-E/S, indicando en hexadecimal la primera y la última dirección de cada zona.

b. (0.25 puntos) Existen ampliaciones de memoria externa de un tamaño mayor que el que permite directamente el microcontrolador gracias a la técnica de conmutación de bancos. Algunas de estas ampliaciones usan el chip de memoria AS7C4096A de la figura. Indique el tipo de memoria de este chip (SRAM o DRAM), su configuración expresada como n.<sup>o</sup> de direcciones x n.<sup>o</sup> de datos, y su tamaño total en bits.

A0	1	36	NC
A1	2	35	A18
A2	3	34	A17
A3	4	33	A16
A4	5	32	A15
CE	6	31	OE
I/O1	7	30	I/O8
I/O2	8	29	I/O7
V <sub>CC</sub>	9	28	GND
GND	10	27	V <sub>CC</sub>
I/O3	11	26	I/O6
I/O4	12	25	I/O5
WE	13	24	A14
A5	14	23	A13
A6	15	22	A12
A7	16	21	A11
A8	17	20	A10
A9	18	19	NC

**5. Memoria cache** (0.9 puntos). En la web CPU-World podemos encontrar las siguientes características sobre la jerarquía de memoria del microprocesador i7-6700K, de 4 núcleos (tamaño de línea 64 B):

Level 1 cache size	4 x 32 KB 8-way set associative instruction caches 4 x 32 KB 8-way set associative data caches
Level 2 cache size	4 x 256 KB 4-way set associative caches
Level 3 cache size	8 MB 16-way set associative shared cache
Physical memory	64 GB

Indique el nombre y tamaño en bits de los campos de dirección usados para la política de correspondencia, así como el tamaño total en bits de todas las memorias de etiquetas, tamaño total en bits de todas las memorias de datos/instrucciones, y porcentaje de espacio de etiquetas respecto a datos/instrucciones para cada uno de los tres casos siguientes:

a) Dirección física de memoria principal desde el punto de vista de L3:

Tamaño total en bits ocupado por todas las etiquetas en directorios L3:

Tamaño total en bits ocupado por todos los datos/instrucciones en L3:

Porcentaje de espacio ocupado por etiquetas respecto a datos/instrucciones en L3:

b) Dirección física de memoria principal desde el punto de vista de una L2:

Tamaño total en bits ocupado por todas las etiquetas en directorios L2:

Tamaño total en bits ocupado por todos los datos/instrucciones en L2:

Porcentaje de espacio ocupado por etiquetas respecto a datos/instrucciones en L2:

c) Dirección física de memoria principal desde el punto de vista de una L1:

Tamaño total en bits ocupado por todas las etiquetas en directorios L1:

Tamaño total en bits ocupado por todos los datos/instrucciones en L1:

Porcentaje de espacio ocupado por etiquetas respecto a datos/instrucciones en L1:

## Test de Teoría (3.0p)

<b>1</b>	<b>2</b>	<b>3</b>	<b>4</b>	<b>5</b>	<b>6</b>	<b>7</b>	<b>8</b>	<b>9</b>	<b>10</b>	<b>11</b>	<b>12</b>	<b>13</b>	<b>14</b>	<b>15</b>	<b>16</b>	<b>17</b>	<b>18</b>	<b>19</b>	<b>20</b>	<b>21</b>	<b>22</b>	<b>23</b>	<b>24</b>	<b>25</b>	<b>26</b>	<b>27</b>	<b>28</b>	<b>29</b>	<b>30</b>
d	b	b	b	b	b	d	b	c	a	d	bc	d	a	a	c	c	b	a	a	d	d	a	a	d	d	c	d	a	c

▲ cuentan como acierto b,c, como fallo a,d

## Test de Prácticas (4.0p)

<b>1</b>	<b>2</b>	<b>3</b>	<b>4</b>	<b>5</b>	<b>6</b>	<b>7</b>	<b>8</b>	<b>9</b>	<b>10</b>	<b>11</b>	<b>12</b>	<b>13</b>	<b>14</b>	<b>15</b>	<b>16</b>	<b>17</b>	<b>18</b>	<b>19</b>	<b>20</b>
a	a	c	c	c	d	c	c	b	b	a	d	d	a	d	c	a	d	d	c

## Examen de Problemas (3.0p)

### 1. Ensamblador (0.7 puntos).

- a) Se puntuá **0,6p** por el siguiente programa (**0,05p** por instrucción)

```

movswl      y, %eax      // Cambio a int y luego a unsigned
cmp1       x, %eax      // Compara y con x
jae/jnb     .L2          // Comparación unsigned
pushl      $hello
call       printf
addl      $4, %esp

.L2:
    movl      z, %eax      // Cambio a unsigned
    cmp1      %eax, x      // Compara x con z
    jbe/jna   .L1          // Comparación unsigned
    pushl      $world
    call       printf
    addl      $4, %esp

.L1:

```

- b) Imprimiría “**Hello**” (se puntuá **0,1p** si la respuesta es correcta)

1ª comparación: 0xFFFF FFFF > 0xFFFF DEAD

2ª comparación: 0xFFFF FFFF == 0xFFFF FFFF

### 2. Ensamblador (0.4 puntos).

0.1p

Los dos bucles anidados calculan **4 sumas de bits independientes**, de los bits contenidos en el 1º, 2º, 3º y 4º LSByte.

Las máscaras y acumulación no se aplican tras procesar cada elemento, sino al final del todo, con lo cual no se garantiza que cada una de las 4 sumas quepa en el byte reservado para ella.

Si la suma no se hiciera horizontalmente (elemento a elemento), sino verticalmente (posición de bit a posición de bit) cada posición de bit contiene  $2^{19}$  unos a lo largo del array (mitad ceros, mitad unos), y

**cada 8 posiciones contendrían  $2^{19} \times 2^3 = 2^{22}$  bits activados.**

0.1p

La cuenta correspondiente al byte menos significativo (**1º LSByte**) saldría **0x00 40 00 00**, no cabe en el 1º LSByte, desborda al 3º LSbyte.

El sumando correspondiente al **2º LSByte** estaría desplazado 1B a la izquierda, sería **0x40 00 00 00**

Los sumandos correspondientes al **3º y 4º LSByte se perderían** enteros (no caben en 32bits)

0.1p

Aunque la suma se haga elemento a elemento (y no posición de byte a posición de byte), el total no cambia.

Cambia el orden en que van perdiéndose "acarreos", pero los acarreos que se pierden en total son los mismos.

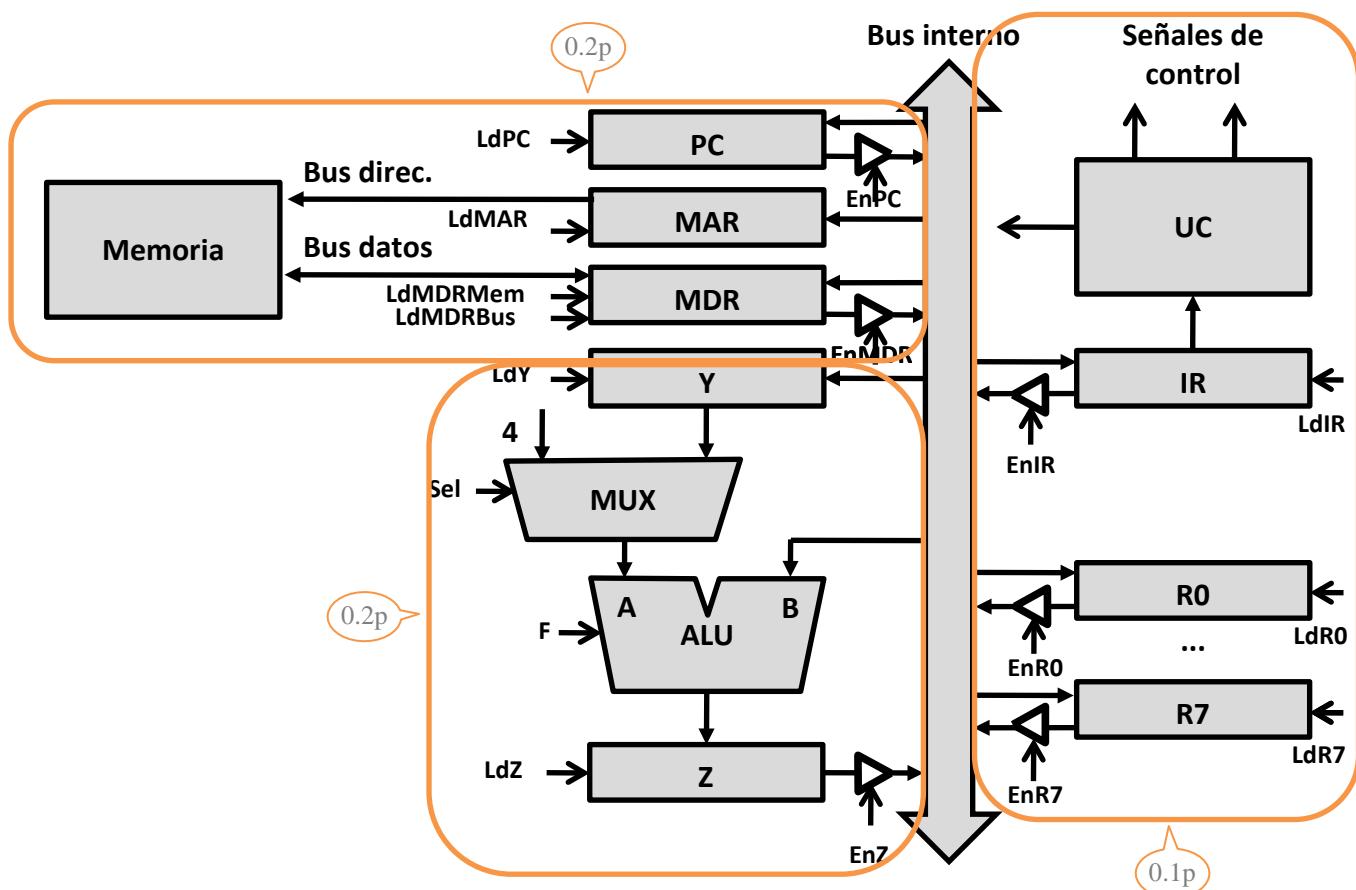
Tampoco cambia porque result sea int en lugar de unsigned. Las sumas son las mismas, se crean con signo o sin signo.

El resultado del bucle for saldría **0x40 40 00 00** (correspondiente al popcount del 1º y 2º bytes menos significativos) y se transforma a → **0x4040 4040 → 0x4080 8080 → 0x0000 0080**, es decir, 128

0.1p

### 3. Unidad de Control (0.5 puntos).

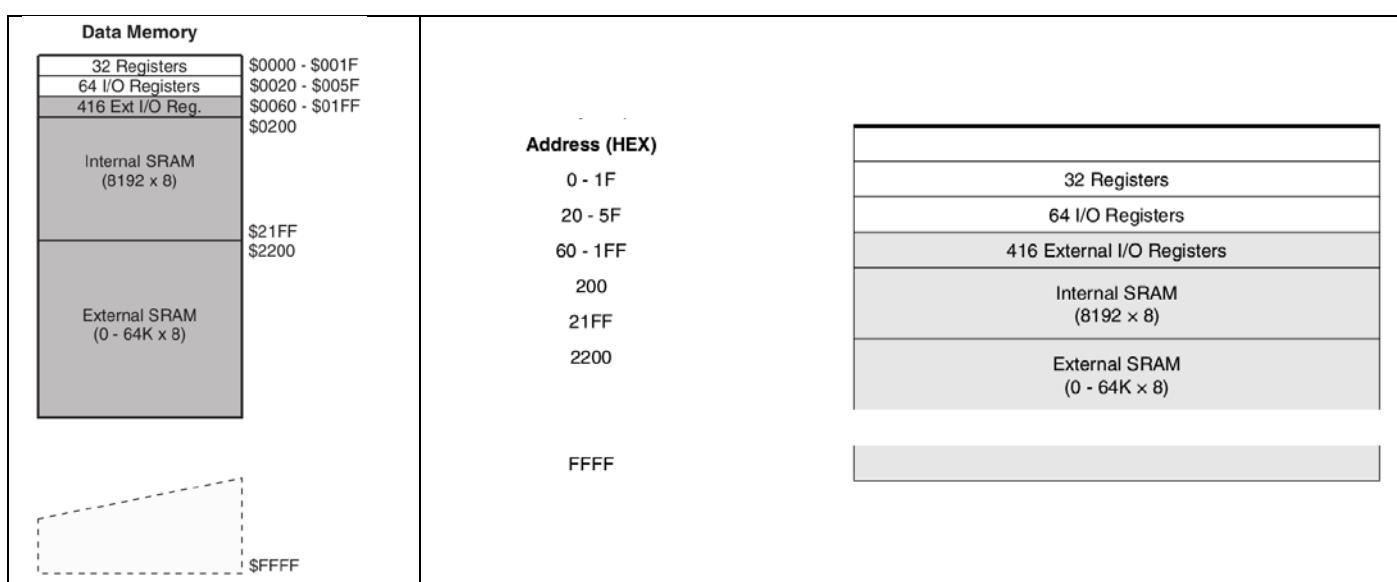
El problema 4 del examen de Septiembre 2016 daba el camino de datos y pedía el pseudocódigo para BRANCH. Esta pregunta es la inversa. La respuesta es aquél camino de datos.



Possiblemente se podría mejorar el dibujo desdoblando en MDR “Bus datos” igual que se desdobló “Bus interno”

### 4. Diseño de E/S y memoria (0.5 puntos).

- a) Se puntuá **0,25p** por cualquier dibujo similar a los siguientes, **0,05p** por cada una de las 5 zonas



#### Referencias:

Atmel ATmega640/V-1280/V-1281/V-2560/V-2561/V datasheet:

[http://www.atmel.com/Images/Atmel-2549-8-bit-AVR-Microcontroller-ATmega640-1280-1281-2560-2561\\_datasheet.pdf](http://www.atmel.com/Images/Atmel-2549-8-bit-AVR-Microcontroller-ATmega640-1280-1281-2560-2561_datasheet.pdf)

ATmega1281/2561/V ATmega640/1280/2560/V datasheet:

[http://pdf.datasheetcatalog.com/datasheets2/30/3055029\\_1.pdf](http://pdf.datasheetcatalog.com/datasheets2/30/3055029_1.pdf)

- b) El chip dispone de patillas /CE y /OE  $\Rightarrow$  **SRAM (0,05 p.)**  
 A0-A18 (19 líneas de dirección) + I/O1-I/O8 (8 líneas de datos)  $\Rightarrow 2^{19} \times 8 = 512\text{ K} \times 8 (0,1\text{ p.})$   
 $2^{19} \times 8 = 2^{19} \times 2^3 = 2^{22} = 4\text{ Mbits}$  (4194304 bits) **(0,1 p.)**

#### Referencias:

QuadRAM Shield for the Arduino Mega/Mega2560:

<https://www.rugged-circuits.com/new-products/quadram>

Arduino Mega 512K SRAM in shield format:

<http://andybrown.me.uk/2013/05/05/512k-xmem-in-shield-format/>

AS7C4096A - Alliance Memory:

[http://www.alliancememory.com/pdf/sram/fa/as7c4096a\\_v1.2.pdf](http://www.alliancememory.com/pdf/sram/fa/as7c4096a_v1.2.pdf)

## 5. Memoria cache (0.9 puntos).

- a) L3 **(0,3p)**

MP: 64 GB =  $2^{36}$  B, L3: 8 MB =  $2^{23}$  B, 64 B/línea =  $2^6$  B/línea, L3: 16 vías =  $2^4$  líneas/conjunto  
 L3:  $2^{23}$  B /  $2^6$  B/línea =  $2^{17}$  líneas,  $2^{17}$  líneas /  $2^4$  líneas/conjunto =  $2^{13}$  conjuntos

Dirección física de memoria principal desde el punto de vista de L3: **(0,15p, 0,05p cada campo)**

etiqueta (17)	conjunto (13)	byte (6)
---------------	---------------	----------

Tamaño total en bits ocupado por todas las etiquetas en directorios L3: **(0,05p)**

1 cache • 16 vías/cache • 8K etiquetas/vía • 17 bits/etiqueta =  $2^{17} \times 17$  bits = **2 228 224 bits**

Tamaño total en bits ocupado por todos los datos/instrucciones en L3: **(0,05p)**

1 cache • 8MB • 8 bits/B =  $2^{23} \times 2^3$  bits = **2<sup>26</sup> bits = 64Mbits = 67 108 864 bits**

Etiquetas / Datos = **3,32%** **(0,05p)**

- b) L2 **(0,3p)**

L3: 8 MB =  $2^{23}$  B, L2: 256 KB =  $2^{18}$  B, 64 B/línea =  $2^6$  B/línea, L2: 4 vías =  $2^2$  líneas/conjunto  
 L2:  $2^{18}$  B /  $2^6$  B/línea =  $2^{12}$  líneas,  $2^{12}$  líneas /  $2^2$  líneas/conjunto =  $2^{10}$  conjuntos

Dirección física de memoria principal desde el punto de vista de una L2:

etiqueta (20)	conjunto (10)	byte (6)
---------------	---------------	----------

Tamaño total en bits ocupado por todas las etiquetas en directorios L2:

4 caches • 4 vías/cache • 1K etiquetas/vía • 20 bits/etiqueta =  $2^{14} \times 20$  bits = **327 680 bits**

Tamaño total en bits ocupado por todos los datos/instrucciones en L2:

4 caches • 256KB/cache • 8 bits/B =  $2^2 \times 2^{18} \times 23$  bits = **2<sup>23</sup> bits = 8Mbits = 8 388 608 bits**

Etiquetas / Datos = **3,91%**

- c) L1 **(0,3p)**

L2: 256 KB =  $2^{18}$  B, L1: 32 KB =  $2^{15}$  B, 64 B/línea =  $2^6$  B/línea, L2: 8 vías =  $2^3$  líneas/conjunto  
 L1:  $2^{15}$  B /  $2^6$  B/línea =  $2^9$  líneas,  $2^9$  líneas /  $2^3$  líneas/conjunto =  $2^6$  conjuntos

Dirección física de memoria principal desde el punto de vista de una L1:

etiqueta (24)	conjunto (6)	byte (6)
---------------	--------------	----------

Tamaño total en bits ocupado por todas las etiquetas en directorios L1:

8 caches • 8 vías/cache • 64 etiquetas/vía • 24 bits/etiqueta =  $2^{12} \times 24$  bits = **98 304 bits**

Tamaño total en bits ocupado por todos los datos/instrucciones en L1:

8 caches • 32KB/cache • 8 bits/B =  $2^3 \times 2^{15} \times 2^3$  bits = **2<sup>21</sup> bits = 2Mbits = 2 097 152 bits**

Etiquetas / Datos = **4,69%**

#### Referencias:

Intel Core i7-6700K specifications en CPU-World:

[http://www.cpu-world.com/CPUs/Core\\_i7/Intel-Core%20i7-6700K.html](http://www.cpu-world.com/CPUs/Core_i7/Intel-Core%20i7-6700K.html)

Cache: A Place for Concealment and Safekeeping:

<http://duartes.org/gustavo/blog/post/intel-cpu-caches/>



Nombre:

DNI:

Grupo:

## Test de Teoría (3.0p)

**Todas las preguntas son de elección simple sobre 4 alternativas.**

Cada respuesta vale 0.1p si es correcta, 0p si está en blanco o claramente tachada, -0.03p si es errónea.

Anotar las respuestas (a, b, c ó d) en la siguiente tabla.

1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30

1. Respecto a direccionamiento a memoria en ensamblador IA32 (syntax AT&T), de la forma D(Rb, Ri, S), sólo una de las siguientes afirmaciones es **FALSA**.  
¿Cuál?
  - a. El desplazamiento D puede ser una constante literal (1, 2 ó 4 bytes)
  - b. EBP no se puede usar como registro base
  - c. ESP no se puede usar como registro índice
  - d. El factor de escala S puede ser 1, 2, 4, 8

---

2. La extensión de signo a m bits de un número original N de n bits, con  $m > n$ , consiste en:
  - a. Realizar la operación  $2^m - N$
  - b. Realizar la operación  $2^m - N - 1$
  - c. Incrementar la cantidad de bits a m preservando el signo y el valor del número.
  - d. Incrementar la cantidad de bits a m rellenando con unos por la izquierda.

---

3. En IA32, ¿cuál de los siguientes fragmentos de programa tiene un efecto sobre los flags distinto al resto?
  - a. sub %edi, %edi  
adc \$0xFFFFFFFF, %edi
  - b. mov \$-1, %edi
  - c. mov \$-1, %edi   
add \$0, %edi
  - d. mov \$0, %edi

---

---

sub \$1, %edi

---

4. Si %rsp vale 0xdeadbeefdeadd0d0, ¿cuál será su nuevo valor después de que se ejecute pushq %rbx?
  - a. 0xdeadbeefdeadd0d4
  - b. 0xdeadbeefdeadd0d8
  - c. 0xdeadbeefdeadd0cc
  - d. 0xdeadbeefdeadd0c8

---

5. ¿Cómo se devuelve en ensamblador x86-64 Linux gcc el valor de retorno de una función long int al terminar ésta?
  - a. La instrucción RET lo almacena en un registro especial de retorno.
  - b. Por convención se guarda en %eax.
  - c. Se almacena en pila justo encima de los argumentos de la función.
  - d. Ninguna de esas formas es la correcta.

---

6. Comparando las convenciones de llamada de gcc Linux IA32 con x86-64 respecto a registros
  - a. En IA32 %ebx es salva-invocante, pero en x86-64 %rbx es salva-invocado
  - b. En IA32 %ecx es salva-invocante, y en x86-64 %rcx es salva-invocante también
  - c. En IA32 %esi es salva-invocado, y en x86-64 %rsi es salva-invocado también
  - d. En IA32 %ebp es especial (marco de pila), y en x86-64 %rbp también

---

7. Son funciones de la unidad de control:

- a. la codificación de las instrucciones máquina
  - b. la lectura de memoria principal de la instrucción apuntada por el µPC
  - c. el secuenciamiento de las instrucciones máquina
  - d. todas las respuestas son ciertas
- 

#### 8. Respecto a MBR y MAR

- a. Ambos son accesibles por el programador
  - b. MAR contiene el dato/instrucción que se leerá o escribirá en memoria
  - c. MAR requiere menos señales de control que MBR
  - d. Ambos permiten guardar información sobre el marco de pila
- 

#### 9. Una instrucción máquina puede desglosarse en las siguientes operaciones elementales:

`sp := sp - 1; m[sp] := pc; pc := x`

Probablemente se trate de una instrucción de:

- a. apilamiento
  - b. llamada a subrutina
  - c. carga local
  - d. almacenamiento local
- 

#### 10. En una unidad de control microprogramada con formato de microinstrucciones vertical, un subcampo que deba especificar 16 señales de control codificadas de tal forma que pueda activarse sólo una o ninguna habrá de tener una anchura mínima de

- a. 4 bits
  - b. 5 bits
  - c. 16 bits
  - d. 17 bits
- 

#### 11. Dado un camino de datos concreto, un posible formato de microprogramación se caracteriza como horizontal o vertical según tenga más o menos (señalar la respuesta falsa)

- a. codificación
  - b. solapamiento
- 

- c. microbifurcaciones
  - d. longitud relativa de microinstrucción
- 

#### 12. El control residual se utiliza para:

- a. reducir el tiempo de ejecución de las instrucciones máquina
  - b. eliminar los bits residuales de la ejecución de las microinstrucciones
  - c. reducir el tamaño de la memoria de control
  - d. ninguna de las anteriores es cierta
- 

#### 13. Un procesador está segmentado en las etapas F, D, E, M y W. Cada una de ellas consume un tiempo $t$ . La aceleración ideal (si no hay riesgos) al ejecutar $n$ instrucciones respecto a un procesador no segmentado será:

- a.  $5n / (4+n)$
  - b.  $(4+n) / 5t$
  - c.  $4n / (5+n)$
  - d.  $(5+n) / 4t$
- 

#### 14. En un procesador con segmentación de cauce, aumentar el número de etapas (p.ej. de 2 a 4, o de 4 a 8), tiene en general como consecuencia:

- a. Un incremento de las prestaciones
  - b. Un mayor retraso en la ejecución de los programas debido al incremento del número de etapas
  - c. Una disminución en la posible dependencia de datos
  - d. Una disminución de la máxima frecuencia de reloj a la que puede operar el cauce
- 

#### 15. En la secuencia de instrucciones siguiente, siendo el primer registro el destino, ¿cuántos riesgos se dan?

`sub r2,r1,r3  
or r8,r6,r2`

- a. Un riesgo estructural
  - b. Un riesgo por dependencia de datos
  - c. Un riesgo estructural y dos por dependencia de datos
  - d. Dos riesgos por dependencia de datos y uno de control
-

16. La precaptación (cola de instrucciones) está relacionada con...

- a. Los riesgos estructurales (intenta evitar el efecto de un fallo de cache)
  - b. Los riesgos de (dependencia de) datos (intenta que el dato esté disponible anticipadamente)
  - c. Los riesgos de control (intenta determinar de antemano el flujo de control)
  - d. Los riesgos de transferencia (intenta agrupar las posibles transferencias de un conjunto de instrucciones)
- 

17. Respecto a la segmentación, ¿cuál de las siguientes afirmaciones es falsa?

- a. La técnica de register forwarding habilita una serie de caminos (buses) que se añaden al cauce para permitir que los resultados de una etapa pasen como entradas a la etapa donde son necesarias
  - b. La reorganización del código y la introducción de instrucciones nop permite evitar dependencias de datos
  - c. Retrasar la fase de decisión saltar/no saltar de las instrucciones de salto condicional contribuye a mejorar el rendimiento del procesador
  - d. Cuantas más etapas tenga un cauce, más instrucciones se estarán ejecutando en distintas fases y más posibilidades se presentan de que existan riesgos entre ellas
- 

18. ¿Cuál de los siguientes modos de direccionamiento es menos preferible para un procesador de 32 bits y con tamaño de instrucción de 32 bits?

- a. registro
  - b. indexado
  - c. indirecto a través de registro
  - d. directo (o absoluto)
- 

19. La conexión entre un dispositivo de E/S y el procesador mediante bus:

- a. Es difícil de expandir
  - b. Permite conectar en paralelo varios dispositivos
- 

- c. Requiere mucha circuitería
  - d. Requiere multiplexores y demultiplexores para las señales de datos
- 

20. El fragmento de código ensamblador de un microprocesador de 8 bits

```
lds IOBuf ; Apuntar puntero pila a  
           ; ...área mem.intermedia  
idx Count ; Inicializar X-contador  
poll lda a Status; Leer estado en A  
          bpl poll ; Signo(A)!=1 => repetir  
          lda a Data ; Leer dato en A  
          psh a      ; Transferir dato a pila  
          dex       ; Decrementar contador X  
          bne poll  ; Seguir leyendo si X!=0
```

corresponde a:

- a. Entrada programada con consulta de estado
  - b. Salida programada sin consulta de estado
  - c. Entrada programada sin consulta de estado
  - d. Salida programada con consulta de estado
- 

21. En la E/S controlada por interrupciones:

- a. El controlador de DMA transfiere bloques de datos por el bus del sistema.
  - b. El controlador de DMA envía una petición de interrupción a la CPU.
  - c. La CPU lee y comprueba el estado de los dispositivos de E/S (en el caso de consulta de estado).
  - d. La CPU transfiere el control a una rutina de servicio cuando recibe una interrupción.
- 

22. La instrucción máquina DI (Disable Interrupts), conocida como CLI (Clear Interrupt Flag) en x86, se utiliza para desactivar:

- a. Todas las interrupciones enmascarables
  - b. Las interrupciones de inferior o igual prioridad a una dada
  - c. Determinados niveles de interrupción de forma selectiva
  - d. Las interrupciones software
-

23. Con nueve controladores de interrupciones 8259 se pueden manejar exactamente:

- a. 8 niveles de prioridad
- b. 16 niveles de prioridad
- c. 24 niveles de prioridad
- d. Ninguna de las anteriores es cierta

24. ¿Cuál de los siguientes es un registro de un controlador de DMA?

- a. IR (Instruction Register)
- b. PC (Program Counter)
- c. SP (Stack Pointer)
- d. WC (Word Count)

25. Respecto al refresco de memorias DRAM, ¿cuál de las siguientes afirmaciones es falsa?

- a. Una operación de refresco consiste en dar un impulso /CAS junto con una dirección de columna.
- b. Los chips DRAM refrescan automáticamente la fila accedida en cualquier ciclo de lectura o escritura.
- c. Se precisa una circuitería auxiliar, externa al chip DRAM o integrada en él, que produzca ciclos de refresco.
- d. Los ciclos de refresco deben producirse cada pocos ms (milisegundos).

26. La tasa de aciertos  $A_i$  del nivel  $i$  de una jerarquía de memoria no depende de:

- a. La capacidad (tamaño)  $s_i$  del nivel  $i$ .
- b. La estrategia de administración de memoria.
- c. La unidad de la transferencia de información  $x_i$  entre el nivel  $i$  y el  $i+1$ .
- d. El ancho de banda  $b_i$  del nivel  $i$ .

27. La política de correspondencia de una memoria cache con 1 único conjunto es:

- a. Directa
- b. Totalmente asociativa
- c. Asociativa por conjuntos con una única línea
- d. Asociativa por conjuntos de una única vía

28. La política de correspondencia de una memoria cache con la mitad de conjuntos que líneas es:

- a. Asociativa por conjuntos de 2 vías
- b. Totalmente asociativa de media vía
- c. Asociativa por conjuntos con 2 líneas
- d. Directa con 2 líneas

29. Para construir una DRAM de 4GB con pastillas de 512Mx4bit hacen falta

- a. 8 pastillas
- b. 16 pastillas
- c. 32 pastillas
- d. 64 pastillas

30. Para diseñar una memoria con ancho de palabra  $k*m$  (y mismo  $n^o$  palabras que los módulos) a partir de módulos con ancho de palabra  $m$ , se utilizan  $k$  módulos

- a. repartiendo las líneas de datos entre los  $k$  módulos: el primero se conecta a  $D_0 \dots D_{k-1}$ , el segundo a  $D_k \dots D_{2k-1}$ , etc
- b. repartiendo las líneas de dirección: el 1º se conecta a  $A_0 \dots A_{k-1}$ , el 2º a  $A_k \dots A_{2k-1}$ , etc
- c. repartiendo líneas datos: el 1º se conecta a  $D_0 \dots D_{m-1}$ , el 2º a  $D_m \dots D_{2m-1}$ , etc
- d. repartiendo líneas dirección: el 1º se conecta a  $A_0 \dots A_{m-1}$ , el 2º a  $A_m \dots A_{2m-1}$ , etc



Nombre:

DNI:

Grupo:

## Test de Prácticas (4.0p)

Todas las preguntas son de elección simple sobre 4 alternativas.

Cada respuesta vale 0.2p si es correcta, 0 si está en blanco o claramente tachada, -0.06p si es errónea.

Anotar las respuestas (a, b, c ó d) en la siguiente tabla.

1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20

1. ¿Cuál de las siguientes instrucciones máquina copia en EAX la dirección efectiva resultante de la operación EDX\*4 + EBX?
    - a. leal 4(%edx, %edx), %eax
    - b. leal (%ebx, %edx, 4), %eax
    - c. movl 4(%edx, %edx), %eax
    - d. movl (%ebx, %edx, 4), %eax
  2. ¿Cuál de las siguientes instrucciones máquina copia en EAX el entero almacenado en la posición de memoria cuya dirección efectiva es el resultado de la operación EDX\*4 + EBX?
    - a. leal 4(%edx, %edx), %eax
    - b. leal (%ebx, %edx, 4), %eax
    - c. movl 4(%edx, %edx), %eax
    - d. movl (%ebx, %edx, 4), %eax
  3. Los switches `-m elf_i386` y `-m elf_x86_64` para trabajar en 32 bits / 64 bits corresponden a la herramienta...
    - a. gcc
    - b. as
    - c. ld
    - d. nm
- 
4. Si ECX vale 0, la instrucción adc \$0,%ecx
    - a. Pone CF=1
    - b. Pone CF=0
    - c. Cambia CF
    - d. No cambia CF
  5. Dada la siguiente definición de datos:

```
lista: .int 0x10000000, 0x50000000,  
       0x10000000, 0x20000000  
longlista: .int (. lista)/4  
resultado: .quad 0x123456789ABCDEF  
formato: .ascii "suma=%llu=%llx hex\n\0"
```

La instrucción para copiar la dirección de memoria donde comienza lista en el registro EBX es:
    - a. movl lista, %ebx
    - b. movl \$lista, %ebx
    - c. movl (lista), %ebx
    - d. movl \$lista, (%ebx)
  6. Dada la siguiente definición de datos:

```
lista: .int 0x10000000, 0x50000000,  
       0x10000000, 0x20000000  
longlista: .int (. lista)/4  
resultado: .quad 0x123456789ABCDEF  
formato: .ascii "suma=%llu=%llx hex\n\0"
```

la instrucción `movl longlista, %ecx` copia el siguiente valor:

- a. 4
  - b. 8
  - c. 16
  - d. 32
- 

7. Dada la siguiente definición de datos:

```
lista: .int 0x10000000, 0x50000000,  
       0x10000000, 0x20000000  
longlista: .int (. - lista) / 4  
resultado: .quad 0x123456789ABCDEF  
formato: .ascii "suma=%llu=%llx hex\n\0"
```

y suponiendo que hemos llamado a una función *suma* que devuelve un número de 64 bits en la pareja EDX:EAX, las instrucciones que copian ese número en *resultado* son:

- a. movl %eax, resultado  
 movl %edx, resultado+4
  - b. movl (%eax), resultado  
 movl (%edx), resultado+4
  - c. movl %eax, resultado+4  
 movl %edx, resultado
  - d. movl (%eax), resultado+4  
 movl (%edx), resultado
- 

8. Dada la siguiente definición de datos:

```
lista: .int 0x10000000, 0x50000000,  
       0x10000000, 0x20000000  
longlista: .int (. - lista) / 4  
resultado: .quad 0x123456789ABCDEF  
formato: .ascii "suma=%llu=%llx hex\n\0"
```

la llamada correcta a *printf* será:

- a. push resultado+4  
 push resultado  
 push \$formato  
 call printf  
 add \$12, %esp
  - b. push resultado+4  
 push resultado  
 push resultado+4  
 push resultado
- 

```
push $formato  
call printf  
add $20, %esp
```

- c. push resultado  
 push resultado+4  
 push \$formato  
 call printf  
 add \$12, %esp
  - d. push resultado  
 push resultado+4  
 push resultado  
 push resultado+4  
 push \$formato  
 call printf  
 add \$20, %esp
- 

9. En la práctica “media” se pide sumar una lista de enteros **\*con\*** signo de 32 bits en una plataforma de 32 bits sin perder precisión, esto es, evitando overflow. ¿Cuál es el menor valor positivo que repetido en los **\*dos\*** primeros elementos de la lista causaría overflow con 32 bits al realizar la suma de **\*esos dos\*** primeros elementos de la lista?

- a. 0x0400 0000
  - b. 0x0800 0000
  - c. 0x4000 0000
  - d. 0x8000 0000
- 

10. ¿Cuál de las siguientes afirmaciones es cierta respecto al lenguaje C?

- a. En lenguaje C, al llamar a una subrutina o función se introducen los parámetros en la pila y después se realiza una llamada a la subrutina
- b. Los parámetros se introducen en la pila en el orden en el que aparecen en la llamada de C, es decir, empezando

por el primero y acabando por el último

- c. Antes de volver de la rutina llamada, el programa en C se encarga de quitar de la pila los parámetros de llamada realizando varios pop
  - d. Pasar a una función un puntero a una variable se traduce en introducir en la pila el valor de la variable
- 

11. ¿Cuál de los siguientes registros tiene que ser salvaguardado (si va a modificarse) dentro de una subrutina según la convención cdecl para IA32?

- a. eax
  - b. ebx
  - c. ecx
  - d. edx
- 

12. ¿En qué registro se pasa el primer argumento a una función en Linux gcc x86-64?

- a. ecx
  - b. edx
  - c. esi
  - d. edi
- 

13. La práctica "popcount" debía calcular la suma de bits (peso Hamming) de los elementos de un array. Un estudiante entrega la siguiente versión de popcount3:

```
int popcount3(unsigned* array,
              int len){
    int i, res = 0;
    unsigned x;
    for( i=0; i<len; i++ ) {
        x = array[i];
        asm("ini3:          \n"
             "shr %[x]       \n"
             "adc $0, %[r]   \n"
             "add $0, %[x]   \n"
             "jne ini3      \n"
             : [r] "+r" (res)
             : [x] "r" (x) );
    }
    return res;
}
```

Esta función produce siempre el resultado correcto, a pesar de que una instrucción máquina en la sección **asm()** es distinta a la que se esperaba después de haber estudiado **pcount\_r** en teoría. La instrucción distinta también se podría haber cambiado por...

- a. sar %[x]
  - b. adc \$0, %[x]
  - c. test %[x], %[x]
  - d. cmp %[x], %[r]
- 

14. En la práctica de la bomba necesitamos estudiar el código máquina de la bomba del compañero. A veces dicho código no se visualiza directamente en el depurador ddd, y algunas de las técnicas que se pueden probar para conseguir visualizarlo son... (marcar la opción **\*falsa\***)

- a. comprobar que está activado el panel *View → Machine Code Window*
  - b. escribir **info line main** en el panel de línea de comandos gdb
  - c. recompilar con información de depuración, por si se nos había olvidado, ya que sin **-g** el ejecutable no contiene información de depuración
  - d. asegurarse de que se ha escrito correctamente el nombre del ejecutable
- 

15. En la práctica de la bomba, el primer ejercicio consistía en “saltarse” las “explosiones”, para lo cual se puede utilizar...

a. objdump o gdb

b. gdb o ddd

c. ddd o hexedit

d. hexedit u objdump

---

16. En la práctica de la bomba, el segundo ejercicio consistía en crear un ejecutable sin “explosiones”, para lo cual se puede utilizar...

a. objdump o gdb

b. gdb o ddd

c. ddd o hexedit

d. hexedit u objdump

---

17. En la práctica de la bomba, el tercer ejercicio consistía en usar un editor hexadecimal para crear un ejecutable sin “explosiones”. Para saber qué contenidos del fichero hay que modificar, se puede utilizar... (marcar la opción falsa)

a. objdump

b. gdb

c. ddd

d. hexedit

---

18. Suponer una memoria cache con las siguientes propiedades: Tamaño: 512 bytes. Política de reemplazo: LRU. Estado inicial: vacía (todas las líneas inválidas). Suponer que para la siguiente secuencia de direcciones enviadas a la cache: 0, 2, 4, 8, 16, 32, la tasa de acierto es 0,33. ¿Cuál es el tamaño de bloque de la cache?

a. 4 bytes

b. 8 bytes

c. 16 bytes

d. Ninguno de los anteriores

---

19. Abajo se ofrece el listado de una función para multiplicar matrices  $C = A \times B$ .

```
void mult_matr(float A[N][N],  
float B[N][N],float C[N][N]) {  
/*Asume val.ini. C={0,0...}*/  
int i,j,k;  
for (i=0; i<N; i++)  
    for (j=0; j<N; j++)  
        for (k=0; k<N; k++)  
            C[i][j] += A[i][k]*B[k][j];  
}
```

Suponer que:

- El computador tiene una cache de datos de 8 MB, 16-vías, líneas de 64 bytes.
- N es grande, una fila o columna no cabe completa en cache.
- El tamaño de los tipos de datos es como en IA32.
- El compilador optimiza el acceso a  $C[i][j]$  en un registro.

Aproximadamente, ¿qué tasa de fallos se podría esperar de esta función para valores grandes de N?

a. 1/2

b. 1/4

c. 1/8

d. 1/16

---

20. Sea un computador de 32 bits con una memoria cache L1 para datos de 32 KB y líneas de 64 bytes asociativa por conjuntos de 2 vías. Dado el siguiente fragmento de código:

```
int v[262144];  
for (i=0; i<262144; i+=8)  
    v[i] = 9;
```

¿Cuál será la tasa de fallos aproximada que se obtiene en la ejecución del bucle anterior?

a. 0 (ningún fallo)

b. 1/2 (mitad aciertos, mitad fallos)

c. 1/8 (un fallo por cada 8 accesos)

d. 1 (todo son fallos)

---



Nombre:	
DNI:	Grupo:

### Examen de Problemas (3.0 p)

**1. Ensamblador** (0.3 puntos). Escriba la instrucción máquina (una sola) necesaria para llevar a cabo cada una de las siguientes operaciones:

- a. (0.05) Quitar un entero de la cabecera de la pila y guardarlo en el reg. EAX:
- b. (0.05) Sumar el valor del indicador de acarreo (CF) al registro ECX:
- c. (0.05) Llamar a la función *f*:
- d. (0.05) Copiar el carácter (byte) situado en la cabecera de la pila en el registro CL sin alterar la pila:
- e. (0.05) Multiplicar el contenido del registro EAX por 5, dejando el resultado en el mismo registro:
- f. (0.05) Poner el registro EDX a cero sin usar la instrucción mov:

**2. Estructuras** (0.8 puntos). Considerar las siguientes declaraciones de estructuras en lenguaje C:

```
struct a {  
    float* f;  
    char c;  
    int i;  
    char  
    z[4];  
    double d;  
    short s;  
};  
struct b {  
    struct a a1;  
    int j;  
    struct a a2;  
};
```

- a. (0.2) Indicar cuántos bytes ocupa struct a en Linux gcc x86:
- b. (0.2) Indicar cuántos bytes ocupa struct a en Linux gcc x86-64:
- c. (0.2) Indicar cuántos bytes ocupa struct b en Linux gcc x86:
- d. (0.2) Indicar cuántos bytes ocupa struct b en Linux gcc x86-64:

**3. Depuración** (0.3 puntos). La práctica "popcount" debía calcular la suma de bits (peso Hamming) de los elementos de un array. Un estudiante entrega la siguiente versión de `popcount3`:

```
#define NBITS 20
#define SIZE ( 1<<NBITS )
unsigned lista[SIZE];
#define RESULT (NBITS*(1<<(NBITS-1)))

int popcount3(int* array,
              int len){
    long val = 0;
    int i;
    unsigned x;

    for (i=0; i<len; i++){
        x= array[i];
        do{
            val += x & 0x1;
            x >= 1;
        }while (x);
        val += (val >> 16);
        val += (val >> 8);
    }
    return val & 0xFF;
}
```

Aunque para `popcount3` se pedía una implementación `asm()`, esta función produce el resultado correcto con los tres ejemplos “pequeños” (4, 156 y 116), pero falla con el ejemplo “grande” (`lista={0,1, ... 1048575}`) con el cual produce el resultado 237. Es posible razonar que esta función produce el resultado correcto ( $NBITS \cdot 2^{NBITS-1}$ ) siempre que queden sin efecto las partes erróneas del código (`val>>16`, `val>>8` y `val&0xFF`), lo cual sucede para valores de `NBITS` menores que o iguales a...

`NBITS <=`

**4. Unidad de Control** (0.4 puntos). El contenido de la memoria de control de una unidad de control microprogramada es:

Dirección	Contenido
0000	0000 0010
0001	0011 0111
0010	0010 0100
0011	0100 1010
0100	0000 0011
0101	0000 0010
0110	0011 0111
0111	0100 1010
1000	0100 1010
1001	0011 0111
1010	0100 1010
1011	0011 0111
1100	0000 0011
1101	0011 0111

Calcule:

- a. (0.1) El tamaño en bits total de dicha memoria de control:
- b. (0.2) El tamaño en bits total que requeriría un diseño nanoprogramado:
- c. (0.1) El ahorro en bits expresado en porcentaje:

**5. Entrada/Salida** (0.5 puntos). Complete el código ensamblador de una función

```
void write_value (unsigned char value)
```

que realice una operación de salida con consulta de estado. El puerto de estado, en la dirección 0x22C, puede leerse con la instrucción `inb $0x22C, %al`. El puerto de datos de salida, en la misma dirección 0x22C, puede escribirse con la instrucción `outb %al, $0x22C`. La consulta de estado consistirá en leer del puerto de estado mientras el bit 7 valga 1, o sea, leer el puerto hasta que el bit 7 valga 0. Sólo entonces puede procederse a escribir el dato pasado a la función en el puerto de datos de salida.

```
write_value:  
    pushl %ebp  
    movl %esp, %ebp  
  
    ; Escribir código de salida programada  
    ; con consulta de estado desde aquí...  
  
    ; ...hasta aquí  
  
    popl %ebp  
    ret
```

**6. Configuración de memoria** (0.4 puntos). Disponemos de circuitos SRAM de 32 K x 4 + EPROM de 16 K x 4 y queremos construir una memoria direccionable por bytes con un bus de datos de 8 bits y la siguiente configuración: SRAM de 64 K x 8 a partir de la dirección 0xE0000, EPROM de 32 K x 8 a partir de la dirección 0x00000. A cada chip SRAM se conectan las patillas de dirección A<sub>14</sub>-A<sub>0</sub>. A cada chip EPROM se conectan las patillas del bus de direcciones A<sub>13</sub>-A<sub>0</sub>.

- a. (0.1) ¿Cuántos circuitos SRAM necesitamos?
- b. (0.1) ¿Cuántos circuitos EPROM necesitamos?
- c. (0.1) ¿Cuál es la última dirección de la SRAM?
- d. (0.1) ¿Cuál es la última dirección de la EPROM?

**7. Memoria cache** (0.3 puntos). La siguiente tabla indica la frecuencia de fallos de varias caches:

Tamaño	Cache sólo instrucciones	Cache sólo datos	Cache unificada
8 KB	5.8%	6.8%	8.3%
16 KB	3.6%	5.3%	5.9%
32 KB	2.2%	4.0%	4.3%

El porcentaje de referencias a instrucciones es 53%. Se desea conocer cuál de estas dos configuraciones tiene una frecuencia de fallos menor:

- Configuración 1: una caché de instrucciones de 16 KB + una caché de datos de 16 KB
- Configuración 2: una caché unificada de 32 KB.

- a. (0.1) Frecuencia de fallos para configuración 1:
- b. (0.1) Frecuencia de fallos para configuración 2:
- c. (0.1) ¿Cuál tiene una frecuencia de fallos menor, separada o unificada?

### Test de Teoría (3.0p)

<b>1</b>	<b>2</b>	<b>3</b>	<b>4</b>	<b>5</b>	<b>6</b>	<b>7</b>	<b>8</b>	<b>9</b>	<b>10</b>	<b>11</b>	<b>12</b>	<b>13</b>	<b>14</b>	<b>15</b>	<b>16</b>	<b>17</b>	<b>18</b>	<b>19</b>	<b>20</b>	<b>21</b>	<b>22</b>	<b>23</b>	<b>24</b>	<b>25</b>	<b>26</b>	<b>27</b>	<b>28</b>	<b>29</b>	<b>30</b>
b	c	<b>bd</b>	d	d	b	c	c	b	b	c	c	a	a	b	a	c	d	b	a	d	a	d	d	a	d	b	a	b	c

↑ cuentan como acierto b,d, como fallo a,c

errata detectada en Ene'2020 por Jose Miguel Márquez Herreros y compañeros. Gracias!!!

### Test de Prácticas (4.0p)

<b>1</b>	<b>2</b>	<b>3</b>	<b>4</b>	<b>5</b>	<b>6</b>	<b>7</b>	<b>8</b>	<b>9</b>	<b>10</b>	<b>11</b>	<b>12</b>	<b>13</b>	<b>14</b>	<b>15</b>	<b>16</b>	<b>17</b>	<b>18</b>	<b>19</b>	<b>20</b>
b	d	c	b	b	a	a	b	c	a	b	d	c	c	b	<b>bc</b>	d	b	a	b

↑ cuentan como acierto b,c, como fallo a,d

### Examen de Problemas (3.0p)

#### 1. Ensamblador (0.3 puntos). Se puntúa **0,05p** por instrucción

- a) `pop %eax` # alternativa
- `popl %eax`
- b) `adc $0, %ecx` # alternativa
- `adcl $0, %ecx`
- c) `call f`
- d) `mov (%esp), %cl` # alternativa
- `movb (%esp), %cl`
- e) `lea (%eax, %eax, 4), %eax` # alternativas
- `leal (%eax, %eax, 4), %eax`
- `imul $ 5, %eax`
- `imull $0x5, %eax`
- f) `xor %edx, %edx` # alternativas
- `xorl %edx, %edx`
- `and $0x0, %edx`
- `andl $ 0, %edx`
- `sub %edx, %edx`
- `subl %edx, %edx`
- `imul $ 0, %edx`
- `imull $0x0, %edx`
- `lea 0x0, %edx`
- `leal 0, %edx`

#### 2. Estructuras (0.8 puntos). Se puntúa **0,2p** por apartado

- a) **28** f 4B c 1B X 3B i 4B z 4B d 8B s 2B X 2B tot: 28B req: 4x
- b) **40** f 8B c 1B X 3B i 4B z 4B X 4B d 8B s 2B X 6B tot: 40B req: 8x
- c) **60** a1 28B j 4B a2 28B tot: 60B req: 4x
- d) **88** a1 40B j 4B X 4B a2 40B tot: 88B req: 8x

#### 3. Depuración (0.3 puntos).

NBITS <= **6**

Si NBITS=6, RESULT=6·2<sup>5</sup>=192<255, y si NBITS=7, RESULT=7·2<sup>6</sup>=448>255

En cuanto val>=0xFF, val>>8 no es cero y es de esperar que se calcule erróneamente un valor mayor que el correcto

**4. Unidad de control** (0.4 puntos). Se puntúa **0,1-0,2-0,1p**

- a) **112** bits =  $14 \cdot 8$
- b) **82** bits =  $14 \cdot 3 + 5 \cdot 8$
- c) **26.8%** ahorro  $112 - 82 = 30$  bits, porcentaje  $30/112 = 0.267857143$

**5. Entrada/Salida** (0.5 puntos). Se puntúa **0,1p** por instrucción correcta

```
poll:  inb   $0x2C, %al  # alternativas
       test  %al, %al    # testb      # and $0x80, %al  # shl %al  # rol/rcl/rcr $1,
       js    poll        # jnz poll    # jc poll

       movb  8(%ebp), %al
       outb %al, $0x2C
```

**6. Configuración de memoria** (0.4 puntos). Se puntúa **0,1p** por apartado

- a) **4**  $32K \times 4 \rightarrow 64K \times 8 \Rightarrow 2 \times 2$
- b) **4**  $16K \times 4 \rightarrow 32K \times 8 \Rightarrow 2 \times 2$
- c) **0xE FFFF**  $64K = 2^{16} \text{ --- } 16 \text{ bits} \Rightarrow 0x0000\dots0xFFFF$
- d) **0x0 7FFF**  $32K = 2^{15} \text{ --- } 15 \text{ bits} \Rightarrow 0x0000\dots0x7FFF$

**7. Memoria cache** (0.3 puntos). Se puntúa **0,1p** por apartado

- a) **4.399%**  $0.53 \cdot 3.6\% + 0.47 \cdot 5.3\% = 4.399\% = 4.4\%$
- b) **4.3%** directo de la tabla
- c) **Unificada**  $4.3 < 4.4$

**Nombre:**
**DNI:**
**Grupo:**

### Test de Teoría (3.0p)

Todas las preguntas son de elección simple sobre 4 alternativas.

Cada respuesta vale 0.1p si es correcta, 0p si está en blanco o claramente tachada, -0.03p si es errónea.

Anotar las respuestas (a, b, c ó d) en la siguiente tabla.

1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30

1. ¿Cuál es el valor mínimo (más negativo) que puede tomar un entero de 32 bits en complemento a dos?

- a.  $-2^{32}$
- b.  $-2^{32} + 1$
- c.  $-2^{31}$
- d.  $-2^{31} + 1$

2. Un datapath con bus de direcciones de 32 bits y bus de datos de 16 bits tiene un registro de 16 bits conectado al bus de datos y a la unidad de control. Puede tratarse del registro

- a. IR
- b. SP
- c. MAR
- d. PC

3. ¿Cuál de las siguientes características sobre RISC es **\*FALSA\***?

- a. Para acelerar un procesador RISC se deberían emplear técnicas de segmentación.
- b. Las instrucciones máquina en un procesador RISC deberían ser complejas y potentes.
- c. La decodificación de las instrucciones debe ser simple: un procesador RISC debería emplear pocos formatos de instrucción.
- d. La unidad de control de un procesador RISC debería ser cableada, no microprogramada.

4. ¿Cuál de las siguientes instrucciones máquina copia en el registro EDX la dirección efectiva resultante de la operación  $EAX*8 + EBX$ ?

- a. movl (%ebx, %eax, 8), %edx
- b. movl 8(%edx, %eax), %edx
- c. leal (%ebx, %eax, 8), %edx
- d. leal 8(%edx, %eax), %edx

5. En el contexto general del lenguaje máquina, el acrónimo ISA suele referirse a:

- a. Internal Standard Architecture
- b. Integrated Set Assembly
- c. Instruction System Architecture
- d. Instruction Set Architecture

6. En una suma de dos números en complemento a dos, se produce desbordamiento cuando

- a. Sumamos dos positivos y el resultado es negativo o bien sumamos dos negativos y el resultado es positivo.
- b. Sumamos dos positivos y el resultado es positivo.
- c. Sumamos un número positivo y uno negativo.
- d. Sumamos dos negativos y el resultado es negativo.

7. Usando el repertorio IA32, para intercambiar el valor de 2 registros se pueden usar...

- a. 4 mov, no menos (debido a la arquitectura R/M)
- b. 3 mov, no menos (se le llama "intercambio circular")
- c. dos instrucciones mov
- d. una instrucción mov y una instrucción lea

8. Al ejecutar el fragmento de código:

```
leal    -1(%eax), %edx
cmpl    $9, %edx
ja     .L2
```

se salta a .L2 si el contenido del registro %eax:

- a. es menor o igual que 1
- b. es mayor o igual que 10
- c. está fuera del intervalo [1,10]
- d. está dentro del intervalo [1,10]

9. ¿Cuál de las siguientes instrucciones convierte  $\%eax = 5 * \%eax$ ?

- 1) `mov 4(%eax, %eax), %eax`
- 2) `lea 4(%eax, %eax), %eax`

- a. Sólo la 2
- b. Sólo la 1
- c. Ambas, la 1 y la 2
- d. Ninguna de las dos

10. Si el registro **r12b** contiene la variable booleana **cond**, y **rax** la variable **valor**, la secuencia de instrucciones:

```
testb %r12b, %r12b
movq $13, %rax
cmovc $17, %rax
```

realiza la operación:

- a.  $\text{valor} = \text{cond} ? 13 : 17;$
- b.  $\text{valor} = 17;$
- c.  $\text{valor} = \text{cond} ? 17 : 13;$
- d.  $\text{valor} = 13;$

11. En una matriz declarada como “int a[n][n];” en lenguaje C...

- a. los n elementos de una columna se almacenan en memoria de manera contigua
- b. los n elementos de una fila se almacenan en memoria de manera contigua
- c. podría haber huecos de relleno al final de cada columna para alineamiento, dependiendo de n
- d. podría haber huecos de relleno al final de cada fila para alineamiento, dependiendo de n

12. ¿Cuáles de las siguientes señales son entradas a la unidad de control?

- a. El contenido del contador de programa
- b. Las señales de habilitación de buffers triestado entre registros y buses

c. El contenido del registro de instrucción

d. Las señales de control de la ALU

13. Una CPU con bus de direcciones de 64 bits y bus de datos de 32 bits tiene un registro de 64 bits conectado al bus de direcciones de la memoria. Probablemente se trata del registro

- a. IR
- b. MBR
- c. Acumulador
- d. MAR

14. En la secuencia de instrucciones siguiente, siendo el primer registro el destino, ¿cuántos riesgos se dan?

```
sub r2, r1, r3
or r8, r6, r1
```

- a. Un riesgo por dependencia de datos
- b. Un riesgo estructural
- c. Un riesgo por dependencia de datos y uno de control
- d. Ninguno

15. Un sistema no segmentado tarda 10 ns en procesar una tarea. La misma tarea puede ser procesada en un cauce (pipeline) con un ciclo de reloj de 5 ns. Cuando se procesan muchas tareas, la ganancia máxima de velocidad que se obtiene se aproxima a:

- a. 2
- b. 5
- c. 10
- d. 20

16. ¿Cuál de las siguientes técnicas **no** se puede usar para determinar la causa de una interrupción?

- a. línea de reconocimiento INTA#
- b. interrupciones vectorizadas
- c. consulta de estado, o polling
- d. múltiples líneas de interrupción INT1#, INT2#... con un dispositivo en cada línea

17. Señale cuál de las siguientes opciones es una técnica para llevar a cabo la transferencia de datos entre el computador y los dispositivos de E/S externos:

- a. E/S por flanco
- b. E/S programada
- c. Acceso indirecto a memoria
- d. E/S por nivel

**18.** ¿Cuál de las siguientes afirmaciones es **\*FALSA\***?

- a. La operación de lectura de una celda DRAM es destructiva
  - b. Las memorias DRAM son en general más lentas que las SRAM
  - c. Una celda DRAM no pierde la información al desconectar la alimentación
  - d. Las memorias DRAM presentan generalmente una capacidad de almacenamiento mayor que las SRAM
- 

**19.** La memoria DRAM:

- a. Se inventó en la década de los 2000
  - b. Necesita 6 transistores por cada celda
  - c. Se denomina dinámica porque su contenido puede alterarse, al contrario que la SRAM
  - d. Es más densa que la memoria SRAM
- 

**20.** Indique cuál es la dirección de la instrucción **mov** en el siguiente desensamblado, donde se ha borrado parte de la dirección

0804xxxx: 74 12 je 08048391  
0804xxxx: b8 00 00 00 00 mov \$0, %eax

- a. 08048391 + 12 = 08048403
  - b. 08048391 - 12 = 08048379
  - c. 0804837d
  - d. 0804837f
- 

**21.** Dada la siguiente declaración en lenguaje C, una estructura de este tipo podría ocupar, bien sea en un sistema Linux IA32 o bien en uno x86-64, un total de...

```
struct a{  
    int i;  
    double d;  
    char c;  
    short s; };
```

- a. 18 B
  - b. 20 B
  - c. 22 B
  - d. 24 B
- 

**22.** Dado un camino de datos concreto, un posible formato de microprogramación se caracteriza como horizontal o vertical según tenga más o menos (señalar la respuesta **\*FALSA\***)

- a. codificación
  - b. solapamiento
  - c. microbifurcaciones
- 

d. longitud relativa de microinstrucción

---

**23.** En una unidad de control microprogramada con formato de microinstrucciones vertical, un subcampo que deba especificar 16 señales de control codificadas de tal forma que pueda activarse sólo una o ninguna habrá de tener una anchura mínima de

- a. 4 bits
  - b. 5 bits
  - c. 16 bits
  - d. 17 bits
- 

**24.** Motivos que impiden que la ganancia (aceleración) de un cauce segmentado sea ideal (señale la respuesta **\*FALSA\***)

- a. registros de acople (coste de la segmentación)
  - b. fragmentación desigual (duración desigual de etapas)
  - c. riesgos (*hazards*)
  - d. cola de instrucciones (precaptación)
- 

**25.** Un procesador de 1 GHz sin segmentación de cauce tarda 4 ns en ejecutar 4 instrucciones. ¿Cuánto tardaría en ejecutar 9 instrucciones una versión de dicho procesador con segmentación de cauce de 4 etapas si no existiera ningún retraso en ninguna de las instrucciones?

- a. 2 ns
  - b. 3 ns
  - c. 4,5 ns
  - d. 9 ns
- 

**26.** Respecto al salto retardado y al salto anulante, ¿cuál permite que se ejecute la siguiente instrucción, y cuál no?

- a. el retardado ejecuta la siguiente instrucción (con el correspondiente retraso), el anulante no la ejecuta (de hecho la anula)
  - b. el retardado la ejecuta sólo si se cumple la condición de salto, el anulante sólo si no se cumple
  - c. el retardado la ejecuta sólo si no se cumple la condición de salto, el anulante no la ejecuta nunca
  - d. el retardado la ejecuta siempre, el anulante la ejecuta sólo si se cumple la condición de salto
-

**27.** Sobre la E/S mapeada en memoria podemos decir que:

- a. Usa el espacio común de direccionamiento para acceder a puertos de E/S
  - b. La CPU necesita el pin IO/M#
  - c. Dispone de instrucciones especiales de E/S
  - d. Todas las respuestas anteriores son falsas
- 

**28.** ¿Cuál de las siguientes tareas **no** es responsabilidad de un circuito de interfaz o controlador de periféricos sencillo?

- a. Adaptar el formato de las señales
  - b. Ajustar la temporización entre el procesador y los dispositivos de E/S
  - c. Recibir señales de control desde el procesador
  - d. Ejecutar el programa de transferencia de información entre el procesador y los dispositivos de E/S
- 

**29.** ¿Cuál es el ancho del bus de direcciones de un chip DRAM de 1G palabra, siendo la longitud de palabra de 16 bits?

- a. 20
  - b. 16
  - c. 30
  - d. 15
- 

**30.** Sea un computador de 32 bits que dispone de una memoria cache de 512 KB y líneas de 64 bytes. ¿Cuántas líneas tiene la cache?

- a. 64
  - b. 1024
  - c. 8192
  - d. 65536
-

**Nombre:**
**DNI:**
**Grupo:**

## Test de Prácticas (4.0p)

**Todas las preguntas son de elección simple sobre 4 alternativas.**

**Cada respuesta vale 0.2p si es correcta, 0 si está en blanco o claramente tachada, -0.06p si es errónea.**

**Anotar las respuestas (a, b, c ó d) en la siguiente tabla.**

1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20

1. ¿Cuál de los siguientes fragmentos es correcto para comenzar un programa en ensamblador que conste de un solo archivo .s?
- a. `.text:`  
`_start:`
  - b. `.text`  
`.local _start`  
`_start:`
  - c. `.text`  
`.start _global`  
`_start:`
  - d. `.text`  
`.global _start`  
`_start:`
- 
2. Suponga una memoria cache con las siguientes propiedades: Tamaño: 512 bytes. Política de reemplazo: LRU. Estado inicial: vacía (todas las líneas inválidas). Suponga que para la siguiente secuencia de direcciones enviadas a la cache: 1, 2, 4, 8, 16, 32, la tasa de acierto es 0,333. ¿Cuál es el tamaño de línea de la cache?
- a. 4 bytes
  - b. 8 bytes
  - c. 16 bytes
  - d. 32 bytes
- 
3. En la convención cdecl estándar para arquitecturas x86 de 32 bits, cuál de las siguientes afirmaciones es cierta:
- a. Los 6 primeros parámetros se pasan a través de registros
  - b. Solamente es necesario salvar el registro EAX
- 
- c. Los registros EBX, ESI y EDI son salvados por el invocante
- d. Ninguna de las anteriores es cierta
- 
4. ¿Cuál es el popcorn (peso Hamming, nº de bits activados) del número 0x10101010?
- a. 4
  - b. 8
  - c. 16
  - d. 32
- 
5. Compilar de fuente C a ejecutable usando sólo as y ld, sin gcc...
- a. Se puede, repartiendo entre as y ld los modificadores (switches) que corresponda
  - b. Basta usar ld, con los modificadores de gcc que corresponda, y añadiéndole el runtime de C
  - c. Se puede, repartiendo modificadores entre as y ld, y añadiendo al comando ld el runtime de C
  - d. No se puede
- 
6. La función gettimeofday() en la práctica de popcorn y parity se utiliza para
- a. Comparar las duraciones de las distintas soluciones del programa
  - b. Imprimir la fecha y hora
  - c. Cifrar el código en función de la hora actual
  - d. Cronometrar lo que tarda el usuario en pulsar una tecla
- 
7. ¿Cuál de los siguientes registros tiene que ser salvaguardado (si va a modificarse) dentro de

una subrutina según la convención cdecl para IA32?

- a. ECX
  - b. EAX
  - c. EBP
  - d. EDX
- 

8. ¿Qué hace gcc -O1?

- a. Compilar .s → .o (fuente ASM a objeto)
  - b. **Compilar con optimización**
  - c. Compilar .c → .o (fuente C a objeto)
  - d. Compilar .c → .s (C → ASM sin generar objeto)
- 

9. Dada la siguiente definición de datos:

```
lista: .int 0x10000000, 0x50000000,  
       0x10000000, 0x20000000  
longlista: .int (. - lista)/4  
resultado: .quad 0x123456789ABCDEF  
formato: .ascii "%llu=%llx hex\n\n\0"
```

La instrucción para copiar la dirección de memoria donde comienza lista en el registro EBX es:

- a. movl lista, %ebx
  - b. **movl \$lista, %ebx**
  - c. movl (lista), %ebx
  - d. movl \$lista, (%ebx)
- 

10. En la práctica "media" se programa la suma de una lista de 32 enteros de 4 B para producir un resultado de 8 B, primero sin signo y luego con signo. Si la lista se rellena con el valor 0x8000 0000, ¿en qué se diferencian los resultados de ambos programas?

- a. no se diferencian
  - b. en uno ocupa 32 bits, en otro 64 bits
  - c. **en uno los 16 bits superiores son 0xFFFF, en el otro no**
  - d. en uno los 16 bits inferiores son 0x0000, en el otro no
- 

11. En la práctica "media" se pide sumar una lista de 32 enteros CON signo de 32bits en una plataforma de 32bits sin perder precisión, esto es, evitando desbordamiento. Un estudiante entrega un programa que se diferencia de la versión recomendada en el siguiente bucle, en particular en la instrucción adc.

```
bucle:  
    mov (%ebx,%esi,4), %eax  
    cltd  
    add %eax, %edi  
    adc %eax, %ebp
```

```
inc %esi  
cmp %esi, %ecx  
jne bucle
```

Esta versión de la suma CON signo

- a. produce siempre el resultado correcto
  - b. fallaría con lista: .int 1, 1, 1, 1, ...
  - c. fallaría con lista: .int -1,-1,-1,-1, ...
  - d. no siempre produce el resultado correcto, pero el error no se manifiesta en los ejemplos propuestos, o se manifiesta en ambos
- 

12. En la práctica "media" un estudiante usa el siguiente bucle para acumular la suma en EBP:EDI antes de calcular la media y el resto

**bucle:**

```
    mov (%ebx,%esi,4), %eax  
    cltd  
    add %eax, %edi  
    adc %edx, %ebp  
    jnc nocarry  
    inc %edx  
nocarry:  
    inc %esi  
    cmp %esi,%ecx  
    jne bucle
```

Este código es una mezcla de las soluciones recomendadas para suma sin signo y para suma con signo. Estando bien programado todo lo demás, este código

- a. produce siempre el resultado correcto
  - b. fallaría con lista: .int 0,1,2,3
  - c. fallaría con lista: .int -1,-2,-4,-8
  - d. no siempre produce el resultado correcto, pero el error no se manifiesta en los ejemplos propuestos, o se manifiesta en ambos
- 

13. ¿Cuál es el popcorn (peso Hamming, nº de bits activados) de una lista de N números inicializada con los valores 0..N-1?

- a.  $(N-1)*N/2$
  - b.  $N*(N+1)/2$
  - c. si N es par,  $N*(N/2)$
  - d. si N es potencia de 2,  $\log_2(N)*N/2$**
- 

14. ¿Cuál es la paridad (XOR "lateral" de todos los bits) del número 199?

- a. 0
  - b. 1**
  - c. 2
  - d. 4
-

15. Comparando los popcounts (pop(199) vs. pop(99)) y paridades (par(199) vs. par(99)) de los números 199 y 99, se verifica que

- a.  $\text{pop}(199) > \text{pop}(99)$
  - b.  $\text{par}(199) < \text{par}(99)$
  - c.  $\text{pop}(199) < \text{par}(99)$
  - d.  $\text{par}(199) > \text{pop}(99)$
- 

16. ¿Cuál es la suma de paridades (suma de los XOR "laterales" de los bits de cada número) de una lista de N números inicializada con los valores 0..N-1?

- a.  $(N-1)*N/2$
  - b.  $N*(N+1)/2$
  - c. si N es par,  $N/2$
  - d. si N es potencia de 2,  $(\log_2(N)*N)/2$
- 

17. En la práctica “popcount/paridad”, para cronometrar sistemáticamente las diversas versiones necesitamos una función crono() a la que se le pueda pasar como argumento cuál versión queremos cronometrar. En lenguaje C esto se puede hacer con punteros a funciones. Sabiendo que todas las versiones devuelven un valor entero, el prototipo de la función crono() debería ser:

- a. void crono( int \* func (), char\* msg);
  - b. void crono( int (\* func)(), char\* msg);
  - c. void crono((int \*)func (), char\* msg);
  - d. void crono( int \* func , char\* msg);
- 

18. Para corregir la práctica “bomba digital”, un profesor dispone de 26 ejecutables (y la lista de claves correspondientes) para asignar en el Laboratorio una bomba distinta a cada estudiante. Cuando un estudiante diga que la ha resuelto el profesor exigirá ver al estudiante ejecutando la bomba y tecleando la contraseña y el pin correctos para comprobar que no explota, para así anotarla como resuelta y que le puntúe al estudiante.

Un estudiante (usando ordenador del Laboratorio con Ubuntu 10.04) dice que ha resuelto su bomba, y cuando el profesor pide que se tecleen las claves, el ddd se “bloquea” y le empieza a “parpadear” al estudiante. Para poder puntuar, lo más recomendable para el estudiante sería...

- a. teclear las claves (contraseña y pin), aunque ddd esté “bloqueado” y “parpadeando”

- b. probar en orden los remedios básicos: pulsar  $<\text{Ctrl}>-\text{C}$  varias veces, pulsar  $<\text{Enter}>$  repetidamente, comprobar que está seleccionada “Machine Code Window”, teclear “info line main”, y si todo falla, ejecutar “`rm -rf ~/.ddd`”
  - c. matar la ventana ddd, abrir un terminal con un shell, ejecutar la bomba desde línea de comandos y teclear las claves
  - d. reinstalar un paquete ddd más actualizado usando “`sudo apt-get install`”
- 

19. Suponer el mismo contexto de la pregunta anterior, donde el profesor tiene una lista de las 26+26 claves (contraseñas y pines)

Un estudiante dice que ha resuelto su bomba, y cuando el profesor pide que se tecleen las claves, la contraseña coincide con la que tiene anotada el profesor, pero el pin no, y de todas formas la bomba no explota. Debería hacerse lo siguiente:

- a. la bomba tiene que puntuarle al estudiante porque no ha explotado
  - b. el profesor puede pedirle que vuelva a descargar la bomba original e intente repetir con ese ejecutable las claves que acaba de teclear
  - c. la bomba debe marcarse como inválida y no hay que hacer más comprobaciones
  - d. no puede suceder lo que dice el enunciado, y en ningún caso el profesor tiene derecho a hacer comprobaciones adicionales como pedir que se vuelva a descargar la bomba
- 

20. En la práctica de la cache, el código de “line.cc” incluye la sentencia

```
for (unsigned long long line=1;
     line<=LINE; line<<=1) { ... }
```

¿Qué objetivo tiene la expresión  $\text{line} <<= 1$ ?

- a. salir del bucle si el tamaño de línea se volviera menor o igual que 1 para algún elemento del vector
  - b. duplicar el tamaño del salto en los accesos al vector respecto a la iteración anterior
  - c. volver al principio del vector cuando el índice excede la longitud del vector
  - d. sacar un uno (1) por el stream line
-

Nombre:	
DNI:	Grupo:

### Examen de Problemas (3,0 p)

- 1. Ensamblador** (0.75 puntos). La principal secuencia de instrucciones de la vulnerabilidad Meltdown es la siguiente:

```
# rcx = dirección del kernel a acceder
# rbx = array en el espacio de usuario
retry:
    mov (%rcx), %al
    shl $0xc, %rax
    jz retry
    mov (%rbx,%rax), %rbx
```

Suponga que inicialmente RAX vale 0, RCX vale 0x601037 (en este caso accedemos a un dato en nuestro espacio de usuario, no a una dirección prohibida del kernel), RBX vale 0x601038. A partir de la dirección 0x601037 de memoria, incluida, más de 1 M bytes están inicializados al valor 0x10. Rellene las celdas en blanco de la siguiente tabla:

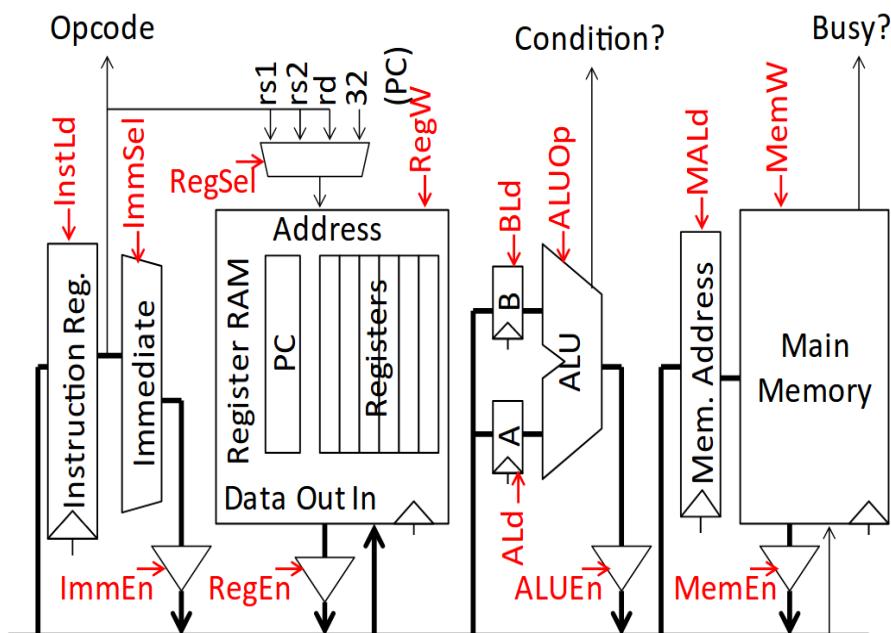
Instrucción	Fuente			Destino		
	Modo de direccio-namiento	Dirección de memoria	Contenido o valor	Modo de direccio-namiento	Contenido o valor antes de ejecutar la instrucción	Contenido o valor después de ejecutar la instrucción
<code>mov (%rcx), %al</code>						
<code>shl \$0xc,%rax</code>		N/A				
<code>mov (%rbx,%rax), %rbx</code>						

- 2. Ensamblador.** (0.75 puntos). Escriba una función en ensamblador de IA32 que convierta un vector de 100 shorts de big endian a little endian. La función recibirá como argumento un puntero al vector de shorts almacenados en big endian. Al terminar, todos los shorts en el vector deben estar almacenados siguiendo el criterio de ordenación de bytes little endian.

**3. Unidad de control** (0.5 puntos). El repertorio de instrucciones de la arquitectura RISC-V contiene la instrucción máquina BEQ (saltar si igual) que compara dos registros rs1 y rs2 y salta a la etiqueta destino sumando una cantidad inmediata, desplazamiento u offset al contador de programa PC. El formato de esta instrucción es el siguiente:

31	30	25-24	20-19	15-14	12-11	8	7	6	0
imm[12]	imm[10:5]	rs2	rs1	funct3	imm[4:1]	imm[11]		opcode	
1 offset[12,10:5]	6 src2	5 src1	5 BEQ/BNE	3 offset[11,4:1]	4 1		7 BRANCH		

imm[0] siempre es 0, de forma que se puede saltar en un rango de  $\pm 4$  KiB. Una posible implementación del RISC-V con un único bus es la siguiente:



Escriba las microinstrucciones (en pseudo-código) para implementar la captación de instrucción y la ejecución de la instrucción BEQ, suponiendo que existen las operaciones de la ALU A-B, A+B y A+4 entre otras, una microinstrucción “wait” que espera hasta que la memoria está lista, y que Immediate extiende el signo de 13 bits del campo imm de la instrucción a 32 bits.

**4. Configuración de memoria** (0.5 puntos). Queremos conectar un microprocesador de 8 bits a un sistema de memoria constituido por una ROM de 8 KB y una RAM de 8 KB. El microprocesador dispone de las patillas MREQ#, RD#, WR#, A15-A0 y D7-D0. Para la ROM tenemos un chip con las patillas CE#, OE#, A0-A12 y D0-D7. Para la RAM tenemos un chip con las patillas CS#, OE#, WE#, A0-A12, D0-D7. Para la conexión disponemos de un decodificador con 1 patilla de habilitación E#, 3 patillas de selección A, B, C y 8 patillas de salida Y7-Y0.

Dibuje un esquema con la memoria y su conexión al microprocesador, de tal modo que la ROM esté ubicada en las direcciones 0x0000-0x1FFF y la RAM en las direcciones 0x2000-0x3FFF.

**5. Memoria cache** (0.5 puntos). Las características de la jerarquía de memoria del microprocesador AMD Ryzen Threadripper 1950X, de 16 núcleos, son las siguientes:

Cache line size	64 B
Level 1 cache size	16 x 64 KB 4-way set associative instruction caches 16 x 32 KB 8-way set associative data caches
Level 2 cache size	16 x 512 KB 8-way set associative unified caches
Level 3 cache size	4 x 8 MB 16-way set associative shared caches
Physical memory	1 TB

Indique el nombre y tamaño en bits de los campos de dirección usados para la política de correspondencia, así como el tamaño total en bits de todas las memorias de etiquetas, tamaño total en bits de todas las memorias de instrucciones/datos, y porcentaje de espacio de etiquetas respecto a instrucciones/datos para cada uno de los dos casos siguientes:

- a. Dirección física de memoria principal desde el punto de vista de una L1 de instrucciones:

Tamaño total en bits ocupado por todas las etiquetas en directorios L1 (instrucciones):

Tamaño total en bits ocupado por todas las instrucciones en L1 (instrucciones):

Porcentaje de espacio ocupado por etiquetas respecto a instrucciones en L1 (instrucciones):

- b. Dirección física de memoria principal desde el punto de vista de una L1 de datos:

Tamaño total en bits ocupado por todas las etiquetas en directorios L1 (datos):

Tamaño total en bits ocupado por todos los datos en L1 (datos):

Porcentaje de espacio ocupado por etiquetas respecto a datos en L1 (datos):

### Test de Teoría (3.0p)

1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30
c	a	b	c	d	a	b	c	d	a	b	c	d	d	a	a	b	c	d	d	d	c	b	d	b	d	a	d	c	

### Test de Prácticas (4.0p)

1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20
d	b	d	a	d	a	c	b	b	c	b	a	d	b	a	c	b	c	b	b

### Examen de Problemas (3.0p)

#### 1. Ensamblador (0.75 puntos).

Se puntuá **0.05p** por celda (las tres celdas de modo dir. registro cuentan como una, total **0.05 x 15 = 0.75p**)

Instrucción	Fuente			Destino		
	Modo de direccio-namiento	Dirección de memoria	Contenido o valor	Modo de direccio-namiento	Contenido o valor antes de ejecutar la instrucción	Contenido o valor después de ejecutar la instrucción
<code>mov (%rcx),%al</code>	Indirecto (a través de registro)	0x60 1037	0x10	Registro	0	0x10
<code>shl \$0xc,%rax</code>	Inmediato (constante literal)	N/A	0x0c		0x10 (=16)	0x1 0000 (=65536)
<code>mov (%rbx,%rax),%rbx</code>	Indexado respecto a base	0x61 1038 + 0x01 0000 = 0x61 1038	0x10		0x60 1038	0x1010 1010 1010 1010

#### 2. Ensamblador (0.75 puntos).

Se supone convención cdecl, a falta de más información sobre cómo se pasa el argumento puntero  
 Infinitud soluciones válidas entre 10-15 líneas, se valora brevedad, claridad, simplicidad... tal vez eficiencia.  
 Si programa correcto de 10-15 líneas, nota completa. Pasar de 15 líneas puntuá negativamente.

Correspondería a aproximadamente **0.05p** por línea/instrucción (**0.05 x 15 = 0.75p**)

**little2big:**

```
push %ebp
mov %esp, %ebp

mov 8(%ebp), %eax # arg. ptr
mov $100, %ecx # cont/idx
```

**big2little:**

```
push %ebp
mov %esp, %ebp

mov 8(%ebp), %eax
xor %ecx, %ecx
```

**ALTERNATIVAS CUERPO BUCLE**

```
mov (%eax,%ecx,2), %dl
mov 1(%eax,%ecx,2), %dh
mov %dh, (%eax,%ecx,2)
mov %dl,1(%eax,%ecx,2)
```

```
mov (%eax,%ecx,2), %dh
mov 1(%eax,%ecx,2), %dl
mov %dx, (%eax,%ecx,2)
```

```
mov (%eax,%ecx,2), %dx
mov %dh, (%eax,%ecx,2)
mov %dl,1(%eax,%ecx,2)
```

```
mov (%eax,%ecx,2), %dx
xchg %dh, %dl
mov %dx, (%eax,%ecx,2)
```

```
dec %ecx      # 100..1
jnz .bucle
pop %ebp
ret
```

```
inc %ecx      # 0..99
cmp $100, %ecx
jnz .bucle
pop %ebp
ret
```

### 3. Unidad de Control (0.5 puntos).

Solución (inspirada en <http://slideplayer.com/slide/11064764/>)

Se puntuá **0.05p** por micro-instrucción (aproximadamente 11 micro-instrucciones, una de regalo, total **0.05 x 10 = 0.50p**)

```

fetch:      MA:=PC; A:=PC
            PC:=A+4
            wait
            Inst:=Mem[MA]
            goto f(IR)
            ...

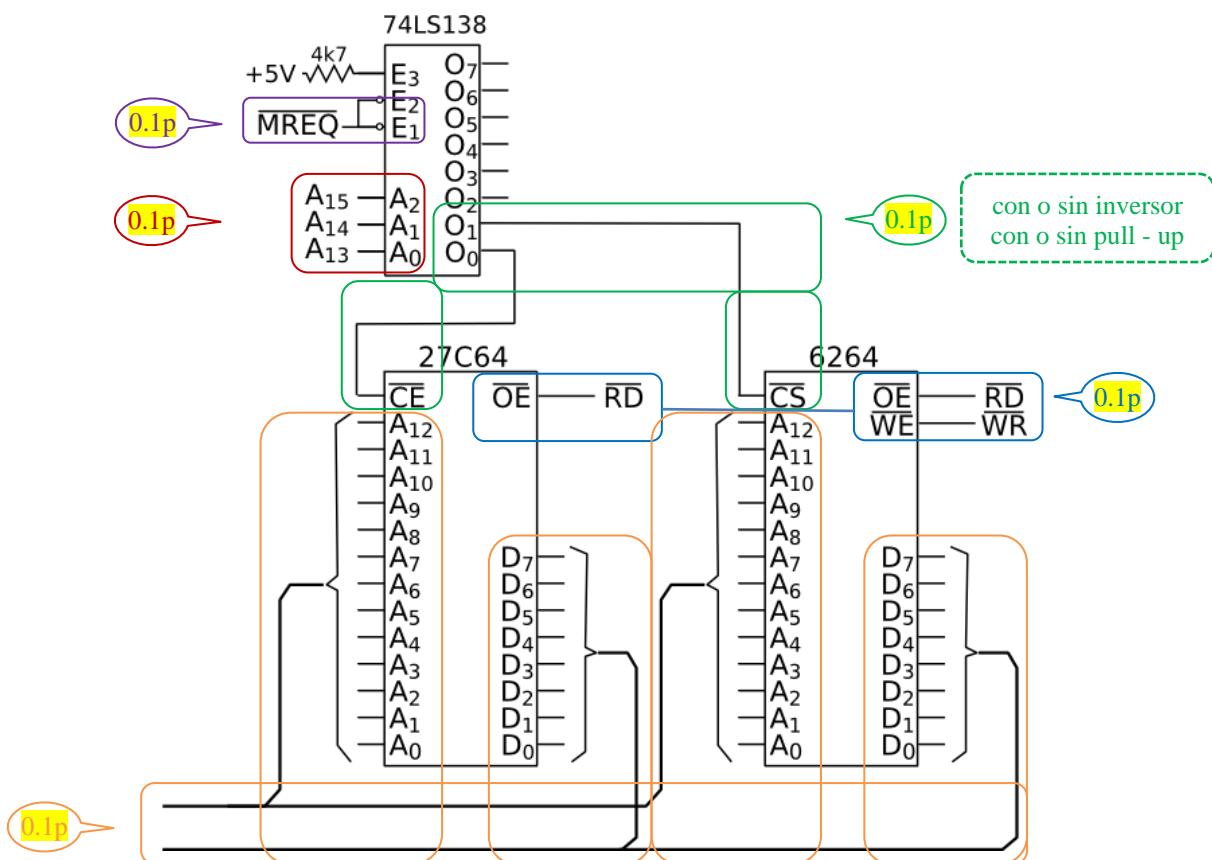
BEQ:       A:=Reg[rs1]
            B:=Reg[rs2]
            if (A-B != 0) goto fetch
            A:=PC
            B:=Immediate
            PC:=A+B; goto fetch

```

### 4. Configuración de memoria (0.5 puntos).

Solución (ver <https://www.eevblog.com/forum/beginners/z80-single-board-memory-bank-switching/>)

Aproximadamente **0.1p** por zona (**MREQ#**, **A<sub>15-13</sub>**, **CE#/CS#**, **OE#/WE#**, **A<sub>12-0</sub>/D<sub>7-0</sub>**)



### 5. Memoria cache (0.5 puntos).

$64 \text{ B/línea} = 2^6 \text{ B/línea} \Rightarrow 6 \text{ bits campo offset (desplazamiento, byte, ...)}$

L1i  $16 \times 64\text{KB}$  4 vías  $\Rightarrow 4$  líneas/conjunto L1i

L1d  $16 \times 32\text{KB}$  8 vías  $\Rightarrow 8$  líneas/conjunto L1d

MP 1TB  $= 2^{40}$  B  $\Rightarrow 40$  bits dirección física

a) L1 instrucciones (0.25p)

$$L1i: 64 \text{ KB} / 64 \text{ B/línea} = 2^{16} \text{ B} / 2^6 \text{ B/línea} = 2^{10} \text{ líneas} (=1024)$$

1024 líneas / 4 vías =  $2^{10}$  líneas /  $2^2$  líneas/conjunto =  $2^8$  conjuntos  $\Rightarrow$  8 bits campo conjunto  
resto bits: etiqueta = 40 - 8 - 6 = 26 bits campo etiqueta

Dirección física de memoria principal desde el punto de vista de L1i: (0.10p = 0.05p + 0.025p +0.025p)

etiqueta (26)	conjunto (8)	byte (6)
---------------	--------------	----------

Tamaño total en bits ocupado por todas las etiquetas en directorios L1i: (0.05p)

16 caches • 1024 líneas/cache • 26 bits/etiqueta =  $2^4 \times 2^{10} \times 26$  bits = **2<sup>14</sup>x26 bits = 425 984 bits**  
alternativamente, una sola cache L1i = 1024 líneas • 26 bits/etiqueta = **26 Kbits**

Tamaño total en bits ocupado por todos los datos/instrucciones en L1i: (0.05p)

16 caches • 64KB/cache • 8 bits/B =  $2^4 \times 2^{16} \times 2^3$  bits = **2<sup>23</sup> bits = 8Mbits = 8 388 608 bits**  
alternativamente, una sola cache L1i =  $2^{16} \times 2^3$  bits = **2<sup>19</sup> bits = 512 Kbits**

Porcentaje Etiquetas / (Datos/Instrucciones) = 425 984 / 8 388 608 = **5.08% (0.05p)**

alternativamente, una sola cache L1i: 26Kb / 512Kb = **5.08%**

b) L1 datos (0.25p)

$$L1d: 32 \text{ KB} / 64 \text{ B/línea} = 2^{15} \text{ B} / 2^6 \text{ B/línea} = 2^9 \text{ líneas} (=512)$$

512 líneas / 8 vías =  $2^9$  líneas /  $2^3$  líneas/conjunto =  $2^6$  conjuntos  $\Rightarrow$  6 bits campo conjunto  
resto bits: etiqueta = 40 - 6 - 6 = 28 bits campo etiqueta

Dirección física de memoria principal desde el punto de vista de L1d: (0.10p = 0.05p + 0.025p +0.025p)

etiqueta (28)	conjunto (6)	byte (6)
---------------	--------------	----------

Tamaño total en bits ocupado por todas las etiquetas en directorios L1d: (0.05p)

16 caches • 512 líneas/cache • 28 bits/etiqueta =  $2^4 \times 2^9 \times 28$  bits = **2<sup>13</sup>x28 bits = 229 376 bits**  
alternativamente, una sola cache L1d = 512 líneas • 28 bits/etiqueta = **14 Kbits**

Tamaño total en bits ocupado por todos los datos/instrucciones en L1d: (0.05p)

16 caches • 32KB/cache • 8 bits/B =  $2^4 \times 2^{15} \times 2^3$  bits = **2<sup>22</sup> bits = 4Mbits = 4 194 304 bits**  
alternativamente, una sola cache L1d =  $2^{15} \times 2^3$  bits = **2<sup>18</sup> bits = 256 Kbits**

Porcentaje Etiquetas / (Datos/Instrucciones) = 229 376 / 4 194 304 = **5.47% (0.05p)**

alternativamente, una sola cache L1d: 14Kb / 256Kb = **5.47%**

**Nombre:**
**DNI:**
**Grupo:**

### Test de Teoría (3.0p)

**Todas las preguntas son de elección simple sobre 4 alternativas.**

**Cada respuesta vale 0.1p si es correcta, 0p si está en blanco o claramente tachada, -0.03p si es errónea.**

**Anotar las respuestas (a, b, c ó d) en la siguiente tabla.**

1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30

1. ¿Cuál es el valor mínimo (más negativo) que puede tomar un entero de 32 bits en complemento a dos? (el punto se usa como separador)

- a. -2.147.483.647
- b. -2.147.483.648
- c. -4.294.967.295
- d. -4.294.967.296

2. ¿Cómo se representa el valor -1 como entero con signo en 14 bits?

- a. 0xFFFF
- b. 0x3FFF
- c. las respuestas anteriores no son válidas porque usan hexadecimal; habría que usar binario
- d. no se puede porque 14 no es múltiplo de 4

3. ¿Cuál de las siguientes no es una unidad de la arquitectura Von Neumann?

- a. Unidad central de proceso
- b. Memoria principal
- c. Sistema de entrada/salida
- d. Núcleo del sistema operativo

4. ¿Cuál de las siguientes afirmaciones es verdadera?

- a. La arquitectura Von Neumann en la que se basan los computadores tradicionales consiste en tener los datos separados de las instrucciones en memorias distintas.
- b. El registro de estado es un registro transparente al usuario, ya que éste no

puede utilizarlo en las instrucciones máquina.

- c. El registro de instrucción es un registro transparente al usuario, ya que éste no puede utilizarlo en las instrucciones máquina.
- d. La unidad de control necesita como entrada el registro contador de programa, para saber cuál es la instrucción que debe ejecutar a continuación.

5. ¿Qué es el lenguaje máquina?

- a. Conjunto de datos binarios que representan señales eléctricas internas de la unidad de control de un microprocesador.
- b. Conjunto de sentencias en un lenguaje escrito que se utilizan para generar programas codificados en lenguaje ensamblador.
- c. Conjunto formado por las siglas asignadas a las instrucciones del repertorio de instrucciones más un conjunto de directivas que facilitan la generación del código binario.
- d. Conjunto de instrucciones en formato binario que entiende un determinado procesador.

6. ¿Cuál de los siguientes elementos **no** forma parte de la Arquitectura del Repertorio de Instrucciones (ISA)?

- a. Descripción del espacio de direccionamiento de la memoria y de la E/S.

- b. Descripción de los campos de bits en los que están organizadas conceptualmente las microinstrucciones.
- c. Descripción de los registros de datos, registros de estado y control.
- d. Descripción de los tipos de datos sobre los que opera el lenguaje máquina.
- 
7. ¿Cuál de las siguientes definiciones de modos de direccionamiento es **\*incorrecta\***?
- a. Inmediato: el dato está codificado dentro de la propia instrucción, en uno de los campos en los que se divide el formato de instrucción.
- b. Registro: el dato se encuentra en un registro de propósito general.
- c. Directo: la dirección se calcula como la suma de un dato codificado en la propia instrucción y el contenido de un registro de propósito general.
- d. Indirecto: el dato está contenido en una posición de memoria que es apuntada por un registro de propósito general.
- 
8. Respecto a los registros enteros en arquitectura IA32 de 32bits (x86)
- a. Se puede acceder a 8, y en cada uno de esos 8 registros enteros, se puede acceder a todos los 32 bits (p.ej. EAX), a los 16 bits menos significativos (p.ej. AX) ó a los 8 LSBs (p.ej. AL)
- b. Se puede acceder a 8, y en cada uno de esos 8 registros enteros, se puede acceder a todos los 32 bits (p.ej. EAX), a los 16 bits menos significativos (p.ej. AX), a los 8 LSBs (p.ej. AL) o a los bits 8-15 (p.ej. AH)
- c. Se puede acceder a 8 de cada tamaño (32, 16, 8 bits), aunque no todos los registros tienen versión de 8 y 16 bits
- d. No hay distintos tamaños, son sólo registros de 32 bits, como corresponde a dicha arquitectura
- 
9. ¿Cuál de las siguientes instrucciones es errónea? (sale mensaje de error al intentar ensamblar):
- a. movw %dx, (%eax)
- b. movb \$0xFF, (%dl)
- c. movswl (%eax), %edx
- d. movzbl %dl, %eax
- 
10. ¿Qué modo de direccionamiento usa el operando fuente en la instrucción mov (%rcx), %al?
- a. Directo a memoria
- b. Indirecto a memoria a través de registro
- c. Registro
- d. Inmediato
- 
11. Si el contenido del registro %rax es 0x10 antes de ejecutar la instrucción shl \$0xc,%rax, ¿cuánto es su contenido tras ejecutarla?
- a. 0x10000
- b. 0x1000
- c. 0x4000
- d. 0x800
- 
12. En el fragmento de código
- ```
804854e:e8 3d 06 00 00 call 8048b90  
8048553:50          pushl %eax
```
- la instrucción call suma al contador de programa la cantidad:
- a. 0x0000063d
- b. 0x08048553
- c. 0x0804854e
- d. 0x50
- 
13. ¿Cuál de los siguientes registros x86-64 es distinto del resto en convenio de uso? (salva-invocante/invocado)
- a. RBX
- b. RCX
- c. RSI
- d. R8
- 
14. Respecto a requisitos de alineamiento de structs en gcc/IA32 x86 y x86-64, una de las siguientes afirmaciones es **\*FALSA\***
- a. en x86 Linux alinea double a 4x
- b. en x86 Linux alinea long double a 4x
- c. en x86-64 Linux alinea double a 8x
- d. en x86-64 Linux alinea float a 8x
- 
15. Se definen las variables, unión y función C siguientes:
- ```
float      f1;  
unsigned   u1=0x80000000;  
float      f2;  
typedef union {  
    float f;  
    unsigned u;  
} bit_float_t;
```

```

float bit2float(unsigned u) {
    bit_float_t arg;
    arg.u = u;
    return arg.f;
}

```

¿Cuál afirmación es verdadera?

- a. Si asignamos f1=bit2float(u1); entonces f1== 2147483648.00
  - b. Si asignamos f1=bit2float(u1); entonces f1== -0.0
  - c. Si asignamos f2= (float)u1 ; entonces f2== 4294967296.00
  - d. Si asignamos f2= float(u1); entonces f2== 0.0
- 

**16.** Convertir un vector de 100 shorts de formato little endian a formato big endian consiste en:

- a. Intercambiar el elemento 0 del vector con el 99, el 1 con el 98, el 2 con el 97 y así sucesivamente.
  - b. Intercambiar el elemento 0 del vector con el 1, el 1 con el 2, el 3 con el 4 y así sucesivamente.
  - c. Cambiar el orden en memoria de los 4 bytes de cada elemento, es decir, en cada elemento intercambiar el byte 0 con el 3 y el 1 con el 2.
  - d. Cambiar el orden en memoria de los 2 bytes de cada elemento, es decir, en cada elemento intercambiar el byte 0 con el 1.
- 

**17.** Motivos que impiden que la ganancia (aceleración) de un cauce segmentado sea ideal (señalar la respuesta **\*FALSA\***)

- a. registros de acople (coste de segmentación)
  - b. fragmentación desigual (duración de etapas)
  - c. riesgos (hazards)
  - d. cola de instrucciones (precaptación)
- 

**18.** Un procesador de 1GHz tarda 4ns en realizar 4 instrucciones sin realizar segmentación de cauce. ¿Cuanto tardaría en realizar 9 instrucciones con segmentación de cauce de 4 etapas si no existiera ningún retraso en ninguna de las instrucciones?

- a. 2 ns
  - b. 3 ns
  - c. 4.5 ns
  - d. 9 ns
- 

**19.** ¿Qué es un controlador de E/S?

- a. Un circuito electrónico que implementa la memoria del computador.
  - b. Un circuito impreso del tipo DIMM.
  - c. Un circuito electrónico que puede guardar temporalmente datos enviados desde el procesador al periférico o viceversa.
  - d. Un bus que permite interconectar distintos periféricos entre sí.
- 

**20.** Respecto a la interfaz de E/S, ¿cuál de las siguientes afirmaciones es **\*FALSA\***?

- a. Involucra tareas que se pueden implementar parte en hardware y parte en software.
  - b. Permite configurar el funcionamiento del periférico en un momento determinado, y además conocer su estado.
  - c. Puede guardar temporalmente en registros internos tanto datos generados por el periférico para ser enviados al procesador, como datos que son enviados desde el procesador al periférico.
  - d. Una interfaz de entrada recibe los datos desde el procesador y los transforma y envía al periférico en formato digital.
- 

**21.** ¿Cuál de las siguientes características corresponde a E/S mapeada en memoria?

- a. Determinadas zonas del espacio de direccionamiento del procesador se asignan por convenio a controladores de E/S.
  - b. Un ejemplo de mecanismo de E/S mapeada en memoria es la instrucción IN de los procesadores Intel.
  - c. Una misma dirección se usa alternativamente para E/S y para memoria en distintos momentos de ejecución de un programa.
  - d. Un pin IO/M# del procesador permite distinguir si accedemos a E/S o a memoria.
- 

**22.** ¿Cuál de las siguientes afirmaciones es **\*FALSA\***?

- a. La consulta del estado del dispositivo por parte de la CPU se suele hacer con E/S programada (salvo con dispositivos que siempre están listos para transferir) y con E/S por IRQ (cuando se usa polling para determinar el origen de la IRQ).
- b. Se suele avisar a la CPU (mediante una IRQ) de que debe realizar alguna tarea, tanto en E/S por IRQ (obligatoriamente, la tarea es la transferencia) como en E/S por DMA (optativamente, el controlador DMA puede avisar de que acabó).

- c. Sólo E/S por DMA libera a la CPU de realizar la consulta de estado del dispositivo de E/S.
- d. Sólo E/S por DMA libera a la CPU de realizar la transferencia de los datos de E/S.

23. La instrucción máquina DI (Disable Interrupts), conocida como CLI (Clear Interrupt Flag) en x86, se utiliza para desactivar:

- a. Todas las interrupciones enmascarables
- b. Las interrupciones de inferior o igual prioridad a una dada
- c. Determinados niveles de interrupción de forma selectiva
- d. Las interrupciones software

24. ¿Cuál de los siguientes es un registro de un controlador de DMA?

- a. IR (Instruction Register)
- b. PC (Program Counter)
- c. SP (Stack Pointer)
- d. WC (Word Count)

25. El ancho de banda de memoria es:

- a. el número de bits que se pueden transferir entre ésta y la CPU en paralelo en una sola operación de lectura o escritura
- b. el número de bytes que se pueden leer/escribir por unidad de tiempo
- c. el tiempo que se tarda en transferir una palabra entre memoria y CPU
- d. el intervalo de frecuencias de reloj permitidas entre memoria y CPU

26. ¿Cuál de las siguientes afirmaciones sobre la memoria DRAM es **\*incorrecta\***?

- a. El principio de funcionamiento de los circuitos electrónicos de la memoria DRAM consiste en cargar o descargar un transistor.
- b. Los bits de memoria se organizan dentro del circuito integrado en forma de matriz de celdas de bit, en la que se pueden diferenciar filas y columnas.
- c. Un transistor en cada celda permite o no permitir circular la corriente eléctrica a través de él. Cuando el transistor no deja pasar la corriente, la información queda almacenada durante un tiempo en el condensador. Cuando el transistor deja pasar corriente, el condensador se carga o se descarga.
- d. Cada celda de memoria está compuesta por un transistor y un condensador y almacena un bit de información.

27. ¿Cuál de las siguientes afirmaciones sobre memorias es correcta?

- a. La memoria cache se construye con tecnología electrónica de tipo DRAM.
- b. La memoria principal se construye con tecnología electrónica de tipo SRAM.
- c. Los chips de memoria DRAM se conectan entre sí en un circuito impreso constituyendo lo que se denomina DIMM.
- d. Las memorias SRAM no son volátiles; es decir, cuando no están alimentadas eléctricamente siguen guardando toda la información.

28. ¿Cuál de los siguientes grupos de señales no se usa en un chip de memoria SRAM?

- a. Selección de filas RAS# y de columnas CAS#.
- b. Datos D<sub>n-1</sub>-D<sub>0</sub>.
- c. Direcciones A<sub>n-1</sub>-A<sub>0</sub>.
- d. Selección de chip CS# y habilitación de escritura WE#.

29. ¿Qué es el tiempo de refresco de memoria?

- a. La cantidad de datos transferidos por segundo entre dos niveles de la jerarquía de memoria.
- b. El tiempo que se tarda en recargar los condensadores que almacenan los bits de datos para que no se pierdan.
- c. El tiempo que transcurre entre la solicitud de una operación en un determinado nivel de la jerarquía de memoria (lectura o escritura) y la recepción de todos los datos solicitados.
- d. El tiempo que tiene que transcurrir entre sucesivas solicitudes de acceso a un determinado nivel de la jerarquía de memoria.

30. ¿Cuál de las siguientes políticas está **\*menos\*** relacionada con la jerarquía memoria?

- a. Política de escritura: determina cómo se actualiza el nivel de la memoria i+1 cuando se ejecutan instrucciones de almacenamiento en el nivel i.
- b. Política de reemplazo: qué bloque se tiene que sustituir (reemplazar) cuando se trae un bloque desde otro nivel.
- c. Política de planificación: en qué orden se ejecutarán los procesos pendientes.
- d. Política de colocación: dónde se almacena un bloque de datos dentro de la memoria.

**Nombre:**
**DNI:**
**Grupo:**

## Test de Prácticas (4.0p)

**Todas las preguntas son de elección simple sobre 4 alternativas.**

**Cada respuesta vale 0.2p si es correcta, 0 si está en blanco o claramente tachada, -0.06p si es errónea.**

**Anotar las respuestas (a, b, c ó d) en la siguiente tabla.**

1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20

1. El punto de entrada de un programa ensamblador en GNU/as Linux x86 se llama

- a. main
  - b. begin
  - c. \_start
  - d. \_\_init
- 

2. En un programa en ensamblador queremos crear espacio para una variable entera var inicializada a 1. La línea que hemos de escribir en la sección de datos es:

- a. .int var 1
  - b. var: .int 1
  - c. .int: var 1
  - d. int var 1
- 

3. En la práctica "media" se pide sumar una lista de 32 enteros SIN signo de 32 bits en una plataforma de 32 bits sin perder precisión, esto es, evitando perder acarreos. Un estudiante entrega la siguiente versión

```
# $lista en EBX, longlista en ECX
suma:
    mov $0, %eax
    mov $0, %edx
    mov $0, %esi
bucle:
    add (%ebx,%esi,4), %eax
    jne nocarry
    inc %edx
nocarry:
    inc %esi
    cmp %esi,%ecx
    jne bucle
    ret
```

Esta función presenta una única diferencia frente a la solución recomendada en clase, relativa al salto condicional.

Esta función suma:

- a. produce siempre el resultado correcto
  - b. fallaría con lista: .int 1,1,1,1, 1,1,1,1, ...
  - c. fallaría con lista: .int 1,2,3,4, 1,2,3,4, ...
  - d. no es correcta pero el error no se manifiesta en los ejemplos propuestos, o se manifiesta en ambos
- 

4. En la misma práctica "media" un estudiante entrega la siguiente versión de suma sin signo:

```
main: .global main
...
call suma
...
mov $1, %eax
mov $0, %ebx
int $0x80
suma:
...
bucle:
...
nocarry:
    inc %esi
    cmp %esi,%ecx
    jne bucle
```

Notar que falta la instrucción ret final. Al desensamblar el código ejecutable se obtiene

```
08048445 <nocarry>:
8048445: 46    inc %esi
8048446: 39 f1 cmp %esi,%ecx
8048448: 75 f5 jne 804843f <bucle>
804844a: 90    nop
804844b: 90    nop
..... 90    nop
804844f: 90    nop
```

```

08048450 <__libc_csu_fini>:
 8048450: 55      push    %ebp
 8048451: 89 e5   mov     %esp,%ebp
 8048453: 5d      pop     %ebp
 8048454: c3      ret

```

Este programa:

- a. está correctamente diseñado, la instrucción ret final es optativa, y no es concebible que quitar el ret cause algún error
  - b. produce un error "Segmentation fault" cuando empieza a acceder a memoria que no le corresponde (EIP)
  - c. funciona bien, pero si pusiéramos en el código fuente primero la definición de suma y luego la de main, el ejecutable terminaría accediendo a memoria que no le corresponde (ESP) y hará "Segmentation fault"
  - d. no se puede marcar ninguna de las opciones anteriores
- 

5. ¿Cuál de las siguientes sumas con signo produce desbordamiento con 32 bits?

- a. 0xFFFFFFFF + 0xFFFFFFFF
  - b. 0x7FFFFFFF + 0xFFFFFFFF
  - c. 0x7FFFFFFF + 0x00000001
  - d. 0xFFFFFFFF + 0x00000001
- 

6. En la práctica “media” se pide sumar una lista de 32 enteros \*con\* signo de 32bits en una plataforma de 32bits sin perder precisión, esto es, evitando overflow. ¿Cuál es el menor valor positivo que repetido en toda la lista causaría overflow con 32bits?

- a. 0x0400 0000
  - b. 0x0800 0000
  - c. 0x4000 0000
  - d. 0x8000 0000
- 

7. En la práctica “media” se pide sumar una lista de 32 enteros \*con\* signo de 32bits en una plataforma de 32bits sin perder precisión, esto es, evitando overflow. ¿Cuál es el mayor valor negativo (menor en valor absoluto) que repetido en toda la lista causaría overflow con 32bits?

- a. 0xffff ffff
  - b. 0xfc00 0000
  - c. 0xfbff ffff
  - d. 0xf000 0000
- 

8. ¿Cuál es el popcorn (peso Hamming, nº de bits activados) del número 29?

- a. 2
  - b. 3
  - c. 4
  - d. 5
- 

9. La práctica "popcount" debía calcular la suma de bits (peso Hamming) de los elementos de un array. Un estudiante entrega la siguiente versión de popcorn4:

```

int popcorn4(unsigned* array, int len){
    int val = 0;
    int i, j;
    unsigned x;
    int res = 0;
    for (i=0; i<len; i++){
        x = array[i];
        val = 0;
        for (j=0; j<8; j++){
            val += x & 0x01010101;
            x >>= 1;
        }
        val += (val>>16);
        val += (val>>8);
        res += val;
    }
    return (res & 0xFF);
}

```

Esta función presenta varias diferencias con la versión "oficial" recomendada en clase, incluyendo la "doble inicialización" de val y la acumulación y retorno de res. Esta popcorn4:

- a. produce siempre el resultado correcto
  - b. fallaría con array={0,1,2,3}
  - c. fallaría con array={1,2,4,8}
  - d. no es correcta pero el error no se manifiesta en los ejemplos propuestos, o se manifiesta en ambos
- 

10. En la misma práctica "popcount" un estudiante entrega la siguiente versión de popcorn4:

```

int popcorn4(unsigned* array, int len){
    int i, j;
    unsigned x;
    int result = 0;
    long val;
    for (i=0; i<len; i++){
        x = array[i];
        val = 0;
        for (j=0; j<8*sizeof(int); j++){
            val += x & 0x01010101;
            x >>= 1;
        }
        val += (val>>16);
        val += (val>>8);
        result += val & 0xFF;
    }
    return result;
}

```

}

Esta función presenta varias diferencias con la versión "oficial" recomendada en clase, incluyendo el tipo de val y las condiciones del bucle for.

Esta función popcorn4:

- a. produce siempre el resultado correcto
- b. fallaría con array={0,1,2,3}
- c. fallaría con array={1,2,4,8}
- d. no es correcta pero el error no se manifiesta en los ejemplos propuestos, o se manifiesta en ambos

11. Comparando los popcounts (pop(129) vs. pop(29)) y paridades (par(129) vs. par(29)) de los números 129 y 29, se verifica que

- a. pop(129) > pop(29)
- b. par(129) > par(29)
- c. pop(129) < par(29)
- d. par(129) < pop(29)

12. La práctica "parity" debía calcular la suma de paridades impar (XOR de todos los bits) de los elementos de un array. Un estudiante entrega la siguiente versión de parity4:

```
int parity4(unsigned* array, int len){  
    int val;  
    int i;  
    unsigned x;  
    int result = 0;  
    for (i=0; i<len; i++){  
        x = array[i];  
        val = 0;  
        asm("\n"  
            "ini3: \n\t"  
            "shr $0x1, %[x] \n\t"  
            "adc $0x0, %[r] \n\t"  
            "test %[x], %[x] \n\t"  
            "jnz ini3 "  
            : [r]"+r"(val)  
            : [x]"r"(x)  
            );  
        result += val & 0x1;  
    }  
    return result;  
}
```

Esta función presenta una sentencia asm distinta de la versión "oficial" recomendada en clase. En concreto son distintas la etiqueta y las instrucciones adc/test.

Esta función parity4:

- a. produce siempre el resultado correcto
- b. fallaría con array={0,1,2,3}
- c. fallaría con array={1,2,4,8}

d. no siempre produce el resultado correcto, pero el error no se manifiesta en los ejemplos propuestos, o se manifiesta en ambos

13. En la misma práctica "parity" un estudiante entrega la siguiente versión de parity4:

```
int parity4(unsigned* array, int len){  
    int i;  
    unsigned x;  
    int val, result = 0;  
    for (i=0; i<len; i++){  
        x = array[i];  
        val = 0;  
        asm("\n"  
            "ini4: \n\t "  
            " xor %[x], %[y] \n\t"  
            " shr $1, %[x] \n\t "  
            " cmpl $0, %[x] \n\t "  
            " jnz ini4 \n\t "  
            :[y] "+r" (val)  
            :[x] "r" (x)  
            );  
        result += val & 0x1;  
    }  
    return result;  
}
```

Esta función presenta dos diferencias con la versión "oficial" recomendada en clase, relativas a la instrucción cmp y al nombre [y] escogido para la restricción val. Esta parity4:

- a. produce siempre el resultado correcto
- b. fallaría con array={0,1,2,3}
- c. fallaría con array={1,2,4,8}
- d. no siempre produce el resultado correcto, pero el error no se manifiesta en los ejemplos propuestos, o se manifiesta en ambos

14. En la misma práctica "parity" un estudiante entrega la siguiente versión de parity6:

```
int parity6(int *array, int len){  
    int i, res=0;  
    unsigned x;  
    for (i = 0; i < len; i++){  
        x = array[i];  
        asm(  
            "mov %[x], %%edx \n\t"  
            "shr $16, %[x] \n\t"  
            "xor %[x], %%edx \n\t"  
            "xor %%dh, %%dl\n\t"  
            "setpo %%dl \n\t"  
            "movzx %%dl, %[x]\n\t"  
            : [x] "+r" (x)  
            :  
            : "edx"  
            );  
        res += (x & 0x1);  
    }  
    return res;
```

```
}
```

```
int v[262144];
for (i = 0; i < 262144; i += 8)
    v[i] = 9;
```

Esta función presenta dos diferencias con la versión "oficial" recomendada en clase, relativas al tipo del array y a la máscara y paréntesis usados al acumular.

Esta función parity6:

- a. produce siempre el resultado correcto
  - b. fallaría con array={0,1,2,3}
  - c. fallaría con array={1,2,4,8}
  - d. no siempre produce el resultado correcto, pero el error no se manifiesta en los ejemplos propuestos, o se manifiesta en ambos
- 

15. En la práctica de la bomba, el primer ejercicio consistía en “saltarse” las “explosiones”, para lo cual se puede utilizar...

- a. objdump o gdb
  - b. gdb o ddd
  - c. ddd o hexedit
  - d. hexedit u objdump
- 

16. En la práctica de la bomba, el tercer ejercicio consistía en usar un editor hexadecimal para crear un ejecutable sin “explosiones”. Para saber qué contenidos del fichero hay que modificar, se puede utilizar... (marcar la opción **\*FALSA\***)

- a. objdump
  - b. gdb
  - c. ddd
  - d. hexedit
- 

17. Suponer una memoria cache con las siguientes propiedades: Tamaño: 512 bytes. Política de reemplazo: LRU. Estado inicial: vacía (todas las líneas inválidas). Suponer que para la siguiente secuencia de direcciones enviadas a la cache: 0, 2, 4, 8, 16, 32, la tasa de acierto es 0.33. ¿Cuál es el tamaño de bloque de la cache?

- a. 4 bytes
  - b. 8 bytes
  - c. 16 bytes
  - d. Ninguno de los anteriores
- 

18. Sea un computador de 32 bits con una memoria cache L1 para datos de 32 KB y líneas de 64 bytes asociativa por conjuntos de 2 vías. Dado el siguiente fragmento de código:

¿Cuál será la tasa de fallos aproximada que se obtiene en la ejecución del bucle anterior?

- a. 0 (ningún fallo)
  - b. 1/2 (mitad aciertos, mitad fallos)
  - c. 1/8 (un fallo por cada 8 accesos)
  - d. 1 (todo son fallos)
- 
19. En la práctica de la cache, el código de size.cc accede al vector saltando de 64 en 64. ¿Por qué?
- a. Porque cada elemento del vector ocupa 64 bytes
  - b. Para recorrer el vector más rápidamente
  - c. Porque el tamaño de cache L1 de todos los procesadores actuales es de 64KB
  - d. Para anular los aciertos por localidad espacial, esto es, que sólo pueda haber aciertos por localidad temporal
- 

20. ¿En qué unidades se suelen medir las capacidades de almacenamiento de los niveles de cache L1, L2 y L3 de un microprocesador actual (2017-2018)?

- a. L1 en KB, L2 en KB o MB, L3 en MB.
  - b. L1 en MB, L2 en GB, L3 en GB o TB.
  - c. L1 en MB, L2 en MB, L3 en GB.
  - d. L1 en KB, L2 en MB, L3 en GB.
-

Nombre:	
DNI:	Grupo:

### Examen de Problemas (3,0 p)

#### 1. Ensamblador (1 punto). Una posible implementación en C para la función

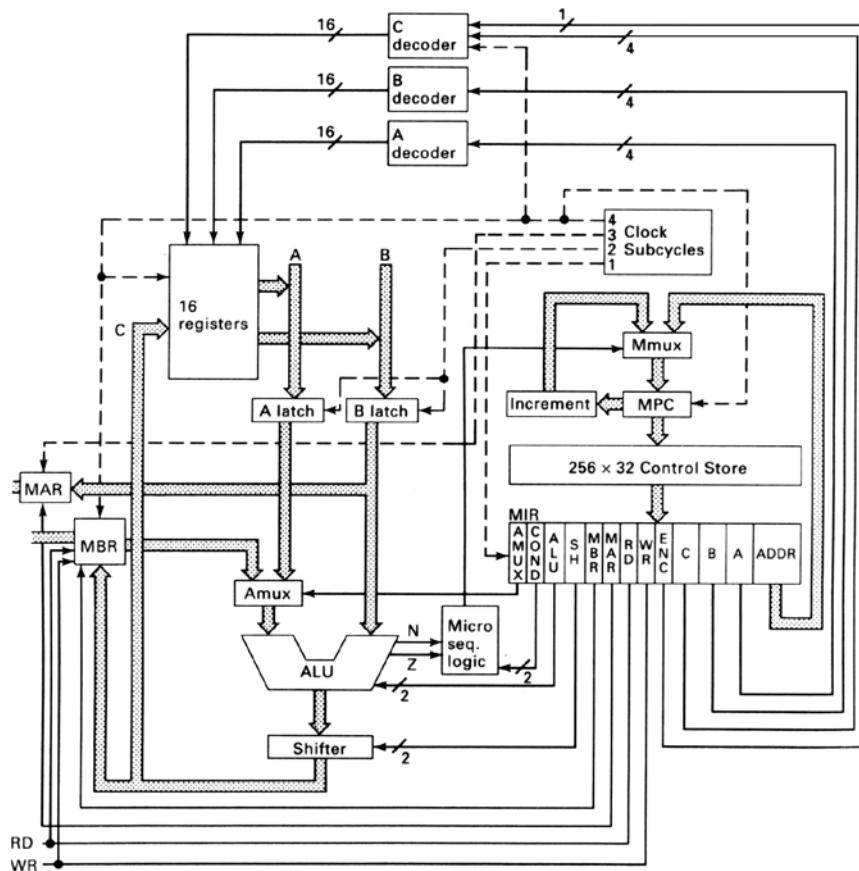
```
size_t mystrlen(const char *s);
```

que calcula la longitud de una cadena de caracteres apuntada por s, es esta:

```
#include <stddef.h>
size_t mystrlen(const char *s) {
    size_t len;
    for (len = 0; s[len] != 0; len++);
    return len;
}
```

Escriba dos soluciones en ensamblador, una para IA32 y otra para x86-64, para dicha función. El tipo size\_t es equivalente a un entero sin signo de 32 bits y 64 bits, respectivamente.

#### 2. Unidad de control (0.6 puntos). La microarquitectura de la siguiente figura cuenta al menos con estos registros de 32 bits:



- SP, que indica la dirección de memoria del tope de la pila.
- IR, que contiene la instrucción que se está ejecutando.
- +1, cuyo valor es +1
- -1, cuyo valor es -1

El registro MAR tiene 28 bits y está conectado a los 28 bits menos significativos del bus B.

Escriba el microcódigo (microinstrucciones) para implementar las siguientes instrucciones, usando pseudocódigo (sin detallar las señales de control):

- **pop dir**, que saca datos del tope de la pila y los almacena en la dirección de memoria dir (dir es un operando de 28 bits).
- **push dir**, que coloca los datos almacenados en la dirección de memoria dir (dir es un operando de 28 bits) en el tope de la pila.

**3. Entrada/salida** (0.4 puntos). Necesitamos conocer el impacto de la sobrecarga que supone el *polling* en una interfaz de ratón que debe ser sondeada 30 veces por segundo para asegurar que no se pierde ningún movimiento realizado por el usuario. Suponga que el número de ciclos que requiere cada operación de *polling*, incluidos el salto a la rutina de encuesta, el acceso al dispositivo y el retorno al programa de usuario, es 400 ciclos, y que el procesador trabaja con un reloj de 2 GHz. Determine la proporción (porcentaje) del tiempo de CPU que consume el *polling*.

**4. Configuración de memoria** (0.5 puntos). Diseñe y dibuje el esquema de interconexión de una memoria SRAM de 4K palabras x 8 bits de longitud de palabra, a partir de módulos de memoria de 2K x 1 bits y módulos combinacionales.

**5. Memoria cache** (0.5 puntos). El SiFive U54-MC Core Complex es una implementación multi núcleo de la arquitectura abierta RISC-V. Cada uno de los núcleos U54 tiene un espacio de direccionamiento virtual de 512 GiB, un espacio de direcciones físicas de 38 bits, una cache L1 de instrucciones de 32 KB y otra L1 de datos de 32 KB. Cada una de las caches es asociativa por conjuntos con 8 vías, con líneas de 64 B.

- (0.1) Indique la primera y la última direcciones virtuales en hexadecimal
- (0.1) Indique la primera y la última direcciones físicas en hexadecimal
- (0.3) Indique los nombres y tamaños de los campos en los que se divide una dirección de memoria física de memoria desde el punto de vista de una cache L1

### Test de Teoría (3.0p)

1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	
b	b	d	c	d	b	c	c	b	b	a	a	a	a	d	b	d	d	b	c	d	a	c	a	d	b	a	c	a	b	c

### Test de Prácticas (4.0p)

1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20
c	b	d	c	c	a	c	c	d	d	d	a	a	a	b	d	b	b	d	a

### Examen de Problemas (3.0p)

#### 1. Ensamblador (1 punto).

11+7=18 instrucciones, 3+3=6 etiquetas, 2+2=4 directivas, alrededor de **0.05p/instrucción, 0.02p/etiqueta**, total **18 x 0.05p + 6 x 0.02p = 0.9p + 0.12p ≥ 1p** total

Alternativamente, **0.5p** cada programa. Las puntuaciones son aproximadas, de todas formas.

#### Solución:

32 bits (0.5 puntos, 11 instrucciones máquina)	64 bits (0.5 puntos, 7 instrucciones máquina)
<pre>.text .globl mystrlen  mystrlen:     pushl %ebp     movl %esp, %ebp      xorl %eax, %eax ; len = 0      movl 8(%ebp), %edx ; edx = &amp;s[0]     cmpb \$0, (%edx) ; s[0] == '\0'?     je .L2 ; == '\0' =&gt; end  .L3:     addl \$1, %eax ; len++     cmpb \$0, (%edx,%eax) ; s[len] == '\0'?     jne .L3 ; != '\0' =&gt; loop  .L2:     popl %ebp     ret</pre>	<pre>.text .globl mystrlen  mystrlen:      xorl %eax, %eax ; len = 0      cmpb \$0, (%rdi) ; s[0] == '\0'?     je .L4 ; == '\0' ==&gt; end  .L3:     addq \$1, %rax ; len++     cmpb \$0, (%rdi,%rax) ; s[len] == '\0'?     jne .L3 ; != '\0' ==&gt; loop  .L4:     ret</pre>

#### 2. Unidad de Control (0.6 puntos).

8 micro-instrucciones, 12 micro-pseudo-ops (incluyendo goto fetch).

Se puntuá **0.05p** por micro-pseudo-op (total **0.05 x 12 = 0.60p**)

```
pop:   MAR := SP[27:0]
       SP := SP + (+1); MBR := M[MAR]      // Leer M[SP++]
       MAR := IR[27:0]
       M[MAR] := MBR; goto fetch           // Escribir M[dir]

push:  MAR := IR[27:0]
       SP := SP + (-1); MBR := M[MAR];    // Leer M[dir]
       MAR := SP[27:0]
       M[MAR] := MBR; goto fetch          // Escribir M[--SP]
```

### 3. Entrada/salida (0.4 puntos).

Se puntuá **0.1p** por operación indicada, **0.1p** por valor correcto, total  **$2 \times 0.1 + 2 \times 0.1 = 0.4p$**

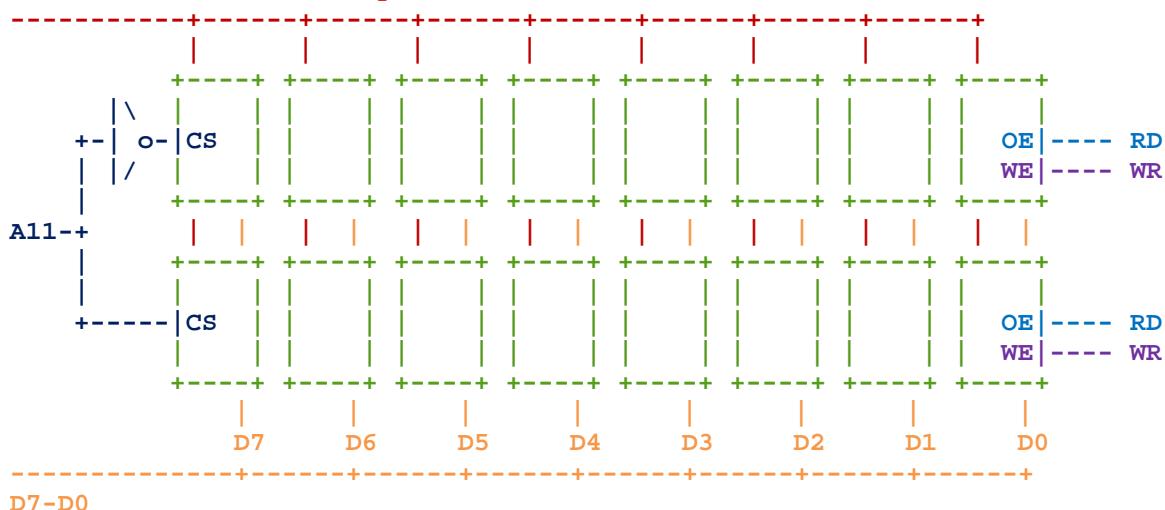
$$30 \text{ pollings/s} * 400 \text{ ciclos/polling} = 12000 \text{ ciclos/s}$$

$$\text{Porcentaje} = 100 * 12000 \text{ ciclos/s} / 2000000000 \text{ ciclos/s} = 12 / 20000 = 0,0006\%$$

### 4. Configuración de memoria (0.5 puntos).

Aproximadamente **0.1p** por zona (**CS/A<sub>11</sub>**, **WE/WR**, **OE/RD**, **A<sub>10-0</sub>**, **D<sub>7-0</sub>**)

**A10-A0 (a todos los chips)**



### 5. Memoria cache (0.5 puntos).

#### Solución:

a) Indique la primera y la última direcciones virtuales en hexadecimal: **0.1p**

0x00 0000 0000

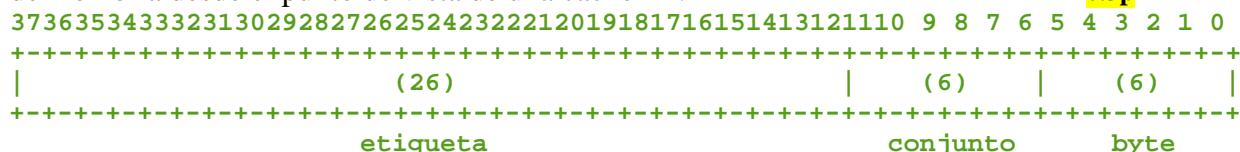
0x7F FFFF FFFF

b) (0,1) Indique la primera y la última direcciones físicas en hexadecimal: **0.1p**

0x00 0000 0000

0x3F FFFF FFFF

c) (0,3) Indique los nombres y tamaños de los campos en los que se divide una dirección de memoria física de memoria desde el punto de vista de una cache L1. **0.3p**



#### Explicación:

$$2^{15} \text{ B}$$

$$----- = 2^9 \text{ líneas}$$

$$2^6 \text{ B/línea}$$

$$2^9 \text{ líneas}$$

$$----- = 2^6 \text{ conjuntos}$$

$$2^3 \text{ líneas/conjunto}$$

$$38 - 12 = 26 \text{ bits cada etiqueta}$$

#### Referencias:

<https://static.dev.sifive.com/U54-MC-RVCCoreIP.pdf>

<https://riscv.org/specifications/privileged-isa/>

**Nombre:**
**DNI:**
**Grupo:**

### Test de Teoría (3.0p)

Todas las preguntas son de elección simple sobre 4 alternativas.

Cada respuesta vale 0.1p si es correcta, 0p si está en blanco o claramente tachada, -0.03p si es errónea.

Anotar las respuestas (a, b, c ó d) en la siguiente tabla.

1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30

1. ¿Cuál es el complemento a 2 del número binario 1110 1101 1000?

- a. 0001 0010 0110
- b. 0001 0010 0101
- c. 0001 0010 0111
- d. 0001 0010 1000

2. Escoger de entre las 4 operaciones la de mayor valor que pueda calcularse con enteros de 4B con signo sin problemas

- a. 100.000.000 + 100.000.000
- b. 300.000.000 + 300.000.000
- c. 1.000.000.000 + 1.000.000.000
- d. 3.000.000.000 + 3.000.000.000

3. Si **rcx** vale -1, tras ejecutar las instrucciones

```
rol $1, %cl
rcr $2, %rcx
```

el nuevo valor de RCX y del flag CF es

- a. hay algún fallo de sintaxis o gramática en esas instrucciones
- b. **RCX** ≠ -1, CF mantiene su valor
- c. **RCX** = -1, CF = 1
- d. no se puede marcar ninguna de las opciones anteriores

4. El registro RAX contiene el número binario 111111101111000010. ¿Cuál será su contenido tras ejecutar la instrucción **sar \$1,%ax**?

- a. 0x3fde1
- b. 0x7fde1
- c. 0xffde1

d. 0xfffffffffffffde1

5. Para comprobar si el contenido del registro RDX es 0 (y posiblemente saltar a continuación usando la instrucción je), el compilador gcc genera:

- a. cmpq %rdx, %rdx
- b. testq %rdx
- c. testq %rdx, %rdx
- d. cmpq %rdx

6. Sabiendo que las instrucciones de salto condicional codifican la dirección de salto con direccionamiento relativo a contador de programa (de 8 o 32 bits con signo), indicar cuál es la dirección de salto de la instrucción je en el siguiente desensamblado, donde se ha tachado precisamente dicha dirección.

```
40042f: 74 f4      je xxxxxxxx
400431: 5d          pop %rbp
```

- a. 400431
- b. 400525
- c. 400425
- d. 40043d

7. Para traducir una asignación condicional (**a = b ? c : d ;**) de lenguaje C a lenguaje ensamblador, gcc puede que utilice...

- a. Un salto incondicional, según la condición expresada en el código C, y otro salto incondicional
- b. Un salto condicional, según la condición opuesta a la del código C, y otro salto condicional

c. Una instrucción de movimiento condicional, pero sólo si el procesador es Pentium Pro/II o superior

d. Una instrucción de movimiento incondicional, pero sólo si el S.O. es de 64bits

---

#### 8. La instrucción **cmovb %rdx,%rax**

- a. copia el byte bajo de rdx en el byte bajo de rax
  - b. copia en rax el byte de memoria apuntado por la dirección contenida en rdx
  - c. copia en rax el contenido de rdx si rax es menor que rdx
  - d. copia en rax el contenido de rdx si CF= 1**
- 

9. Uno de los puntos clave de la traducción que gcc hace de una construcción switch-case de lenguaje C a lenguaje ensamblador es...

- a. el salto condicional hacia atrás
  - b. el salto relativo a contador de programa
  - c. el salto directo
  - d. el salto indirecto**
- 

10. El procesador utiliza el puntero de pila...

- a. En las instrucciones de llamadas y retornos de subrutinas**
  - b. En todo tipo de instrucciones de saltos, incluyendo llamadas y retornos a subrutinas
  - c. En todas las instrucciones que tengan al menos dos accesos a memoria
  - d. En todas las instrucciones
- 

11. ¿Cuál de las siguientes instrucciones situada al principio de una función se utilizará probablemente para crear espacio en la pila para variables locales sin inicializar?

- a. sub \$0x30, %rsp**
  - b. add \$0x30, %rsp
  - c. sub \$0x30, %rbp
  - d. add \$0x30, %rbp
- 

12. En la convención de llamada SystemV AMD64 seguida por gcc Linux/x86-64...

- a. RAX es un registro salva-invocante, por eso en cualquier función hay que salvarlo antes de modificarlo
  - b. R10 es un registro salva-invocante, por eso si es necesario hay que salvarlo antes de llamar a función**
- 

c. R11 es un registro salva-invocado, por eso en cualquier función hay que salvarlo antes de modificarlo

d. RBP es un registro salva-invocado, por eso si es necesario hay que salvarlo antes de llamar a función

---

13. Un procedimiento llamado por una instrucción call debe guardar y restaurar los registros siguientes siempre que los altere:

- a. %rsi, %rdi
  - b. %rax, %rbx, %rcx, %rdx**
  - c. %rax, %rdx, %rcx
  - d. %rbx, %rbp
- 

14. Dada una función que devuelve la suma de 8 enteros en x86-64, ¿cuál de las siguientes instrucciones suma el 7º argumento?

- a. add -0x8(%rsp), %eax
  - b. add 0x8(%rsp), %eax**
  - c. add -0x4(%rsp), %eax
  - d. add 0x4(%rsp), %eax
- 

15. En el fragmento de programa siguiente:

```
66b: e8 8a ff ff ff  callq 5fa <f>
670: 48 83 c4 10      add $0x10,%rsp
```

¿Cuál es el valor que introduce en la pila la instrucción callq?

- a. 0x670**
  - b. 0xfffffff8a
  - c. 0x66b
  - d. 0x5fa
- 

16. En el fragmento de programa siguiente:

```
66b: e8 8a ff ff ff  callq 5fa <f>
670: 48 83 c4 10      add $0x10,%rsp
```

la instrucción callq suma al contador de programa la cantidad:

- a. -0x76**
  - b. 0x5fa
  - c. 0xfffffff8a
  - d. 0x76
- 

17. Suponga la siguiente llamada a una función f de 4 argumentos:

```
mov    $0x1, %ecx
mov    $0x2, %edx
mov    $0x3, %esi
mov    $0x4, %edi
callq 5fa <f>
```

El primer parámetro de llamada a la función:

- a. Es el valor inmediato 1
  - b. Es el contenido de la dirección de memoria 0x1
  - c. Es el valor inmediato 4
  - d. Es el contenido de la dirección de memoria 0x4
- 

18. Habiendo declarado **int array={0,1,2,3};** y **long long \*ptr=array;** ¿cuánto vale **ptr[1]**?
- a. 0x0000 0001 0002 0003
  - b. 0x0000 0001 0000 0000
  - c. 0x0003 0002 0001 0000
  - d. 0x0000 0003 0000 0002
- 

19. Una función C llamada **get\_el(...)** genera el siguiente código ensamblador.

```
leaq    (%rdi,%rdi,4), %rax
addq    %rax, %rsi
movl    arr(%rsi,4), %eax
ret
```

Se puede adivinar que:

- a. arr es un array multi-nivel (punteros a enteros) de cuatro filas
  - b. arr es un array multi-nivel pero no se pueden adivinar las dimensiones
  - c. arr es un array bidimensional de enteros, no se pueden adivinar dimensiones
  - d. arr es un array bidimensional de enteros, con cinco columnas
- 

20. Las microoperaciones de la fase de captación de una instrucción:

- a. Son comunes para todas las instrucciones
  - b. Dependen del código de operación de la instrucción que se encuentra en el registro de instrucción
  - c. Dependen de los indicadores de estado y del código de operación de la instrucción que se encuentra en el registro de instrucción
  - d. Dependen del valor del contador de programa
- 

21. Para el procesador con unidad de control microprogramada estudiado en clase, Tanenbaum propone codificar los 16 registros y añadir una señal “PERC” para habilitar la carga desde el bus C (recordar que era un diseño típico con 3 buses) y así no perder expresividad/paralelismo. El ahorro de bits en cada microinstrucción debido a esta técnica es de

- a. 40 bits
  - b. 39 bits
  - c. 35 bits
  - d. 29 bits
- 

22. Un procesador está segmentado en k etapas. Cada una de ellas consume un tiempo t. La aceleración ideal (si no hay riesgos) al ejecutar 5 instrucciones respecto a un procesador no segmentado será:
- a. 5k / (4+k)
  - b. (4+k) / 5t
  - c. 4k / (5+k)
  - d. (5+k) / 4t
- 

23. ¿Cuál de las siguientes afirmaciones sobre la E/S programada con consulta de estado es cierta?
- a. Si se emplea E/S programada puede hacerse con consulta de estado o sin consulta de estado
  - b. Un programa que realice salida programada con consulta de estado no ejecutará ninguna instrucción de entrada o carga
  - c. Sólo la E/S por DMA libera a la CPU de realizar la consulta de estado del dispositivo de E/S
  - d. La escritura de un led requiere consulta de estado
- 

24. En un sistema de interrupciones vectorizado y en daisy-chain, ¿cuál de las siguientes afirmaciones es cierta?
- a. El procesador informa de un ciclo de reconocimiento de interrupción con la señal de reconocimiento de interrupción (INTA) y la identificación de los dispositivos se realiza por consulta de estado
  - b. La gestión de prioridades queda establecida por el orden en que los dispositivos reciben la señal INTA y el dispositivo se identifica por un dato que deposita en el bus
  - c. La gestión de prioridades queda establecida por el orden en que los dispositivos reciben la señal INTA y la identificación de los dispositivos se realiza leyendo sus registros de estado
  - d. El daisy-chain asigna a todos los dispositivos la misma prioridad y la identificación de los dispositivos se realiza leyendo sus registros de estado
-

**25.** ¿Cuál de las siguientes características es menos probable que pueda programarse en un canal DMA?

- a. dos direcciones (origen y destino)
  - b. dos tamaños (copia origen y copia destino)**
  - c. cuál de las dos direcciones es de E/S (si alguna lo es) en lugar de Memoria
  - d. si se desea producir una IRQ al terminar
- 

**26.** En un computador con una jerarquía de memoria de dos niveles se observa experimentalmente que el tiempo medio de acceso a la memoria es de 300 ns cuando en realidad el tiempo medio de acceso al primer nivel es de 6 ns. Sabiendo que el tiempo de acceso al segundo nivel es de 3 microsegundos, ¿cuál sería aproximadamente el porcentaje de fallos en los accesos al primer nivel?

- a. 90%
  - b. 1%
  - c. 10%**
  - d. 99%
- 

**27.** El orden de magnitud del tiempo de acceso a la memoria DRAM de un computador es de:

- a. Picosegundos
  - b. Nanosegundos**
  - c. Microsegundos
  - d. Milisegundos
- 

**28.** Una memoria estática tiene un bus de datos de 32 bits y su bus de direcciones es de 20 bits, ¿cuál es su capacidad?

- a. 4 MBytes**
  - b. 1 MByte
  - c. 32 MBytes
  - d. 80 GBytes
- 

**29.** En una cache asociativa por conjuntos de  $2^v$  vías con  $2^b$  líneas (marcos de bloque) de  $2^w$  palabras, el gestor de memoria **no** considera como campo (conjunto de bits contiguos con significado o relevancia) los siguientes bits:

- a. últimos w bits (0...w-1) (los menos significativos)
  - b. bits w...w+c-1 (con c=b-v)
  - c. bits w...w+c-1 (siendo  $2^c=n$ º conjuntos)
  - d. bits b...b+c-1 (siendo  $2^c=n$ º conjuntos)**
- 

**30.** Para obtener una única velocidad comparativa final, el benchmark SPEC CPU combina las ganancias en velocidad de ejecución de una serie de tests, respecto a un ordenador de referencia, usando...

- a. la mediana
  - b. la media aritmética
  - c. la media geométrica**
  - d. la moda
-

**Nombre:**
**DNI:**
**Grupo:**

## Test de Prácticas (4.0p)

**Todas las preguntas son de elección simple sobre 4 alternativas.**

**Cada respuesta vale 0.2p si es correcta, 0 si está en blanco o claramente tachada, -0.06p si es errónea.**

**Anotar las respuestas (a, b, c ó d) en la siguiente tabla.**

1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20
															c		d		

1. Habiendo definido en código fuente ASM **longsal: .quad .-saludo** justo detrás de un string **saludo** que ocupaba 28 bytes, si se comparan los comandos gdb siguientes: **x/1xg &longsal** frente a **print (long) &longsal:**

- a. ambos nos muestran la longitud del string (que es/vale/equivale a 28)
- b. el primero (x) nos muestra la longitud, y el segundo (print) nos muestra otro valor distinto
- c. el segundo (print) nos muestra la longitud, y el primero (x) nos muestra otro valor distinto
- d. alguno o ambos contienen algún error gramatical (falta o sobra algún &, algún typecast (char\*) o (long), etc.)

2. En la práctica "media" se pide sumar una lista de 16 enteros SIN signo de 32 bits evitando acarreo. ¿Cuál es el menor valor que repetido en toda la lista causaría acarreo en 32 bits?

- a. 0xFFFF FFFF
- b. 0x7FFF FFFF
- c. 0x1000 0000
- d. 0x0FFF FFFF

3. En la práctica "media" se pide usar **adc** para sumar una lista de 16 enteros SIN signo de 32 bits en dos registros de 32 bits mediante extensión con ceros. Un estudiante entrega la siguiente versión:

```

...
resultado: .quad 0
...
main: .global main
    mov $lista, %rbx
    mov $16, %ecx
    call suma

```

```

mov %eax, resultado
# código para printf ...
# código para _exit ...
suma:
    mov $0, %eax
    mov $0, %rdx
bucle:
    adc (%rbx, %rdx, 4), %eax
    inc %rdx
    cmp %rdx, %rcx
    jnle bucle
    ret

```

Este programa no usa la variable **longlista**, guarda el resultado con una instrucción MOV, usa como índice RDX, no usa la instrucción ADD, y usa JNLE para el salto condicional.

Al empezar un programa CF no está activado. Esta versión de la suma SIN signo mediante extensión con ceros da resultado correcto:

- a. con lista: **.int 0x10000000, ... (16 elementos)**
- b. con lista: **.int 200000000, ... (16 elementos)**
- c. con ambos ejemplos
- d. con ninguno de los dos ejemplos

4. En la práctica "media" se pide usar **cltd/cdq** para sumar una lista de 16 enteros CON signo de 32 bits en dos registros de 32 bits mediante extensión de signo. Un estudiante entrega la siguiente versión:

```

...
main: .global main
    mov $lista, %rbx
    mov longlista, %rcx
    call suma
    mov %eax, resultado
    mov %edx, resultado+4

```

```

movq    $formato, %rdi
movq    resultado,%rsi
movq    resultado,%rdx
movl    $0,%eax
call   printf
...

```

El programa produce la siguiente salida con el test #01 (16 elementos con valor -1):

```

__TEST01-----
resultado =          -16 (sng)
                = 0x ffffffff0 (hex)
                = 0x 00000010 9f816d80

```

Recordar que todo el texto aparecía tal cual literalmente en el formato (ignorar la errata sng) y los números llevaban especificación de formato (%18ld, %18lx, etc). De esta versión de la suma CON signo mediante extensión de signo se puede afirmar que:

- al inicio de main EAX vale 0 y R8 contiene un valor inferior a 0x7fffffff00000000
  - al llamar a suma RBX contiene un valor inferior a 0x600000 y RCX vale 16
  - al llamar a printf, ECX vale 16 y R8 contiene un valor superior a 0x80000000
  - tras volver de printf RAX contiene un valor superior a 60 y RDI superior a 0x600000
5. En la práctica "media" se pide usar `cltq/cdq` y `cqto/cqo` para hallar la media y resto de una lista de 16 enteros CON signo de 32 bits usando registros de 64 bits. Un estudiante entrega la siguiente versión:

```

...
media: .double 0
resto: .double 0
formatoq:
.ascii "media = %lld resto = %lld\n"
.asciz "\t = 0x %08x \t = 0x %08x\n"
...
mov     $lista, %rbx
mov     longlista, %ecx
call   sumaq

mov     $formatoq, %rdi
mov     media,%rsi
mov     resto,%rdx
mov     $0,%eax
call   printf
...
sumaq:
push   %rdx
push   %rsi
mov    $0, %rax
mov    $0, %rsi
mov    $0, %rdx
mov    $0, %r8
bucleq:
    mov    (%rbx,%rsi,4), %eax
    cdqe                         # EAX -> RAX

```

```

add   %rax, %r8
inc   %rsi
cmp   %rsi,%rcx
jne   bucleq
mov   %r8, %rax
cqo
        # RAX -> RDX:RAX
idivq %rsi
# mov  %rdx, %r10
mov   %rdx, resto
mov   %rax, media
pop   %rdx
pop   %rsi
ret

```

Este programa es muy diferente a la versión "oficial" recomendada en clase. Notar los `push/pop`, los `mov $0` adicionales, `idiv %rsi` en lugar de `%rcx`, los `mov media/resto` al final de la subrutina en lugar de tras la llamada en `main`, y el tipo de ambas variables.

¿Qué media y resto imprime esta versión para el test #03? (16 elementos con valor 0x7fffffff)

- `media = 2147483647 resto = 0`  
`= 0x 7fffffff = 0x 00000000`
- `media = 16 resto = -16`  
`= 0x 7fffffff = 0x 00000000`
- `media = 2147483647 resto = 0`  
`= 0x 00000010 = 0x ffffffff0`
- `media = 16 resto = -16`  
`= 0x 00000010 = 0x ffffffff0`

6. ¿Cuál expresión es cierta?

- `popcount(15) < popcount(51)`
- `popcount( 2 ) == popcount(64)`
- `popcount( 7 ) > popcount(60)`
- `popcount(96) != popcount( 3 )`

7. La práctica "popcount" debía calcular la suma de bits (peso Hamming) de los elementos de un array. Un estudiante entrega la siguiente versión de `popcount1`:

```

int pc1(unsigned* array, size_t len){
    size_t i,j;
    int res=0;
    unsigned x;
    for (i=0; i<len; i++){
        x = array[i];
        for (j=0; j<8*sizeof(int);j++){
            x >>= 1;
            unsigned bit = x & 0x1;
            res+=bit;
        }
    }
    return res;
}

```

Esta función se diferencia de la versión "oficial" recomendada en clase en el cuerpo del bucle interno. Esta función `popcount1`:

- a. produce siempre el resultado correcto
- b. fallaría con array={0,1,2,3}
- c. fallaría con array={1,2,4,8}
- d. no es correcta pero el error no se manifiesta en los ejemplos propuestos, o se manifiesta en ambos

8. En la misma práctica "popcount" un estudiante entrega la siguiente versión de `popcount2`:

```
int pc2(int* array, size_t len){  
    size_t i;  
    int res=0;  
    unsigned x;  
    unsigned bit;  
    for (i=0; i<len; i++){  
        x = array[i];  
        while(x){  
            bit += x & 0x1;  
            x >>= 1;  
            res = res + bit;  
        } }  
    return res;  
}
```

Esta función se diferencia de la versión "oficial" recomendada en clase en el tipo del **array**, la variable **bit** y el cuerpo del bucle interno. Esta función `popcount1`:

- a. produce siempre el resultado correcto
- b. fallaría con array={0,1,2,3}
- c. fallaría con array={1,2,4,8}
- d. no es correcta pero el error no se manifiesta en los ejemplos propuestos, o se manifiesta en ambos

9. En la misma práctica "popcount" un estudiante entrega la siguiente versión de `popcount4`:

```
int pc4(unsigned* array, size_t len){  
    size_t i;  
    int res=0;  
    unsigned x;  
    for (i=0; i<len; i++){  
        x = array[i];  
        asm("\n\t"  
            "clc\n"  
            "ini4:\n"                "\n\t"  
            "adc $0, %[r]\n\t"  
            "test %[x],%[x]\n\t"  
            "shr %[x]\n\t"  
            "jne ini4\n\t"  
            "adc $0, %[r]\n\t"  
            : [r]"+r" (res)  
            : [x] "r" (x) );  
    }  
    return res;  
}
```

Esta función se diferencia de la versión "oficial" en que tiene una instrucción ensamblador adicional. Este `popcount4`:

- a. produce siempre el resultado correcto
- b. fallaría con array={0,1,2,3}

- c. fallaría con array={1,2,4,8}
- d. no es correcta pero el error no se manifiesta en los ejemplos propuestos, o se manifiesta en ambos

10. En la misma práctica "popcount" un estudiante entrega la siguiente versión de `popcount4`:

```
int pc4(unsigned* array, size_t len){  
    size_t i;  
    int res=0;  
    unsigned x;  
    for (i=0; i<len; i++){  
        x = array[i];  
        asm("\n\t"  
            "clc\n"  
            "ini4:\n"                "\n\t"  
            "adc $0, %[r]\n\t"  
            "fin4:\n"                "\n\t"  
            "shr %[x]\n\t"  
            "jne ini3\n\t"  
            : [r]"+r" (res)  
            : [x] "r" (x) );  
    }  
    return res;  
}
```

Esta función es muy diferente a la versión "oficial". Notar el salto condicional a "ini3" en la función `popcount3` (en donde sí se hizo bien el `ini3:/shr/adc/test/jne` recomendado). Esta función `popcount4`:

- a. produce siempre el resultado correcto
- b. fallaría con array={0,1,2,3}
- c. fallaría con array={1,2,4,8}
- d. no es correcta pero el error no se manifiesta en los ejemplos propuestos, o se manifiesta en ambos

11. En la práctica de la bomba, el primer ejercicio consistía en saltarse las explosiones, para lo cual se puede utilizar... (marcar opción **falsa**)

- a. objdump
- b. gdb
- c. ddd
- d. eclipse

12. ¿Para qué se utiliza la función `gettimeofday()` en la práctica de la "bomba digital"?

- a. Para cronometrar y poder comparar lo que tardan las distintas versiones del programa
- b. Para imprimir la hora en la pantalla
- c. Para cifrar la clave en función de la hora actual
- d. Para lanzar un error cuando el usuario tarde demasiado tiempo en introducir la clave

13. ¿Para qué se utiliza la función `scanf()` en la práctica de la "bomba digital"?

- a. Para escanear el fichero ejecutable "bomba" y asegurarse de que no contenga virus

- b. Para leer la contraseña (clave alfanumérica)
- c. Para leer el PIN (clave numérica)
- d. Para lanzar un error cuando el usuario tarde demasiado tiempo en introducir la clave

14. Respecto a las bombas estudiadas en la práctica "bomba digital", ¿en cuál de los siguientes tipos de bomba sería más **difícil** descubrir la contraseña? Se distingue entre strings definidos en el código fuente de la bomba, y strings solicitados al usuario por teclado. Por "cifrar" podemos entender la cifra del César, por ejemplo.

- a. 1 string del fuente se cifra, se invierte y se compara con el string del usuario
- b. el string del usuario se cifra y se compara con 1 string del fuente
- c. 2 strings del fuente se invierten, se concatenan, se cifra el resultado, y se compara con el string del usuario
- d. el string del usuario se concatena con 1 string del fuente, luego se invierte 1 string del fuente, se cifra y se compara con el concatenado

15. En una bomba como las estudiadas en prácticas, del tipo...

```
0x40080f <main+180> lea 0xc(%rsp),%rsi
0x400814 <main+185> lea 0x1dd(%rip),%rdi
                                # 0x4009f8
0x40081b <main+192> mov    $0x0,%eax
0x400820 <main+197> call   0x400620<scanf>
0x400825 <main+202> mov    %eax,%ebx
0x400827 <main+204> test   %eax,%eax
0x400829 <main+206> jne   0x40083c <m+225>
0x40082b <main+208> lea   0x1c9(%rip),%rdi
                                # 0x4009fb
0x400832 <main+215> mov    $0x0,%eax
0x400837 <main+220> call   0x400620<scanf>
0x40083c <main+225> cmp    $0x1,%ebx
0x40083f <main+228> jne   0x4007f9 <m+158>
0x400841 <main+230> mov   0x200819(%rip),
                           %eax # 0x601060
0x400847 <main+236> cmp    %eax,0xc(%rsp)
0x40084b <main+240> je    0x400852 <m+247>
0x40084d <main+242> call   0x400727 <boom>
0x400852 <main+247> lea   0x10(%rsp),%rdi
```

...el código numérico (pin) es...

- a. el entero 0x601060
- b. el entero cuya dirección está almacenada en la posición de memoria 0x4009f8
- c. el entero almacenado a partir de la posición de memoria 0x4009fb
- d. el entero almacenado a partir de la posición de memoria 0x200819+0x400847

16. La función **setup()** de Arduino es llamada:

- a. Al principio de cada iteración de la función **loop()**

- b. Cuando se sube desde el **kit entorno** de desarrollo o se pulsa el botón de reset, pero no cuando se conecta la alimentación
- c. Cuando se conecta la alimentación a la placa, se pulsa el botón de reset, o se sube desde el **kit entorno** de desarrollo
- d. Cuando se conecta la alimentación a la placa, se pulsa el botón de reset, pero no cuando se sube desde el **kit entorno** de desarrollo

17. ¿Qué sentencia usamos en el programa **blink** (led intermitente) para encender el led integrado en la placa Elegoo Mega2560?

- a. **digitalWrite (LED\_BUILTIN, HIGH);**
- b. **pinMode (LED\_BUILTIN, OUTPUT);**
- c. **analogWrite (LED\_BUILTIN, HIGH);**
- d. **pulseIn (LED\_BUILTIN, HIGH);**

18. Sobre el resultado devuelto por la función de Arduino **map(sensorValue, sensorLow, sensorHigh, 50, 4000)**; usada en el programa del Theremín de luz:

- a. Si **sensorValue** vale 50, devuelve **sensorLow**
- b. Si **sensorValue** vale 50, devuelve **sensorHigh**
- c. Si **sensorValue** vale **25202025**, devuelve la mitad entre **sensorLow** y **sensorHigh**
- d. Si **sensorValue** vale la mitad entre **sensorLow** y **sensorHigh**, devuelve **25202025**

19. En el programa **line.cc** de la práctica de cache, si para cada tamaño de línea (**line**) recorremos una única vez el vector, la gráfica resultante es decreciente porque:

- a. hay un mayor historial de accesos y es más probable que un nuevo acceso sea un acierto
- b. cada vez los tamaños de línea escogidos van decreciendo y se tarda menos en leerlos
- c. cada vez los tamaños de línea escogidos van decreciendo y hay menor localidad espacial
- d. el vector se indexa con la variable de control del bucle, con un incremento o paso de **line**

20. En la práctica de la cache, en **size.cc** se accede al vector saltando de 64 en 64. ¿Por qué?

- a. Para recorrer el vector más rápidamente
- b. Porque con un salto menor que 64 habría aciertos por localidad espacial y haría menos clara la gráfica
- c. Porque cada elemento del vector ocupa 64 B
- d. Para evitar aciertos por localidad temporal y que sólo haya aciertos por localidad espacial

Nombre:	
---------	--

| DNI: |  |
| Grupo: |  |

### Examen de Problemas (3,0 p)

- 1. Estructuras.** (0.6 puntos). La estructura para almacenar los datos de una asignatura en SWAD es similar a **struct Course** en el siguiente programa. El tamaño del vector **Courses[10]** es 8400 cuando se compila con gcc en GNU/Linux 64 bits.

```
#include <stdio.h>

#define MAX_BYTES_NAME 255
struct Course
{
    long CrsCod;
    char InstitutionalCrsCod[
        MAX_BYTES_NAME];
    long DegCod;
    unsigned Year;
    unsigned Status;
    long RequesterUsrCod;
    unsigned NumUsrs[10];
    char ShrtName[MAX_BYTES_NAME];
    char FullName[MAX_BYTES_NAME];
} Courses[10];
```

```
int main (void)
{
    // Imprime 8400
    printf ("%lu\n", sizeof(Courses));
    return 0;
}
```

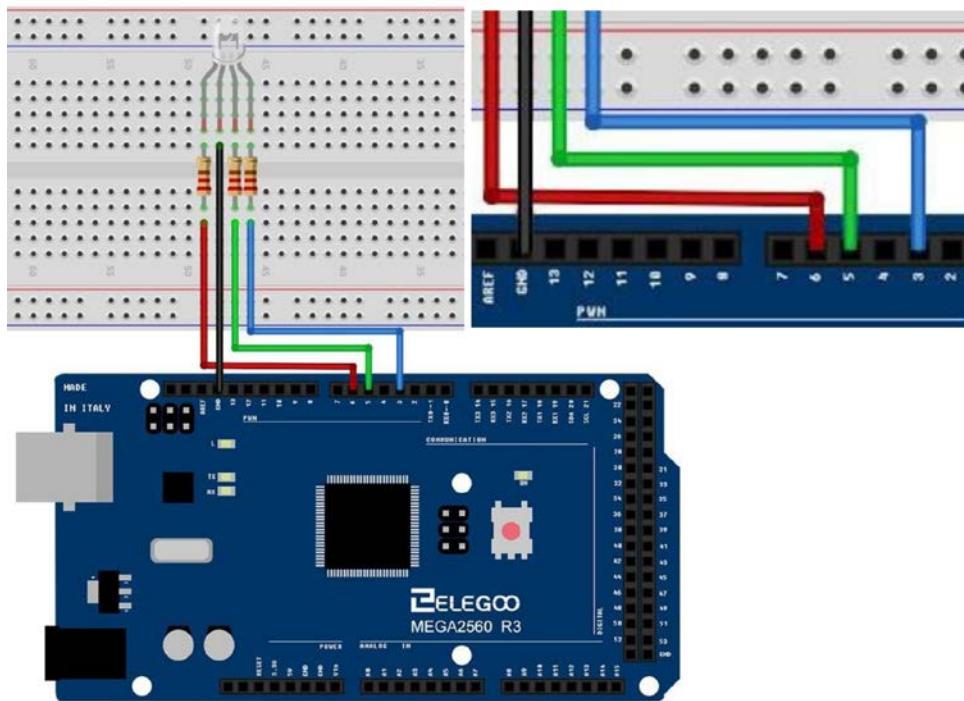
- a) Indique el desplazamiento relativo de inicio de cada uno de los campos en la siguiente tabla:

Tipo	Campo	Desplazamiento
long	Courses[0].CrsCod;	0
char	Courses[0].InstitutionalCrsCod[MAX_BYTES_NAME_0];	
long	Courses[0].DegCod;	
unsigned	Courses[0].Year;	
unsigned	Courses[0].Status;	
long	Courses[0].RequesterUsrCod;	
unsigned	Courses[0].NumUsrs[10_0];	
char	Courses[0].ShrtName[MAX_BYTES_NAME_0];	
char	Courses[0].FullName[MAX_BYTES_NAME_0];	
long	Courses[1].CrsCod;	

- b) ¿Cuál es el valor máximo de la constante **MAX\_BYTES\_NAME** para el cual **sizeof(Courses)** seguiría valiendo 8400?

**2. Unidad de control** (0.5 puntos). Dibuje un camino de datos de un único bus y varios registros de propósito general, y escriba (en lenguaje de transferencia de registros o de alto nivel) la parte del microprograma correspondiente a la fase de captación de instrucción.

**3. Entrada/Salida** (0.8 puntos). Una placa Elegoo Mega2560 está conectada a un led RGB como se indica en la figura (rojo a patilla 6, verde a patilla 5 y azul a patilla 3):



Dentro del LED RGB en realidad hay tres LED, uno rojo, uno verde y uno azul. Al controlar el brillo de cada uno de los LED individuales, puede obtenerse prácticamente cualquier color que se desee. Si configuramos el brillo de los tres LED para que sea el mismo, entonces el color de la luz será blanco. El negro no es tanto un color como una ausencia de luz. Por lo tanto, lo más cerca que podemos llegar al negro con nuestro LED es apagar los tres colores.

Imagine que añadimos al diseño de la figura una fotorresistencia en la patilla de entrada analógica A0. Escriba un programa que lea el valor de la fotorresistencia y encienda en color blanco el led RGB de modo que con poca luz captada por la fotorresistencia el LED se apague por completo y con la máxima luz ambiental el LED se encienda al valor máximo. Recuerde que un programa en Arduino consta de una función **setup()** que se ejecuta una vez al reiniciar la placa y una función **loop()** que repite las sentencias que contiene en un bucle infinito. Puede usar las siguientes funciones y **#define** para las patillas.

```
pinMode(pin, mode);           // mode puede ser INPUT, OUTPUT o INPUT_PULLUP

value = analogRead(pin);      // pin puede ser A0      0 <= value <= 1023
                             // no hay que configurar pin con pinMode

analogWrite(pin, value);     // 0 <= value <= 255, hay que configurar...
                             // ...pin como salida con pinMode en setup()
```

**4. Diseño del sistema de memoria** (0.5 puntos). Disponemos de un procesador con buses de datos y direcciones de 16 bits. Diseñe un sistema de memoria de 128 KB direccionable por palabras de 16 bits a partir de módulos SRAM de 16Kx8 y ROM de 8Kx4. La memoria SRAM debe ocupar las direcciones 0x0000 a 0xBFFF y la ROM 0xC000 a 0xFFFF.

**5. Memoria cache** (0.6 puntos). Las características de la memoria cache L1 de instrucciones de un microprocesador son las siguientes:

Tamaño de línea	64 B
Tamaño de L1 (instrucciones)	64 KB, asociativa de 4 vías
Memoria física	1 TB

Indique el nombre y tamaño en bits de los campos de dirección usados para la política de correspondencia, así como el tamaño total en bits de la memoria de etiquetas (directorio), tamaño total en bits de la memoria de instrucciones, y porcentaje de espacio de etiquetas respecto a instrucciones:

- a) Dirección física de memoria principal desde el punto de vista de L1-instrucciones:

- b) Tamaño total en bits ocupado por las etiquetas en directorio L1-instrucciones:

- c) Tamaño total en bits ocupado por las instrucciones en L1-instrucciones:

- d) Porcentaje de espacio ocupado por etiquetas respecto a instrucciones en L1-instrucciones:

### Test de Teoría (3.0p)

1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30
d	c	c	b	c	c	c	d	d	a	a	b	d	b	a	a	c	d	d	a	c	a	a	b	b	c	b	a	d	c

### Test de Prácticas (4.0p)

1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20
b	c	b	c	c	b	d	d	a	d	a	d	c	b	d	c	a	d	d	b

Las preguntas 16 y 18 tenían lapsus en su redacción que no pudimos detectar a tiempo: kit por entorno, y 2520 por 2025. A todos los que entregaron se les puntuó como acertadas, independientemente de lo que contestaran, independientemente de si contestaron.

### Examen de Problemas (3.0p)

La pregunta 1.a tenía un lapsus en su redacción que no pudimos detectar a tiempo: utilizar las dimensiones del array para nombrar el campo, surgiendo la ambigüedad de si se pide un desplazamiento “ficticio” a un elemento inexistente, o desplazamiento “real” al principio del campo de dicho nombre (equivalente al índice 0 del array). Ambas alternativas puntúan como válidas, siempre que no se mezclen respuestas de una y otra alternativas.

#### 1. Estructuras (0.6 puntos).

Se puntuó 0.06p por cada número (total  $0.06 \times 10 = 0.60p$ ).

- a) Alternativas: la que se pretendía redactar, y la que se acepta por el lapsus.

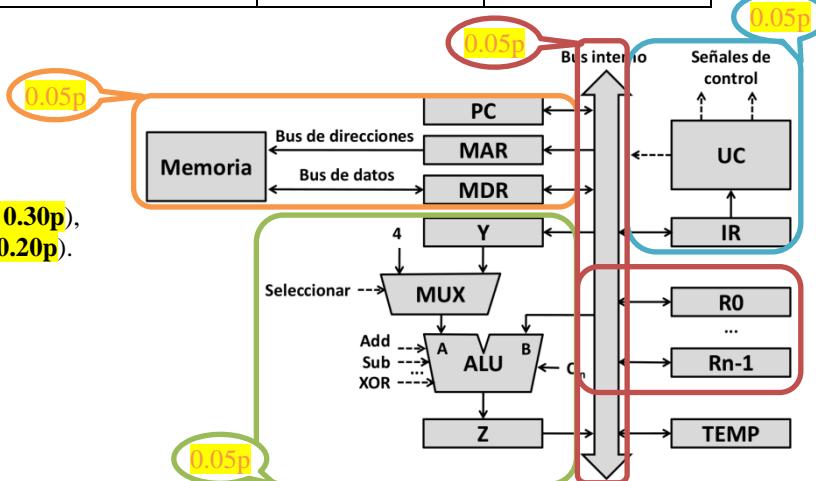
Tipo	Campo	Desplazamiento	Desplazamiento
long	Courses[0].CrsCod;	0	0
char	Courses[0].InstitutionalCrsCod[0/MAX...];	8	263
long	Courses[0].DegCod;	264	264
unsigned	Courses[0].Year;	272	272
unsigned	Courses[0].Status;	276	276
long	Courses[0].RequesterUsrCod;	280	280
unsigned	Courses[0].NumUsrs[0/10];	288	328
char	Courses[0].ShrtName[0/MAX...];	328	583
char	Courses[0].FullName[0/MAX...];	583	838
long	Courses[1].CrsCod;	840	840

b) 256

#### 2. Unidad de Control (0.5 puntos).

Se puntuó 0.05p por pseudo-instrucción (total  $0.05 \times 6 = 0.30p$ ), 0.05p por cada zona dibujada (total  $0.05 \times 4 = 0.20p$ ).

```
FETCH: MAR := PC; Z := PC+4
MDR := M[MAR]; PC := Z
IR := MBR
goto f(IR)
```



### 3. Entrada/Salida (0.8 puntos).

Se puntuá si el programa funciona, o alternativamente **0.1p** por cada sentencia (total **0.1 x 8 = 0.8p**).

```
#define SENSOR      A0
#define RED         6
#define GREEN        5
#define BLUE         3

void setup()
{
    pinMode(RED, OUTPUT);
    pinMode(GREEN, OUTPUT);
    pinMode(BLUE, OUTPUT);
}

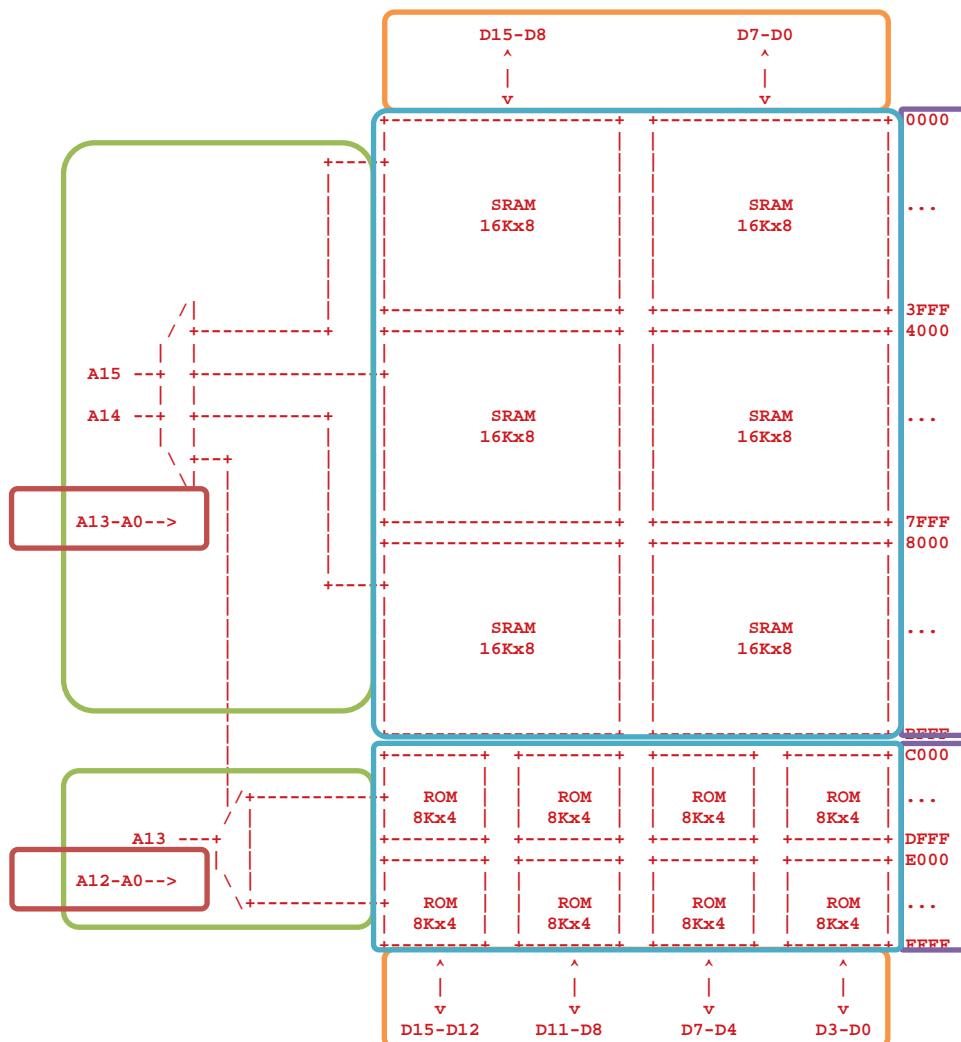
void loop()
{
    // leer el valor del sensor
    int sensorValue = analogRead(SENSOR);

    // mapear el valor del sensor (de 0 a 1023) a un valor de luz (de 0 a 255)
    int light = sensorValue / 4;

    // escribir el valor de luz en el LED RGB
    analogWrite(RED, light);
    analogWrite(GREEN, light);
    analogWrite(BLUE, light);
}
```

### 4. Diseño del sistema de memoria (0.5 puntos).

Se puntuá **0.05p** por cada zona dibujo (total **0.05 x 10 = 0.50p**).



## 5. Memoria cache (0.6 puntos).

Se puntuá **0.1p** por cada número (total **0.1 x 6 = 0.6p**).

MP: 1 TB =  **$2^{40}$  B**

L1-instrucciones:  $64 \text{ KB} = 2^{16} \text{ B}$ ,  $64 \text{ B/línea} = 2^6 \text{ B/línea}$ ,  $4 \text{ vías} = 2^2 \text{ líneas/conjunto}$

L1-instrucciones:  $2^{16} \text{ B} / 2^6 \text{ B/línea} = 2^{10} \text{ líneas}$ ,  $2^{10} \text{ líneas} / 2^2 \text{ líneas/conjunto} = 2^8 \text{ conjuntos}$

Dirección física de memoria principal desde el punto de vista de L1 (instrucciones):

etiqueta (26)	conjunto (8)	byte (6)
---------------	--------------	----------

Tamaño total en bits ocupado por las etiquetas en el directorio L1-instrucciones:

$$2^{10} \text{ líneas} \cdot 26 \text{ bits/etiqueta} = 2^{10} \times 26 \text{ bits} = \mathbf{26\,624 \text{ bits}}$$

Tamaño total en bits ocupado por las instrucciones en L1-instrucciones:

$$64\text{KB/cache} \cdot 8 \text{ bits/B} = 2^{16} \times 2^3 \text{ bits} = 2^{19} \text{ bits} = \mathbf{524\,288 \text{ bits}}$$

$$\text{Etiquetas / Instrucciones} = 2^{10} \times 26 / 2^{19} = 26 / 2^9 = \mathbf{5,078\%}$$

**Nombre:**
**DNI:**
**Grupo:**

### Test de Teoría (3.0p)

Todas las preguntas son de elección simple sobre 4 alternativas.

Cada respuesta vale 0.1p si es correcta, 0p si está en blanco o claramente tachada, -0.03p si es errónea.

Anotar las respuestas (a, b, c ó d) en la siguiente tabla.

1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30

1. En la memoria de un procesador de la familia Intel 64 (x86-64) se almacena a partir de la dirección N los siguientes contenidos: 0xff, 0xff, 0xff, 0xff, 0x00, 0x00, 0x00, 0x01, 0xff, 0xff, 0xff, 0x00, 0x00, 0x00, 0x02. Posteriormente se lee (carga) %rax desde (a partir de) la dirección N. ¿Cómo es %rax entonces, considerándolo con signo?
- a. menor que -2000 millones
  - b. entre -2000 millones y -1
  - c. entre 0 y 2000 millones
  - d. mayor que 2000 millones
- 
2. ¿Cuál de las siguientes afirmaciones es incorrecta?
- a. El direccionamiento indexado es útil para manejo de estructuras
  - b. El direccionamiento indirecto indica un puntero al operando
  - c. En el direccionamiento implícito no se indica la ubicación del operando
  - d. En el direccionamiento inmediato el dato se encuentra en la propia instrucción
- 
3. Si rcx vale -1, tras ejecutar la instrucción `sal $3,%ecx` el nuevo valor de RCX es
- a. 0xffff ffff ffff ffff
  - b. 0xffff ffff ffff fff8
  - c. 0x1fff ffff ffff ffff
  - d. 0xffff ffff 8
- 

4. ¿Cuál de las siguientes instrucciones **NO** es errónea?

- a. `movb $0xF, (%rax)`
  - b. `movw (%rbx), 4(%rsp)`
  - c. `movl %rcx, (%rsp)`
  - d. `movq %rdx, $0x123`
- 

5. ¿Cuál de las siguientes instrucciones es **errónea**?

- a. `mov (%rax,%rbx,1), %rcx`
  - b. `lea (%rdx,%rsi,2), %rdi`
  - c. `lea (%rbp,%rsp,4), %r8`
  - d. `mov (%r9,%r10,8), %r11`
- 

6. La instrucción `movzlx %eax, %rax`

- a. pone a 0 el registro %rax
  - b. copia en %rax el valor sin signo almacenado en %eax, rellenando con ceros
  - c. copia en %rax el valor de %eax si el indicador de cero está activado
  - d. no existe, se debe usar `mov %eax, %eax`
- 

7. Sabiendo que las instrucciones de salto condicional codifican la dirección de salto con direccionamiento relativo a contador de programa (de 8 o 32 bits con signo), indicar cuál es la dirección de la instrucción pop en el siguiente desensamblado, donde se ha tachado la parte de las direcciones.

```
xxxxxx: 77 02 ja 400547
xxxxxx: 5d pop %rbp
```

- a. 400547
  - b. 400549
  - c. 400545
  - d. 400543
-

8. La instrucción seta %al (seta significa set if above):

- a. pone AL a 1 si ZF=1 y CF=1
  - b. pone AL a 1 si ZF=1 o CF=0
  - c. pone AL a 1 si ZF=0 o CF=1
  - d. pone AL a 1 si ZF=0 y CF=0
- 

9. La instrucción JNGE / JL provoca un salto si...

- a. SF = 1
  - b. CF = 1
  - c. SF ≠ OF
  - d. SF = CF
- 

10. Para traducir una asignación condicional (**a=b ? c : d ;**) de lenguaje C a lenguaje ensamblador, gcc puede que utilice...

- a. Un salto condicional, según la condición expresada en el código C, y otro salto incondicional
  - b. Un salto incondicional, según la condición opuesta a la del código C, y otro salto condicional
  - c. Una instrucción de movimiento incondicional, pero sólo si el procesador es Pentium Pro/II o superior
  - d. Una instrucción de movimiento condicional, pero sólo si el S.O. es de 32 bits
- 

11. El rasgo distintivo de la traducción “salta-en-medio” que gcc hace de un bucle while de lenguaje C a lenguaje ensamblador es...

- a. el salto incondicional hacia adelante
  - b. el salto incondicional hacia atrás
  - c. el salto condicional hacia adelante
  - d. el salto condicional hacia atrás
- 

12. El rasgo distintivo de la traducción “copia-test” que gcc hace de un bucle while de lenguaje C a lenguaje ensamblador es...

- a. el salto condicional hacia adelante, según la misma condición expresada en lenguaje C
  - b. el salto condicional hacia atrás, según la misma condición expresada en lenguaje C
  - c. el salto condicional hacia adelante, según la condición opuesta a la expresada en leng. C
  - d. el salto condicional hacia atrás, según la condición opuesta a la expresada en leng. C
- 

13. En la convención de llamada SystemV AMD64 seguida por gcc Linux/x86-64...

a. R12 es un registro salva-invocado, por eso en cualquier función hay que salvarlo antes de modificarlo

b. R11 es un registro salva-invocante, por eso en cualquier función hay que salvarlo antes de modificarlo

c. RBX es un registro salva-invocado, por eso si es necesario conservar su valor hay que salvarlo antes de llamar a función

d. RBP es un registro salva-invocante, por eso si es necesario conservar su valor hay que salvarlo antes de llamar a función

---

14. En el fragmento de programa siguiente:

```
66b: e8 00 00 00 00 callq 670 <nxt>
670: 58          pop %rax
```

¿Qué valor termina almacenado en %eax?

- a. 0x670
  - b. 0xe8000000
  - c. 0x66b
  - d. 0x58
- 

15. ¿Cuál sería el “equivalente x86-64” del “pseudo-código C” rcx=((long\*)rax)[rbx]?

- a. mov (%rax,%rbx,4), %rcx
  - b. lea (%rax,%rbx,4), %rcx
  - c. lea (%rax,%rbx,8), %rcx
  - d. mov (%rax,%rbx,8), %rcx
- 

16. Habiendo declarado **int array={-4,-3,-2,-1};** y **char \*ptr=array;** ¿cuánto vale **ptr[1]**?

- a. -1
  - b. -2
  - c. -3
  - d. -4
- 

17. En una unidad de control microprogramada, bits del registro IR direccionan una ROM cuya salida puede cargarse en el registro μPC (micro-PC). Esa unidad de control...

- a. es una UC con secuenciamiento explícito
  - b. puede realizar la micro-operación IR=μPC
  - c. puede realizar la μ-op. IR=ROM[MAR]
  - d. dispone de la funcionalidad “goto f(IR)”
- 

18. Para el procesador con unidad de control microprogramada estudiado en clase, Tanenbaum propone codificar los N registros y añadir una señal “PERC” para habilitar la carga desde el bus C (recordar que era un diseño típico con 3 buses) y así

no perder expresividad/paralelismo. Si fuera  $N=8$ , el ahorro de bits en cada microinstrucción debido a esta técnica es de

- a. 35 bits
  - b. 34 bits
  - c. 25 bits
  - d. 14 bits
- 

19. Para el procesador con unidad de control microprogramada estudiado en clase, Tanenbaum propone codificar 16 microoperaciones sobre 2 de los 16 registros en lugar de especificar directamente las 24 señales de control y los 8 bits de dirección, aunque así se pierda paralelismo. El ahorro de bits en cada microinstrucción debido a esta técnica es de

- a. 24 bits
  - b. 20 bits
  - c. 12 bits
  - d. 8 bits
- 

20. Respecto a saltos retardados y anulantes, condicionales o no, **NO** sería apropiado intentar reordenar instrucciones...

- a. anteriores al condicional retardado para ponerlas después (en memoria)
  - b. en el destino del salto incondicional retardado para ponerlas después (en mem.) del salto
  - c. anteriores al condicional anulante para ponerlas después (en memoria)
  - d. en el destino del salto condicional anulante para ponerlas después (en mem.) del salto
- 

21. Una SRAM de 128Kx8bit (1Mbit) puede venir organizada en 1024 filas, dedicando por tanto al decodificador de columnas...

- a. 7 bits
  - b. 8 bits
  - c. 9 bits
  - d. 10 bits
- 

22. Respecto a las técnicas de direccionamiento por selección lineal, decodificación centralizada y distribuida

- a. todas ellas impiden que haya cortocircuito en el bus de datos
- b. todas ellas impiden que haya cortocircuito en el bus de direcciones

c. la selección lineal permitiría leer una palabra simultáneamente desde varios puertos de E/S

d. usando decodificación distribuida es más fácil realizar expansiones al sistema de E/S

---

23. La instrucción máquina DI (Disable Interrupts), conocida como CLI (Clear Interrupt Flag) en x86, se utiliza para desactivar:

- a. algunas interrupciones no enmascarables
  - b. algunas interrupciones externas
  - c. algunas excepciones o *traps*
  - d. algunos niveles de interrupción selectivamente
- 

24. ¿Cuál de las siguientes características es menos probable que pueda programarse en un canal DMA?

- a. dirección de memoria y dirección de E/S
  - b. si la dirección de la transferencia es hacia memoria, o hacia E/S
  - c. contador de memoria y contador de E/S
  - d. si la dirección de memoria se incrementa o decrementa, y si la dirección de E/S se incrementa o decrementa
- 

25. El ancho de banda de memoria es:

- a. el número de bits del bus de datos en el bus del sistema
  - b. la reunión (en paralelo) de los buses de datos de los módulos que forman la memoria
  - c. la velocidad a la que se pueden leer o escribir los datos en memoria
  - d. el intervalo de frecuencias de reloj permitidas para la CPU que vaya a conectarse a dicha memoria
- 

26. Una memoria estática tiene un bus de datos de 64 bits y su bus de direcciones es de 30 bits, ¿cuál es su capacidad?

- a. 80 MBytes
  - b. 1 GByte
  - c. 8 GBytes
  - d. 64 GBytes
- 

27. ¿Cuántas líneas de dirección (patillas) son necesarias para direccionar un chip de memoria DRAM de 16M x 16?

- a. 10
- b. 12

c. 24

d. 28

---

28. En un computador con una jerarquía de memoria de dos niveles se observa para una carga concreta que el tiempo medio de acceso a la memoria es de 900 ns cuando en realidad el tiempo de acceso al primer nivel es de 9 ns. Sabiendo que el tiempo de acceso al segundo nivel es de 3 microsegundos, ¿cuál sería aproximadamente el porcentaje de fallos en los accesos al primer nivel?

a. 90%

b. 20%

c. 10%

d. 30%

---

29. En una cache con correspondencia directa de  $2^p$  palabras y líneas de  $2^w$  palabras, el gestor de memoria **no** considera como campo (conjunto de bits contiguos con significado o relevancia) los siguientes bits:

a. últimos w bits (0..w-1) (los menos significativos)

b. bits w...w+p-1

c. bits w...p-1

d. primeros bits, desde el más significativo hasta el bit p

---

30. Un sistema monoprocesador con memoria de bytes y direcciones de 32 bits dispone de un único nivel de cache L1 compartida (instrucciones y datos) asociativa por conjuntos de 256KB. El tamaño del campo de etiqueta es

a. 14 bits

b. depende del tamaño de línea

c. depende del número de vías

d. depende del número de vías y del tamaño de línea

---

**Nombre:**
**DNI:**
**Grupo:**

## Test de Prácticas (4.0p)

**Todas las preguntas son de elección simple sobre 4 alternativas.**

**Cada respuesta vale 0.2p si es correcta, 0 si está en blanco o claramente tachada, -0.06p si es errónea.**

**Anotar las respuestas (a, b, c ó d) en la siguiente tabla.**

1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20

1. Habiendo definido en código fuente ensamblador `saludo: .ascii "Hola mundo\n"`, si comparamos los comandos gdb siguientes:  
`x/s &saludo frente a print (char*) &saludo,`  
 se puede decir que:

- a. ambos nos muestran el string (el mensaje “Hola mundo”)
- b. el primero (x) nos muestra el string, y el segundo (print) nos muestra otro valor distinto
- c. el segundo (print) nos muestra el string, y el primero (x) nos muestra otro valor distinto
- d. alguno o ambos contienen algún error gramatical (falta o sobra algún &, algún typecast (char\*) ó (long), etc)

2. En la práctica “media” se pide sumar una lista de 16 enteros SIN signo de 32 bits evitando acarreo. ¿Cuál es el **mayor** valor que repetido en toda la lista **no** causaría acarreo en 32 bits?

- a. 0xFFFF FFFF
- b. 0x7FFF FFFF
- c. 0x1000 0000
- d. 0x0FFF FFFF

3. En la práctica "media" se pide usar `cltq/cdq` y `cqto/cqo` para hallar la media y resto de una lista de 16 enteros CON signo de 32 bits usando registros de 64 bits. Un estudiante entrega la siguiente versión:

```
...
media: .int      0
resto: .int      0
...
main: .global main
    mov     $lista, %rbx
    mov     longlista, %ecx
...
```

```
call  suma
mov   %rax, media
mov   %edx, resto
...
suma:
mov   $0, %rax
mov   $0, %rsi
mov   $0, %edx
mov   $0, %rdi
mov   $0, %ebp
bucle:
    movl  (%rbx,%rsi,4), %eax
    cltq
    #adc  %edx, %ebp
    inc   %rsi
    cmp   %rsi,%rcx
    jne   bucle
    mov   %rdi, %rax
    #mov  %ebp, %edx
    idiv %rcx
    ret
```

Este programa se diferencia de la versión "oficial" recomendada en clase en que se guarda como media todo RAX, la subrutina no se llama **sumaq**, se inicializan a 0 más registros y no se usa CQTO.

¿Qué media y resto calcula este programa para el test #02? (16 elems. con valor -1,-2,-1,-2...)

- a. media = -1, resto = -8
- b. media = -2, resto = 8
- c. media = -1, resto = 8
- d. media = -2, resto = -8

4. En la misma práctica "media" (`cltq/cdq` y `cqto/cqo`), un estudiante entrega la siguiente versión:

```
sumaq:
    mov   $0, %edi
    mov   $0, %esi
```

```

bucleq:
    mov    (%rbx,%rsi,4), %eax
    cltq          # EAX -> RAX
    add    %rax, %rdi
    inc    %esi
    cmp    %esi, %ecx
    jne    bucleq

    mov    %rdi, %rax
    cqto           # RAX -> RDX:RAX
#idiv/idivq %rcx no hace la division,
#queda todo como resto
    idiv    %ecx
    ret

```

Esta subrutina se diferencia de la versión "oficial" recomendada en clase en que se nombran EDI, ESI y ECX. El propio estudiante comentó alguna de las diferencias. ¿Qué media y resto calcula este programa para el test #03? (16 elementos con valor 0x7fffffff)

- a. media = 0x0fffffff, resto = 0
- b. media = 0x7fffffff, resto = 0
- c. media = 0xffffffff, resto = 0
- d. resto distinto de cero

5. En la misma práctica "media" (`cltq/cdq` y `cqto/cqo`), un estudiante entrega la siguiente versión:

```

...
media:      .int 0
resto:      .int 0
formato:    .asciz "..."
formatoq:   .asciz "..."

...
    mov    $lista, %rbx
    mov    longlista, %ecx
    call  sumaq
    mov    %rax, media
    mov    %rdx, resto
    ...

sumaq:
    mov    $0, %rax
    mov    $0, %rsi
    mov    $0, %rdx
    mov    $0, %rdi
    mov    $0, %rbp

bucleq:
    mov    (%rbx,%rsi,4), %rax
    cqo          # RAX -> RDX:RAX
    add    %rax, %rdi
    adc    %rdx, %rdi

    inc    %rsi
    cmp    %rsi,%rcx
    jne    bucleq

    mov    %rdi, %rax
#    mov    %rbp, %rdx
#    mov    $0, %ebp
#    mov    %ecx,%ebp
#    idiv    %ebp
    ret

```

Este programa es muy diferente a la versión "oficial" recomendada en clase, destacando especialmente los `mov media/resto` tras la

llamada a `sumaq`, los `mov $0` adicionales superfluos, mover elementos a `%rax`, usar `cqo` en lugar de `cltq` y no usarlo antes de `idiv %ebp`, que tampoco está bien, y además sobra el `adc`.

Este programa calcula el resultado correcto:

- a. para lista: `.int 1, 2, 1, 2, ... (16 elementos)`
- b. para lista: `.int -1,-2,-1,-2, ... (16 elementos)`
- c. para ambos ejemplos
- d. para ninguno de los dos ejemplos

6. La práctica "popcount" debía calcular la suma de bits (peso Hamming) de los elementos de un array. Un estudiante entrega la siguiente versión de `popcount4`:

```

int pc4(unsigned* array, size_t len){
    size_t i;
    int res=0;
    unsigned x;
    for (i=0; i<len; i++){
        x = array[i];
        asm("\n\t"
            "clc                           \n"
            "ini4:                         \n\t"
            "    shr  %[x]                 \n\t"
            "    adc  $0,  %[r]             \n\t"
            "    test %[x],%[x]            \n\t"
            "    jne  ini4                \n"
            "fin4:                         \n\t"
            "    clc                           \n\t"
            "    : [r]"+r" (res)          \n"
            "    : [x] "r" (x)            );
    }
    return res;
}

```

Esta función es una mezcla de las versiones "oficiales" recomendadas en clase para `popcount3` y `popcount4`, con alguna instrucción cambiada. Esta función `popcount4`:

- a. produce siempre el resultado correcto
- b. fallaría con `array={0,1,2,3}`
- c. fallaría con `array={1,2,4,8}`
- d. no es correcta pero el error no se manifiesta en los ejemplos propuestos, o se manifiesta en ambos

7. En la misma práctica "popcount" un estudiante entrega la siguiente versión de `popcount4`:

```

int pc4(unsigned* array, size_t len){
    size_t i;
    int res=0;
    unsigned x;
    for (i=0; i<len; i++){
        x = array[i];
        asm("\n\t"
            "clc                           \n"
            "ini4:                         \n\t"
            "    adc  $0,  %[r]             \n\t"
            "    shr  %[x]                 \n\t"
            "    jnz  ini4                \n"
            );
    }
}

```

```

        : [r]" + r" (res)
        : [x] "r" (x)      );
    }
    return res;
}

```

Esta función se diferencia de la versión "oficial" recomendada en clase en que le falta una instrucción ensamblador final (última).

Esta función `popcount4` (pensar bien qué pasa con el elemento 0 y con el elemento 1):

- a. produce siempre el resultado correcto
  - b. fallaría con `array={0,1,2,3}`
  - c. fallaría con `array={1,2,4,8}`
  - d. no es correcta pero el error no se manifiesta en los ejemplos propuestos, o se manifiesta en ambos**
- 

8. En la misma práctica "popcount" un estudiante entrega la siguiente versión de `popcount5`:

```

int pc5(unsigned* array, int len){
    size_t i, k, x;
    int res=0;
    for (i=0; i<len; i++){
        long val = 0;
        x = array[i];
        for (k=0; k<8; k++){
            val+= x& 0x0101010101010101L;
            x >>= 1;
        }
        val += (val >> 32);
        val += (val >> 16);
        val += (val >> 8);
        res += (val & 0xFF);
    }
    return res;
}

```

Esta función no sigue la adaptación a 32 bits recomendada en clase como versión "oficial", y conserva muchos vestigios de la versión de 64 bits, como el tipo de `val`, `len` y `x`, la constante `0x01...01L` y el desplazamiento `>>32`.

Esta función `popcount5`:

- a. produce siempre el resultado correcto**
  - b. fallaría con `array={0,1,2,3}`
  - c. fallaría con `array={1,2,4,8}`
  - d. no es correcta pero el error no se manifiesta en los ejemplos propuestos, o se manifiesta en ambos
- 

9. En la misma práctica "popcount" un estudiante entrega la siguiente versión de `popcount5`:

```

int pc5(unsigned* array, size_t len){
    size_t i,j;
    long val=0;
    unsigned long x;
    for (i=0; i<len; i++){
        x = array[i];
        for (j=0; j<8; j++){
            val+= x& 0x0101010101010101L;
            x >>= 1;
        }
    }
}

```

```

    val += (val >> 16);
    val += (val >> 8);
    val += (val >> 4);
    return val & 0xFF;
}

```

Esta función es muy diferente a la versión "oficial" recomendada en clase, destacando especialmente el tipo de `val` y `x`, la inicialización de `val`, la variable `res`, la constante `0x01...01L`, el anidamiento de los desplazamientos y el desplazamiento `>>4`.

Esta función `popcount4` da resultado correcto:

- a. con `array={0,1,2,3}`
  - b. con `array={1,2,4,8}`
  - c. con ambos ejemplos**
  - d. con ninguno de los dos ejemplos
- 

10. En la misma práctica "popcount" un estudiante entrega la siguiente versión de `popcount7`:

```

int pc7(unsigned* array, size_t len){
    size_t i;
    unsigned long x1,x2;
    int res=0;
    const unsigned long m1 = 0x5555555555555555;
    const unsigned long m2 = 0x3333333333333333;
    const unsigned long m4 = 0x0f0f0f0f0f0f0f0f;
    const unsigned long m8 = 0x00ff00ff00ff00ff;
    const unsigned long m16= 0x0000ffff0000ffff;
    const unsigned long m32= 0x00000000ffffffff;

    if (len & 0x3)
        printf("len no múltiplo de 4\n");

    for (i=0; i<len; i+=4){
        x1= *(unsigned long*)&array[i];
        x2= *(unsigned long*)&array[i+2];

        x1 = (x1 & m1 ) + ((x1 >> 1) & m1 );
        x1 = (x1 & m2 ) + ((x1 >> 2) & m2 );
        x1 = (x1 & m4 ) + ((x1 >> 4) & m4 );
        x1 = (x1 & m8 ) + ((x1 >> 8) & m8 );
        x1 = (x1 & m16) + ((x1 >>16) & m16);
        x1 = (x1 & m32) + ((x1 >>32) & m32);

        x2 = (x2 & m1 ) + ((x2 >> 1) & m1 );
        x2 = (x2 & m2 ) + ((x2 >> 2) & m2 );
        x2 = (x2 & m4 ) + ((x2 >> 4) & m4 );
        x2 = (x2 & m8 ) + ((x2 >> 8) & m8 );
        x2 = (x2 & m16) + ((x2 >>16) & m16);
        x1 = (x1 & m32) + ((x1 >>32) & m32);

        res += x1+x2;
    }
    return res;
}

```

Esta función sólo se diferencia de la versión "oficial" recomendada en clase en la última máscara, suma y desplazamiento `>>32`, que se realizan sobre una variable distinta. Esta función `popcount4` da resultado correcto:

- a. con `array={3,2,1,0}`**
- b. con `array={8,4,2,1}`
- c. con ambos ejemplos

d. con ninguno de los dos ejemplos

**11.** Invocando a printf de libC (SystemV AMD64) desde ensamblador...

- a. el primer argumento debe ponerse en %rax
- b. el segundo argumento es el formato
- c. si se desean imprimir tres **long**, el tercero debe ponerse en %rdx
- d. si se desean imprimir cuatro **int**, el cuarto debe ponerse en %r8d**

**12.** Recordando que los argumentos de **\_start**

(argc /argv y variables de entorno) se pasan en pila en SysV AMD64, si en el Ejemplo 1 de la Práctica 3 se hace **gdb -tui --args suma\_01 uno dos tres**, se lanza con **br \_start y run**, y se teclea **p \* (char\*\*)(\$rsp+40)**, ¿qué obtenemos?

- a. tres
- b. 0x0**
- c. una variable de entorno
- d. un error de gdb

**13.** En la práctica de la bomba, el primer ejercicio consistía en “saltarse” las “explosiones”, para lo cual se puede utilizar...

- a. objdump o gdb
- b. gdb o eclipse**
- c. eclipse o ghex
- d. ghex u objdump

**14.** En una bomba como las estudiadas en prácticas, del tipo...

```
0x40079b <main+64> lea    0x30(%rsp),%rdi
0x4007a0 <main+69> mov    0x2008d9(%rip),
                           %rdx # 0x601080
0x4007a7 <main+76> mov    $0x64,%esi
0x4007ac <main+81> call   0x400600 <fgets>
0x4007b1 <main+86> test   %rax,%rax
0x4007b4 <main+89> je    0x400785 <main+42>
0x4007b6 <main+91> lea    0x30(%rsp),%rdi
0x4007bb <main+96> mov    $0xd,%edx
0x4007c0 <main+101> lea   0x2008a1(%rip),
                           %rsi # 0x601068
0x4007c7 <main+108> call  0x4005d0<strncmp>
0x4007cc <main+113> test  %eax,%eax
0x4007ce <main+115> je   0x4007d5<main+122>
0x4007d0 <main+117> call  0x400727 <boom>
0x4007d5 <main+122> lea   0x20(%rsp),%rdi
```

...la contraseña (alfanumérica) es...

- a. el string almacenado a partir de 0x601068
- b. el string alm. a partir de 0x2008a1+0x4007c7
- c. el string almacenado a partir de 0x30(%rsp)
- d. no se puede marcar una y sólo una de las respuestas anteriores**

**15.** En la práctica de E/S en Arduino, la instrucción CBI del repertorio del microcontrolador realiza...

- a. una comparación
- b. un complemento a uno
- c. una operación de bits**
- d. una llamada a subrutina

**16.** En la práctica de E/S en Arduino, la instrucción SBI del repertorio del microcontrolador realiza...

- a. una suma de un valor inmediato
- b. una resta de un valor inmediato
- c. una operación de bits**
- d. un salto incondicional

**17.** En la práctica de E/S en Arduino, DDRB es

- a. el segundo canal de memoria DDR
- b. el registro de dirección de datos del puerto B**
- c. el registro de datos y direcciones B
- d. el registro buffer de datos D

**18.** En la práctica de E/S en Arduino, la instrucción SBIW del repertorio del microcontrolador realiza...

- a. una suma de un valor inmediato
- b. una resta de un valor inmediato**
- c. una operación de bits
- d. un salto incondicional

**19.** En el programa line.cc de la práctica de cache, para cada tamaño de línea (**line**) recorremos una única vez el vector, pero podríamos haber realizado un número fijo y grande de accesos. Al dibujar la gráfica del tiempo de iteración en función del tamaño de línea...

- a. de ambas formas sale gráfica creciente
- b. recorriendo una vez sale gráfica creciente
- c. con núm. fijo accesos sale gráfica creciente**
- d. de ninguna de las dos formas sale gr. creciente

**20.** En la práctica de la cache, en size.cc se realiza un número fijo y grande de accesos, pero podríamos haber recorrido una única vez el vector (saltando también de 64 en 64). Al dibujar la gráfica del tiempo de bucle en función del tamaño del vector...

- a. de ambas formas sale gráfica creciente**
- b. recorriendo una vez sale gráfica creciente
- c. con núm. fijo accesos sale gráfica creciente
- d. de ninguna de las dos formas sale gr. creciente

Nombre:	
DNI:	Grupo:

### Examen de Problemas (3,0 p)

**1. Ensamblador.** (1 punto). Suponga una función f que acepta un argumento de tipo unsigned long y devuelve un resultado de tipo unsigned long:

```
unsigned long f(unsigned long x){
    return operación-sobre-x;
}
```

Escriba en la segunda columna de la siguiente tabla la implementación en lenguaje máquina x86-64 del cuerpo de cada función f indicado en la columna izquierda.

Restricciones:

1. No puede usar las instrucciones mul/imul ni div/idiv.
2. No escriba la instrucción ret final.
3. Puede escribir como mucho tres instrucciones máquina en cada fila.

Cuerpo de la función f	Implementación de f en lenguaje máquina (no escriba la instrucción ret final, puede escribir como mucho tres instrucciones para cada función)
<code>return x + x;</code>	
<code>return x * 5;</code>	
<code>return x * 8;</code>	
<code>return x * 16;</code>	
<code>return x / 16;</code>	
<code>return x * 12;</code>	
<code>return x % 8;</code>	
<code>return x &amp; 0xffffffff;</code>	
<code>return x &amp; 1;</code>	
<code>return x != 0;</code>	

**2. Unidad de control** (0.4 puntos). Un procesador con una unidad de control microprogramada tiene una memoria de control de 2048 palabras de 41 bits, de las cuales 100 son diferentes. ¿Qué ahorro en número de bits obtendríamos si usáramos nanoprogramación? Razona la respuesta con dos dibujos, uno sin nanoprogramación y otro con nanoprogramación.

**3. Entrada/Salida** (0.6 puntos). Disponemos de un microprocesador de 8 bits (bus de datos de 8 bits y bus de direcciones de 16 bits) con E/S independiente. Diseñe un sistema de E/S que permita acceder a los siguientes puertos: puerto 0x0440 de salida y 0x0441 de entrada. Utilice lógica de decodificación distribuida. No emplee decodificadores.

**4. Mapa de memoria** (0.6 puntos). Una implementación de la arquitectura ARM9 tiene un mapa de memoria de 4 GB dividido en las zonas indicadas más abajo (comenzando por la dirección 0 hasta la dirección 0xFF...FF):

- ¿Cuál es el valor de n (número de zonas de tamaño 256 MB situadas en la parte más alta del mapa de memoria)?
- Indique la primera y última dirección en hexadecimal de cada una de las zonas del mapa de memoria indicadas en la siguiente tabla:

.....	Zona de 4 MB
.....	Zona de 4 MB
.....	Zona de 248 MB
.....	Zona 1 de 256 MB
.....	...
.....	Zona <i>n</i> de 256 MB

**5. Memoria cache** (0.4 puntos). Una implementación de la arquitectura ARM9, cuya memoria es direccionable por bytes, tiene una memoria cache de 4KB de instrucciones y otra de datos del mismo tamaño. Ambas caches son asociativas por conjuntos con 4 vías, con un tamaño de línea de 8 palabras (32 bytes). El procesador puede direccionar 4 GB de memoria física. Muestre un esquema de una dirección de memoria física dividida en los campos necesarios para el acceso a cache, con el nombre y tamaño de cada campo.

Dirección física de memoria principal desde el punto de vista de la cache:

### Test de Teoría (3.0p)

1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30
d	a	d	a	c	d	c	d	c	a	a	c	a	a	d	a	d	b	c	a	d	b	c	c	c	b	d	b	c	

### Test de Prácticas (4.0p)

1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20
a	d	b	a	a	a	d	a	c	a	d	b	b	d	c	c	b	b	c	a

### Examen de Problemas (3.0p)

#### 1. Ensamblador (1 puntos).

Se puntuá **0.1p** por cada función (total **0.1 x 10 = 1p**).

Múltiples soluciones válidas, cualquiera de ellas puntuá,  
 sólo se sugieren algunas de ellas, en primer lugar la generada por gcc -Og.  
 Si el resultado es correcto y cumple las restricciones no se pide más.

Algunas instrucciones u operandos mostrados como alternativas **no se han explicado en clase, no se esperaba** que nadie respondiera con dichas alternativas, se indican **sólo como curiosidad**.

	Opción 1	Opción 2	Opción 3	Opción 4
<code>return x + x;</code>	<code>leaq (%rdi,%rdi), %rax</code>	<code>mov %rdi, %rax</code> <code>add %rdi, %rax</code>	<code>mov %rdi, %rax</code> <code>shl %rax</code>	
<code>return x * 5;</code>	<code>leaq (%rdi,%rdi,4), %rax</code>	<code>mov %rdi, %rax</code> <code>shl \$2, %rax</code> <code>add %rdi, %rax</code>		
<code>return x * 8;</code>	<code>leaq 0(%rdi,8), %rax</code>	<code>mov %rdi, %rax</code> <code>shl \$3, %rax</code>		
<code>return x * 16;</code>	<code>movq %rdi, %rax</code> <code>salq \$4, %rax</code>	<code>mov %rdi, %rax</code> <code>shl \$4, %rax</code>	<code>lea (,%rdi,8),%rax</code> <code>add %rax, %rax</code>	<code>lea (,%rdi,8),%rax</code> <code>shl %rax</code>
<code>return x / 16;</code>	<code>movq %rdi, %rax</code> <code>shrq \$4, %rax</code>	<code>xor %rax, %rax</code> <code>shld \$-4,%rdi,%rax</code>		
<code>return x * 12;</code>	<code>leaq (%rdi,%rdi,2), %rdx</code> <code>leaq 0(%rdx,4), %rax</code>	<code>lea (...2), %rax</code> <code>shl \$2, %rax</code>	<code>lea (...2), %rax</code> <code>add %rax, %rax</code> <code>add %rax, %rax</code>	<code>lea (...4), %rax</code> <code>add %rdi, %rax</code> <code>add %rax, %rax</code>
<code>return x % 8;</code>	<code>movq %rdi, %rax</code> <code>andl \$7, %eax</code>	<code>mov %rdi, %rax</code> <code>ror \$3, %rax</code> <code>shr \$61, %rax</code>	<code>mov %rdi, %rax</code> <code>ror \$3, %rax</code> <code>shr \$-3, %rax</code>	<code>xor %rax, %rax</code> <code>shrd \$3,%rdi, %rax</code> <code>rol \$3, %rax</code>
<code>return x &amp; 0xffffffff;</code>	<code>movl %edi, %eax</code>	<code>mov %rdi, %rax</code> <code>and \$-1, %eax</code>	<code>mov %rdi, %rax</code> <code>and \$0xffffffff, %eax</code>	
<code>return x &amp; 1;</code>	<code>movq %rdi, %rax</code> <code>andl \$1, %eax</code>	<code>mov %rdi, %rax</code> <code>and \$1, %rax</code>	<code>xor %rax, %rax</code> <code>rcr %rdi</code> <code>rcl %rax</code>	
<code>return x != 0;</code>	<code>testq %rdi, %rdi</code> <code>setne %al</code> <code>movzbl %al, %eax</code>	<code>xor %rax, %rax</code> <code>test %rdi, %rdi</code> <code>setnz %al</code>		

#### 2. Unidad de Control (0.4 puntos).

Se puntuá **0.1p** por cada dibujo y **0.1p** por el valor final correcto. (total **0.1 x 3 + 0.1 x 1 = 0.4p**).

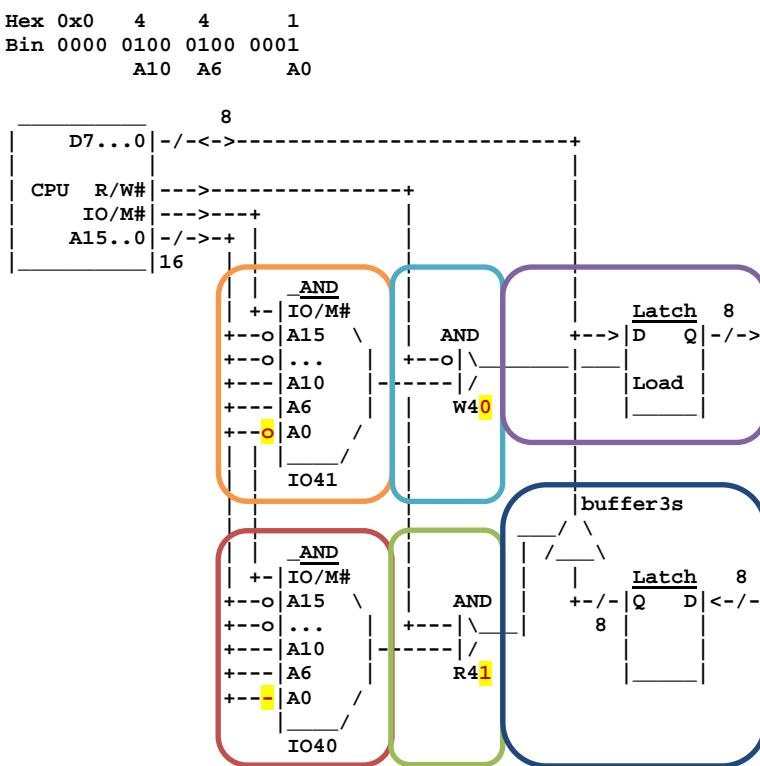
Ejemplo tomado de [https://images.slideplayer.com/11/3265187/slides/slide\\_26.jpg](https://images.slideplayer.com/11/3265187/slides/slide_26.jpg)  
 Ver dibujo al final de la plantilla

$$2048 * 41 - (2048 * 7 + 100 * 41) = 83968 - 18436 = \textcolor{red}{65532}$$

### 3. Entrada/Salida (0.6 puntos).

Se puntuá **0.1p** por cada zona del dibujo. (total  **$0.1 \times 3 + 0.1 \times 3 = 0.6p$** ).

Como el Problema 4 de Septiembre 2015, pero intercambiando direcciones puertos. Modificaciones **resaltadas**. Solución copiada de aquella plantilla, intercambiando el bit menos significativo



### 4. Mapa de memoria (0.6 puntos).

Se puntuá **0.1p** por el valor en a) y **0.05p** por cada dirección en b) (total  **$0.1 + 0.05 \times 10 = 0.60p$** ).

Ejemplo tomado de <http://infocenter.arm.com/help/index.jsp?topic=/com.arm.doc.ddi0201d/I21752.html>

$$4 \text{ M} + 4 \text{ M} + 248 \text{ M} = 256 \text{ M}$$

$$4096 \text{ M} / 256 \text{ M} = 2^{32}/2^{28}=2^4=16; n = 16 - 1 = 15$$

0x0000 0000	Zona de 4MB
0x003F FFFF	
0x0040 0000	Zona de 4MB
0x007F FFFF	
0x0080 0000	Zona de 248MB
0xFFFF FFFF	
0x1000 0000	Zona 1 de 256MB
0xFFFF FFFF	
...	
0xF000 0000	Zona <b>n=15</b> de 256MB
0xFFFF FFFF	

### 5. Memoria cache (0.4 puntos).

Se puntuá **0.1p** por cada número, y **0.1p** por los 3 nombres de los campo (total  **$0.1 \times 3 + 0.1 = 0.4p$** ).

$$\text{MP: } 4 \text{ GB} = 2^{32} \text{ B}$$

$$\text{Cache: } 4 \text{ KB} = 2^{12} \text{ B}, 32 \text{ B/línea} = 2^5 \text{ B/línea}, 4 \text{ vías} = 2^2 \text{ líneas/conjunto}$$

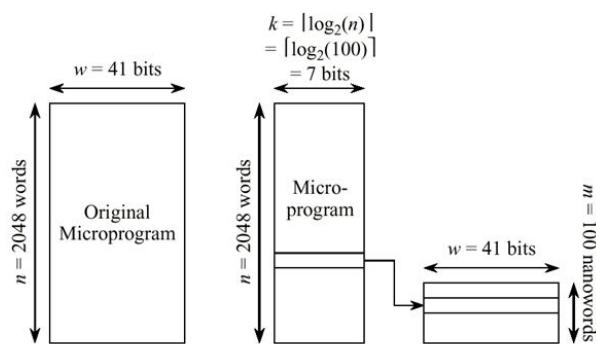
$$\text{Cache: } 2^{12} \text{ B} / 2^5 \text{ B/línea} = 2^7 \text{ líneas}, 2^7 \text{ líneas} / 2^2 \text{ líneas/conjunto} = 2^5 \text{ conjuntos}$$

Dirección física de memoria principal desde el punto de vista de cache:

etiqueta (22)	conjunto (5)	byte (5)

# Microprogramming vs. Nanoprogramming

- (a) Microprogramming vs.
- (b) nano-programming.



Total Area =  $n \times w = 2048 \times 41 = 83,968 \text{ bits}$

Microprogram Area =  $n \times k = 2048 \times 7 = 14,336 \text{ bits}$   
 Nanoprog Area =  $m \times w = 100 \times 41 = 4100 \text{ bits}$   
 Total Area =  $14,336 + 4100 = 18,436 \text{ bits}$

(a)

(b)

**Nombre:**
**DNI:**
**Grupo:**

### Test de Teoría (3.0p)

Todas las preguntas son de elección simple sobre 4 alternativas.

Cada respuesta vale 0.1p si es correcta, 0p si está en blanco o claramente tachada, -0.03p si es errónea.  
 Anotar las respuestas (a, b, c ó d) en la siguiente tabla.

1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30

1. En el direccionamiento indirecto a través de registro, la dirección efectiva...

- a. se encuentra en una dirección de memoria
- b. se encuentra en un registro general del procesador
- c. se encuentra en el registro de instrucción
- d. se calcula como la suma del contenido de dos registros

2. Si N es el número de instrucciones máquina de un programa, F es la frecuencia de reloj, y C el número promedio de ciclos por instrucción, el tiempo de ejecución del programa será:

- a.  $N \cdot F/C$
- b.  $N \cdot C/F$
- c.  $N \cdot F \cdot C$
- d.  $N/(C \cdot F)$

3. ¿Cuál/es de los siguientes tipos de datos tienen distinto tamaño en arquitecturas x86 y x86-64 (Intel 32 bits y 64 bits)?

- a. long
- b. char \*
- c. ninguno de los dos
- d. ambos

4. La secuencia de instrucciones

```
leaq (%rdi,%rdi,2), %rax
salq $3, %rax
```

produce el efecto de...

- a. leer en RAX a partir de la posición  $2 * \text{RDI} + 2$  y multiplicar dicho contenido por 8

b. calcular  $\text{RAX} = (2 * \text{RDI} + 2) * 3 = 6 * (\text{RDI} + 1)$

c. calcular  $\text{RAX} = 24 * \text{RDI}$

d. calcular  $\text{RAX} = 9 * \text{RDI}$

5. ¿Qué valor contendrá el registro RDX tras ejecutar las dos instrucciones siguientes?

```
movq $-1, %rdx
movb $1, %dl
```

- a. 0xFFFF FFFF FFFF FFF1
- b. 0xFFFF FFFF FFFF FF01
- c. 0xFFFF FFFF FFFF 0001
- d. 0x0000 0000 0000 0001

6. En la secuencia de programa siguiente:

```
400544: e8 07 00 00 00 callq 400550 <f>
400549: 48 89 03          mov %rax,(%rbx)
```

¿cuál es el valor que introduce en la pila la instrucción call?

- a. 0x400544
- b. 0x400549
- c. 0x40054b
- d. 0x400550

7. Una función C declarada como **int get\_var\_digit(size\_t index, size\_t digit)** genera como código ensamblador

```
leaq (%rdi,%rdi,4), %rax
addq %rax, %rsi
movl var(,%rsi,4), %eax
ret
```

Se puede adivinar que:

- a. var es un array multi-nivel (punteros a enteros) de cuatro filas

- b. var es un array bidimensional de enteros, con cinco columnas
- c. var es un array multi-nivel pero no se pueden adivinar las dimensiones
- d. var es un array bidimensional de enteros, no se pueden adivinar dimensiones
- 

8. El microcódigo de un procesador consiste en 640 microinstrucciones de 70 bits, de las cuales 280 son únicas. ¿Cuántos bits se podría llegar a ahorrar usando nanoprogramación?

- a.  $640 \times 70 - 280 \times 70$
- b.  $640 \times 70 - 640 \times 9 - 280 \times 70$
- c.  $640 \times 70 - (640 \times \log_2(70) + 280 \times 70)$
- d.  $640 \times 280 - 640 \times 70 - 280 \times 70$
- 

9. ¿A qué instrucción de lenguaje máquina podría corresponder la siguiente secuencia de microinstrucciones del camino de datos con un bus estudiado en clase?

*Enable R1, Load Y  
Enable R2, Select Y, Add, Load Z  
Enable Z, Load R2*

- a. load (R1,R2), R2
- b. add R1, R2
- c. store R1+R2, (R2)
- d. move Y(R1,R2), Z(R2)
- 

10. Suponer que un procesador ideal que ejecuta cada instrucción en T segundos se segmenta en cuatro etapas ideales de duración T/4. Con ello se consigue que

- a. 4 instrucciones se ejecuten en 4T segundos
- b. una instrucción se ejecute en T/4 segundos
- c. cada 4T segundos se terminen de ejecutar 4 instrucciones
- d. cada T/4 segundos se termine de ejecutar una instrucción
- 

11. Precaptar instrucciones antes de que sean necesarias y almacenarlas en una cola de instrucciones, es una técnica que se usa para...

- a. evitar cierto tipo de riesgos estructurales
- b. reducir riesgos por dependencias de datos
- c. corregir algunos riesgos de control
- d. calcular las predicciones de saltos
- 

12. La técnica de Consulta de Estado (polling) puede usarse para... (señalar la opción INCORRECTA)

- a. identificar el origen de una interrupción
- b. consultar si el dispositivo está dispuesto para entregar o recibir datos
- c. consultar el sentido del DMA en curso (desde memoria / hacia memoria)
- d. establecer un mecanismo software de asignación de prioridades a los dispositivos
- 

13. Técnicas que suelen contemplar los procesadores para gestionar el sistema de prioridades entre peticiones de interrupción (señalar la opción INCORRECTA)

- a. gestión de prioridades centralizada
- b. gestión de prioridades distribuida
- c. gestión de prioridades simultánea
- d. gestión de prioridades híbrida
- 

14. Una SRAM de 256Kx4bit (1Mbit) puede venir organizada en 1024 filas, dedicando por tanto al decodificador de columnas...

- a. 6 bits
- b. 7 bits
- c. 8 bits
- d. 10 bits
- 

15. ¿Cuál de los siguientes grupos de señales **no** se usa en un chip de memoria SRAM?

- a. Selección de chip CS# y habilitación de escritura WE#
- b. Selección de filas RAS# y de columnas CAS#
- c. Direcciones A<sub>n-1</sub>-A<sub>0</sub>
- d. Datos D<sub>n-1</sub>-D<sub>0</sub>
- 

16. ¿En qué pareja de registros están la dirección de memoria y el dato que se leerá o escribirá en memoria?

- a. pc y mar
- b. ir y pc
- c. mar y mdr/mbr
- d. mdr/mbr y pc
- 

17. Si queremos almacenar la palabra de 16 bits 0x8965 en memoria según little-endian, quedará almacenada a partir de la posición 0x1000 como:

- a. en el byte 0x1000 se guarda 0xA6 y en el 0x1001 0x91
- b. en el byte 0x1000 se guarda 0x89 y en el 0x1001 0x65

- c. en el byte 0x1000 se guarda 0x91 y en el 0x1001 0xA6
- d. en el byte 0x1000 se guarda 0x65 y en el 0x1001 0x89
- 

18. En x86-64, el registro contador de programa se denomina:

- a. rip
- b. eip
- c. pcr
- d. pc
- 

19. Si rax contiene x, ¿cuál de las siguientes instrucciones calcula x\*9?

- a. leaq 8(%rax,%rax),%rdx
- b. leaq (%rax,%rax,8),%rdx
- c. leaq 3(%rax,%rax,2),%rdx
- d. leaq 5(%rax,%rax,4),%rdx
- 

20. La instrucción jbe / jna provoca un salto si...

- a. SF == 1 || ZF == 1
- b. CF == 1 || ZF == 1
- c. CF == 1
- d. SF != OF
- 

21. En x86-64, es responsabilidad del procedimiento llamado (callee) salvaguardar, entre otros, los registros:

- a. %rbx, %rsi, %rdi
- b. %rax, %rdx, %rcx
- c. %rax, %rbx, %rcx, %rdx
- d. %rbx, %rbp
- 

22. En x86-64, una función con 10 parámetros de tipo long que devuelve el valor del 8º parámetro y no modifica el puntero de pila puede traducirse a ensamblador como:

- a. movq 8(%rsp), %rax  
ret
- b. movq 16(%rsp), %rax  
ret
- c. movq %r10, %rax  
ret
- d. movq %r8, %rax  
ret
- 

23. ¿Cuál de las siguientes características es típica de la microprogramación horizontal?

- a. Muchos campos solapados
- b. Poca codificación
- c. Microinstrucciones cortas

- d. Poca capacidad para expresar paralelismo entre microoperaciones
- 

24. En un camino de datos con un solo bus, para realizar la operación de copia de un registro r1 en un registro r2, es decir  $r2 \leftarrow r1$ , es necesario:

- a. Activar la carga del registro r1 y habilitar la salida triestado del registro r2
- b. Habilitar la salida triestado del registro r1 y activar la carga del registro r2
- c. Habilitar las salidas triestado de los registros r1 y r2 y activar la carga del registro r2
- d. Habilitar la salida triestado del registro r2 y activar la carga de los registros r1 y r2
- 

25. Sobre la segmentación:

- a. La frecuencia de reloj viene impuesta por la etapa más corta.
- b. Existen limitaciones al rendimiento provocadas por las instrucciones de salto y por las dependencias de datos.
- c. Es una técnica para lanzar a ejecutar simultáneamente varias instrucciones con el fin de reducir el tiempo de ejecución.
- d. Un procesador superescalar no puede estar segmentado.
- 

26. La ganancia de velocidad ideal en un cauce de K etapas de igual duración T ejecutando un programa de N instrucciones es:

- a.  $S=(K \cdot N)/(K-N+1)$
- b.  $S=(N \cdot K \cdot T)/((N-K+1) \cdot T)$
- c.  $S=(K \cdot N)/(K+N-1)$
- d.  $S=(N \cdot T)/((N+K-1) \cdot T)$
- 

27. ¿Cuál de las siguientes tareas NO es responsabilidad de un circuito de interfaz o controlador de periféricos sencillo?

- a. Adaptar el formato de las señales.
- b. Ajustar la temporización entre el procesador y los dispositivos de E/S.
- c. Recibir señales de control desde el procesador.
- d. Ejecutar el programa de transferencia de información entre el procesador y los dispositivos de E/S.
- 

28. ¿Con cuál de los siguientes dispositivos tendría sentido utilizar E/S programada sin consulta de estado?

- a. Salida a un display de 7 segmentos
  - b. Entrada desde un disco duro
  - c. Salida a una impresora
  - d. Con ningún dispositivo tiene sentido
- 

29. ¿A qué tipo de localidad de memoria hace referencia la siguiente afirmación: "si se referencia un elemento, los elementos cercanos a él serán referenciados pronto"?

- a. Localidad espacial
  - b. Localidad secuencial
  - c. Localidad temporal
  - d. Localidad asociativa
- 

30. ¿Cuál de las siguientes afirmaciones acerca de la memoria es FALSA?

- a. La memoria dinámica usa señales de control RAS# y CAS#.
  - b. Las celdas de memoria dinámica están constituidas por un transistor y un condensador.
  - c. Las celdas de memoria estática tienen que ser constantemente refrescadas.
  - d. La memoria estática se emplea en las caches L1 y L2.
-

**Nombre:**
**DNI:**
**Grupo:**

## Test de Prácticas (4.0p)

**Todas las preguntas son de elección simple sobre 4 alternativas.**

**Cada respuesta vale 0.2p si es correcta, 0 si está en blanco o claramente tachada, -0.06p si es errónea.**

**Anotar las respuestas (a, b, c ó d) en la siguiente tabla.**

1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20

1. En la práctica "media" se pide como ejercicio previo sumar una lista de N enteros en doble precisión. Alguna de las siguientes técnicas no está relacionada con la aritmética en doble precisión:

- a. acumulación de desbordamientos (OF, overflow flag)
- b. acumulación de acarreos (CF, carry flag)
- c. extensión con ceros
- d. extensión de signo

2. En la misma práctica "media", el programa esqueleto ofrecido (suma.s) no es válido, en concreto... (marcar la opción FALSA)

- a. no está preparado para sumar más de 9 elementos
- b. no hace extensión con ceros de los elementos
- c. no hace extensión de signo de los elementos
- d. no consulta ni el flag de acarreo CF ni el de overflow OF

3. La práctica "popcount" debía calcular la suma de bits (peso Hamming) de los elementos de un array. Un estudiante entrega la siguiente versión de `popcount5`:

```
int popcount5(int* array, size_t len){
    size_t i,j;
    int x, val, result=0;
    for (i=0; i<len; i++){
        x = array[i];
        val = 0;
        for (j=0; j<8; j++){
            val += x & 0x01010101;
            x >>= 1;
        }
        val += (val >> 16);
        val += (val >> 8);
        result+= val & 0xFF;
    }
}
```

```
    }
```

Esta función se diferencia de la versión "oficial" en los tipos de array y x.

Esta función `popcount5`:

- a. produce siempre el resultado correcto
- b. fallaría con array={0,1,2,3}
- c. fallaría con array={0,-1,-2,-3}
- d. no es correcta pero el error no se manifiesta en los ejemplos propuestos, o se manifiesta en ambos

4. Otro estudiante entrega la siguiente versión de `popcount5`:

```
int pcnt5(unsigned* array, size_t len){
    size_t i,j;
    unsigned x;
    int val, result=0;

    for (i=0; i<len; i++){
        x = array[i];
        val = 0;
        for (j=0; j<=8; j++){
            val += x & 0x01010101;
            x >>= 1;
        }
        val += (val >> 16);
        val += (val >> 8);
        result+= val & 0xFF;
    }
    return result;
}
```

Esta función sólo se diferencia de la versión "oficial" recomendada en clase en las condiciones del for interno.

Esta función `popcount5` fallaría:

- a. con array={1, 16, 256, 4096}
- b. con array={1, 32, 1024, 32768}
- c. con ambos ejemplos
- d. con ninguno de los dos ejemplos

5. En una bomba como las estudiadas en prácticas, del tipo...

```
4006fa: lea    0x10(%rsp),%rbx
4006ff: mov    0x20096a(%rip),%rdx
400706: mov    $0x64,%esi
40070b: mov    %rbx,%rdi
40070e: callq  400570 <fgets@plt>
400713: mov    $0xd,%edx
400718: lea    0x200939(%rip),%rsi
40071f: mov    %rbx,%rdi
400722: callq  400560 <strncmp@plt>
400727: test   %eax,%eax
400729: je     400730 <main+0x51>
40072b: callq  400697 <boom>
400730: lea    0x1b5(%rip),%rsi
```

...la contraseña (alfanumérica) es...

- a. el string almacenado a partir de 0x10(%rsp)
- b. el string alm. a partir de 0x20096a+0x4006ff
- c. el string almacenado a partir de 0x601058
- d. el string alm. a partir de 0x200939+0x400718

6. En una bomba como las estudiadas en prácticas, del tipo...

```
400746: lea    0xc(%rsp),%rsi
40074b: lea    0x1ae(%rip),%rdi
400752: mov    $0x0,%eax
400757: callq  400590 <_scanf@plt>
40075c: mov    0x2008ee(%rip),%eax
400762: cmp    %eax,0xc(%rsp)
400766: je     40076d <main+0x8e>
400768: callq  400697 <boom>
40076d: callq  4006bb <defused>
```

...el código numérico (pin) es...

- a. el entero almacenado a partir de 0xc(%rsp)
- b. el entero alm. a partir de 0x1ae+0x40074b
- c. el entero almacenado a partir de 0x6010560
- d. el entero alm. a partir de 0x2008ee+0x400762

7. La función `setup()` de Arduino es llamada:

- a. Al principio de cada iteración de la función `loop()`
- b. Cuando se sube desde el entorno de desarrollo o se pulsa el botón de reset, pero no cuando se conecta la alimentación

c. Cuando se conecta la alimentación a la placa, se pulsa el botón de reset, o se sube desde el entorno de desarrollo

d. Cuando se conecta la alimentación a la placa, se pulsa el botón de reset, pero no cuando se sube desde el entorno de desarrollo

8. Suponga una memoria cache con las siguientes propiedades: Tamaño: 512 bytes. Política de reemplazo: LRU. Estado inicial: vacía (todas las líneas inválidas). Suponga que para la siguiente secuencia de direcciones enviadas a la cache: 0, 1, 2, 4, 8, 16, 32, 64, la tasa de acierto es 0,25. ¿Cuál es el tamaño de línea de la cache?

- a. 4 bytes
- b. 8 bytes
- c. 16 bytes
- d. Ninguno de los anteriores

9. En el programa `size.cc` de la práctica "cache", se accede al vector saltando...

- a. de byte en byte
- b. de 64 en 64 bytes
- c. de 1 KB en 1 KB
- d. de line en line bytes, donde line barre los valores desde 1 hasta 1K en un bucle for

10. En la práctica de cache hemos hecho una gráfica con el código `size.cc` ¿Qué forma tiene la gráfica que se debe obtener?

- a. Forma de U (o V), con un tramo descendente y otro ascendente
- b. Forma de /, una gráfica siempre creciente y sin escalones
- c. Forma de media U seguida de \, es decir, un tramo descendente suave, un pequeño tramo horizontal, y un tramo descendente lineal
- d. Una escalera con varios tramos horizontales

11. La línea de código ensamblador: **mov \$msg, %rsi**

- a. Copia en rsi los primeros 64 bits de memoria desde la posición apuntada por la etiqueta msg.
- b. Copia en rsi todo el contenido de la cadena apuntada por msg.
- c. Copia en rsi la dirección de memoria de 64 bits almacenada en memoria a partir de la posición indicada por la etiqueta msg.
- d. Copia en rsi los 64 bits de la dirección msg.

**12.** En la práctica "media" se programa la suma de una lista de 16 enteros de 4 bytes para producir un resultado de 8 bytes, primero sin signo y luego con signo. Si la lista se rellena con el valor 0x0400 0000, ¿en qué se diferencian los resultados de ambos programas?

- a. no se diferencian
  - b. en uno ocupa 32 bits, en otro 64 bits
  - c. en uno se interpreta como negativo, en otro como positivo
  - d. en uno los 32 bits superiores son 0xFFFF FFFF, en el otro no
- 

**13.** ¿Cuál de las siguientes líneas declara un puntero a función en C?

- a. int \*func;
  - b. int func();
  - c. int \*func();
  - d. int (\*func)();
- 

**14.** Si val es una variable de tipo unsigned long, entonces la sentencia: **val += (val >> 32);**

- a. Pone siempre val a 0.
  - b. Deja siempre val al mismo valor que tuviera antes de la sentencia.
  - c. Su traducción incluye una instrucción shr seguida de una suma.
  - d. Su traducción incluye una instrucción sar seguida de una suma.
- 

**15.** ¿Para qué se utiliza la función gettimeofday en la práctica de la "bomba digital"?

- a. Para cronometrar y poder comparar las duraciones de las distintas soluciones del programa.
  - b. Para imprimir la hora en la pantalla.
  - c. Para cifrar la contraseña en función de la hora actual.
  - d. Para lanzar un error cuando el usuario tarde demasiado tiempo en introducir la contraseña o el PIN.
- 

**16.** En la práctica de la bomba, el primer ejercicio consiste en ir saltando las “explosiones” mientras se depura el código, para lo cual se puede utilizar...

- a. objdump o gdb
- b. gdb o ddd
- c. ddd o hexedit

d. hexedit u objdump

---

**17.** En la placa del kit de Arduino, las patillas de tierra vienen etiquetadas con la leyenda:

- a. A0
  - b. 5V
  - c. GND
  - d. 3.3V
- 

**18.** Las resistencias utilizadas en la práctica de Arduino

- a. Son de color azul claro y tienen 5 bandas de color: las 3 primeras indican un valor, la 4<sup>a</sup> banda es un multiplicador y la 5<sup>a</sup> banda es la tolerancia
  - b. Son de color beige y tienen 4 bandas de color: las 2 primeras indican un valor, la 3<sup>a</sup> banda es un multiplicador y la 4<sup>a</sup> banda es la tolerancia
  - c. Tienen polaridad y el cátodo (polo negativo) es el extremo de la banda de color con una separación mayor respecto a las otras.
  - d. Tienen polaridad y el ánodo (polo positivo) es el extremo de la banda de color con una separación mayor respecto a las otras.
- 

**19.** ¿Cuál de las siguientes afirmaciones sobre las caches es **FALSA**?

- a. Casi ningún procesador actual tiene memoria cache L2.
  - b. Las direcciones a las que accede un programa no son completamente aleatorias, sino que se rigen por ciertos patrones de localidad.
  - c. Un procesador actual tiene varias caches de nivel 1.
  - d. La cache de nivel 3 no contiene toda la memoria que maneja el programa.
- 

**20.** En el programa line.cc, si para cada tamaño de línea (line) recorremos una única vez el vector, la gráfica resultante es decreciente porque:

- a. Cada vez que line aumenta al doble, el número de aciertos por localidad temporal aumenta, porque ya habíamos accedido a cada posición i del vector cuando lo recorrimos en el punto anterior del eje X.
- b. Cada vez que line aumenta al doble, el número de aciertos por localidad espacial aumenta, porque ya habíamos accedido a cada posición i-1 del vector cuando lo recorrimos en el punto anterior del eje X.

- c. Cada vez que line aumenta al doble, se accede con éxito a más posiciones del vector en niveles de la jerarquía de memoria más rápidos.
  - d. Cada vez que line aumenta al doble, realizamos la mitad de accesos al vector que para el valor anterior.
-

Nombre:	
DNI:	Grupo:

## Examen de Problemas (3,0 p)

- 1. Ensamblador.** (0.8 puntos). La siguiente es una función genérica para intercambiar los bytes de una variable numérica cuyo tipo se especifica mediante typedef. En este ejemplo concreto number\_t es un alias del tipo long (8 bytes en x86-64).

```
#include <stddef.h>

typedef long number_t;
#define NUMBER_SIZE sizeof(number_t)

number_t big2little (number_t n) {
    union {
        number_t n;
        char b[NUMBER_SIZE];
    } src, dst;

    src.n = n;
    for (size_t i = 0; i < NUMBER_SIZE; i++)
        dst.b[i] = src.b[NUMBER_SIZE-1 - i];

    return dst.n;
}
```

Escriba el código de una función en ensamblador de x86-64 que realice la misma operación (devolver el número de 8 bytes pasado como argumento con los bytes intercambiados).

Ayuda:

1. *size\_t* es equivalente a *unsigned long*
2. los dos campos *n* y *b[NUMBER\_SIZE]* de la unión están solapados en memoria (ocupan el mismo espacio)
3. *sizeof(src)* es 8, *sizeof(dst)* es 8
4. para almacenar *src* y *dst* se puede usar la zona roja por debajo de *rsp* sin tener que cambiar el valor de *rsp*, es decir, que se puede acceder directamente a *-8(%rsp)*, *-16(%rsp)*, etc.

**2. Ensamblador** (0.2 puntos). Una función devuelve el producto escalar entre dos vectores, definido como:

$$A \cdot B = A_1B_1 + A_2B_2 + \dots + A_nB_n$$

Al compilar dicha función se obtiene el siguiente código en ensamblador:

```
producto_escalar:  
    testq    %rdx, %rdx  
    je       .L4  
    xorl    %eax, %eax  
    xorl    %r8d, %r8d  
.L3:  
    movq    (%rdi,%rax,8), %rcx  
    imulq   (%rsi,%rax,8), %rcx  
    addq    $1, %rax  
    addq    %rcx, %r8  
    cmpq    %rax, %rdx  
    jne     .L3  
    movq    %r8, %rax  
    ret  
.L4:  
    xorl    %r8d, %r8d  
    movq    %r8, %rax  
    ret
```

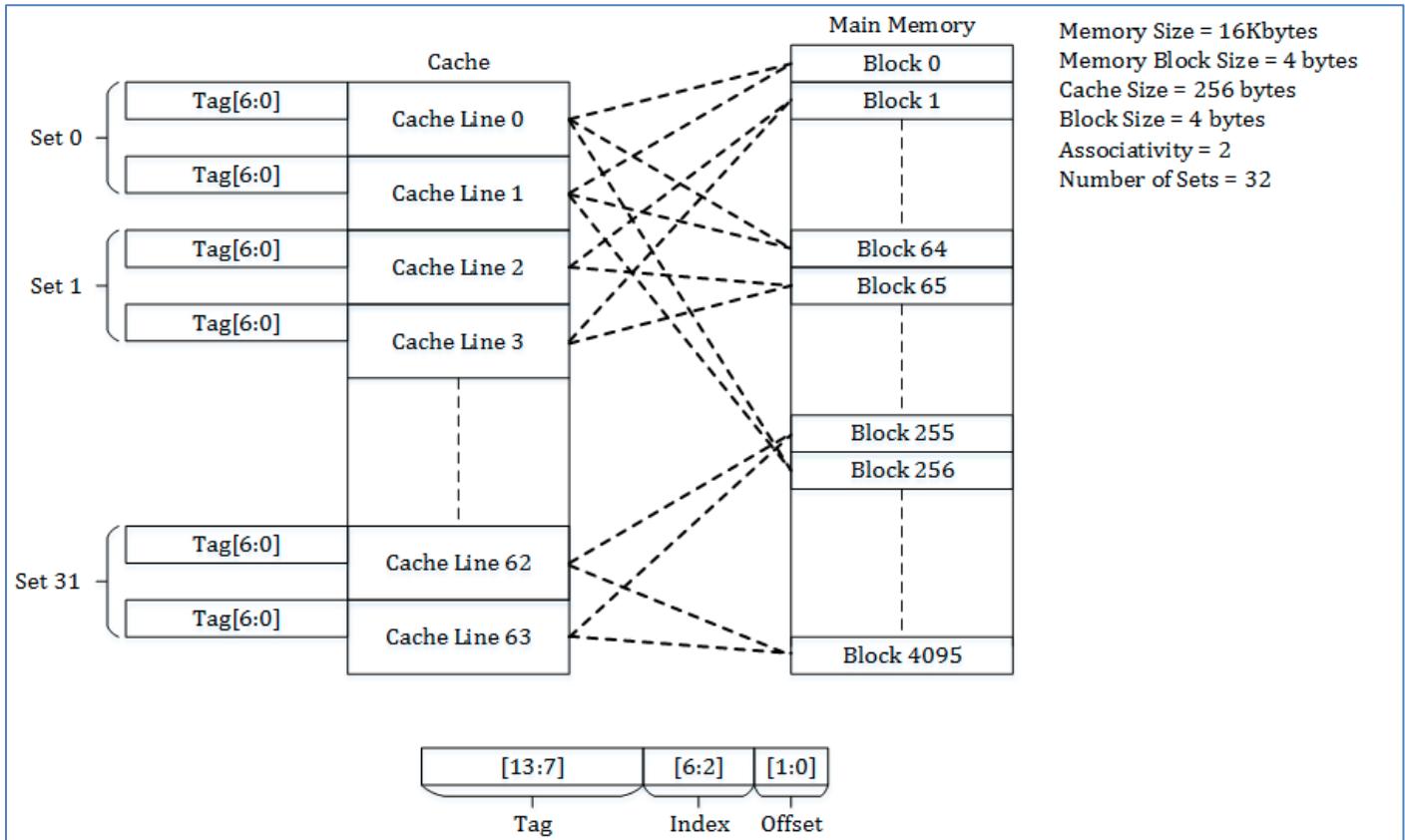
- a) (0.1p) ¿De qué tipo es cada uno de los elementos de los vectores A y B?
- b) (0.1p) ¿De cuántas dimensiones es el espacio vectorial, es decir, cuántos elementos tiene cada uno de los vectores A y B?

**3. Unidad de control** (0.5 puntos). Dibuje un camino de datos de un único bus para una arquitectura de registros de propósito general con dos operandos para las instrucciones del tipo suma, resta, and, etc.

**4. Entrada/Salida** (0.5 puntos). Disponemos de un microprocesador de 8 bits (bus de datos de 8 bits y bus de direcciones de 16 bits) con E/S independiente. Diseñe un sistema de E/S que permita acceder a los siguientes puertos: puerto 0x240 de entrada y 0x241 de salida. Utilice lógica de decodificación distribuida. No emplee decodificadores.

**5. Configuración de memoria** (0.5 puntos). Un módulo de memoria DIMM tiene un tamaño de 8 GB en configuración de  $1G \times 64$  y está constituido por módulos de memoria DRAM de  $1G \times 4$ . El módulo DIMM tiene, entre otras líneas, las siguientes: A14-A0, D63-D0, RAS#, CAS# y WE#. Dibuje un esquema del módulo DIMM incluyendo cada uno de sus chips y las conexiones de patillas.

**6. Cache** (0.5 puntos). El siguiente esquema de Wikimedia Commons muestra la estructura general de una cache asociativa por conjuntos:



[https://commons.wikimedia.org/wiki/File:Set-Associative\\_Cache\\_Snehal\\_Img.png](https://commons.wikimedia.org/wiki/File:Set-Associative_Cache_Snehal_Img.png)

Dibuje un esquema similar al anterior, pero con los números correctos para la cache L3 del procesador Intel Core i7-10710U, que reúne las siguientes características y usa un tamaño de línea (= bloque) de 64 bytes:

Level 1 cache size ?	6 x 32 KB 8-way set associative instruction caches 6 x 32 KB 8-way set associative data caches
Level 2 cache size ?	<b>6 x 256 KB 4-way set associative caches</b>
Level 3 cache size	<b>12 MB 12-way set associative shared cache</b>
Physical memory	64 GB

### Test de Teoría (3.0p)

<b>1</b>	<b>2</b>	<b>3</b>	<b>4</b>	<b>5</b>	<b>6</b>	<b>7</b>	<b>8</b>	<b>9</b>	<b>10</b>	<b>11</b>	<b>12</b>	<b>13</b>	<b>14</b>	<b>15</b>	<b>16</b>	<b>17</b>	<b>18</b>	<b>19</b>	<b>20</b>	<b>21</b>	<b>22</b>	<b>23</b>	<b>24</b>	<b>25</b>	<b>26</b>	<b>27</b>	<b>28</b>	<b>29</b>	<b>30</b>
b	b	d	c	b	b	b	b	b	d	a	c	c	c	b	c	d	a	b	b	d	b	b	b	b	c	d	a	a	c

### Test de Prácticas (4.0p)

<b>1</b>	<b>2</b>	<b>3</b>	<b>4</b>	<b>5</b>	<b>6</b>	<b>7</b>	<b>8</b>	<b>9</b>	<b>10</b>	<b>11</b>	<b>12</b>	<b>13</b>	<b>14</b>	<b>15</b>	<b>16</b>	<b>17</b>	<b>18</b>	<b>19</b>	<b>20</b>
a	a	a	a	c	d	c	a	b	d	d	a	d	c	d	b	c	a	a	d

### Examen de Problemas (3.0p)

#### 1. Ensamblador (0.8 puntos).

Nota completa **0.80p** si el resultado es correcto. Las soluciones ocupan unas 15 líneas incluyendo etiquetas, por eso en modo “rescate” se puntúa **0.05p** por cada línea “acertada” (total **0.05 x 15 = 0.75p**).

Algunas alternativas posibles:

<p><b>Solución de gcc 9.2 con -Og:</b></p> <pre>big2little:     movq %rdi, -8(%rsp)          ; src.n = n;     movl \$0, %eax                ; for (i=0; ...     jmp .L2                      ; jump-to-middle  .L3:     movl \$7, %edx                ; NUM_SIZ-1 ...     subq %rax, %rdx              ; ... - i     movzbl -8(%rsp,%rdx), %edx  ; src.b[ N-1-i]     movb %dl, -16(%rsp,%rax)   ; dst.b[i] = ...     addq \$1, %rax                ; for (... i++)     cmpq \$7, %rax                ; for (... i&lt;8; ...     jbe .L3                      ; unsigned i     movq -16(%rsp), %rax        ; return dst.n;     ret</pre>	<p><b>Solución de gcc 9.2 con -Os:</b></p> <pre>big2little:     movq %rdi, -16(%rsp)     xorl %eax, %eax     leaq -9(%rsp), %rcx  .L2:     movq %rcx, %rdx     subq %rax, %rdx     movb (%rdx), %dl     movb %dl, -8(%rsp,%rax)     incq %rax     cmpq \$8, %rax     jne .L2     movq -8(%rsp), %rax     ret</pre>	
<p><b>Solución de gcc 9.2 con -O1:</b></p> <pre>big2little:     movq %rdi, -8(%rsp)     leaq -1(%rsp), %rax     leaq -16(%rsp), %rdx     leaq -8(%rsp), %rsi  .L2:     movzbl (%rax), %ecx     movb %cl, (%rdx)     movq %rax, %rcx     subq \$1, %rax     addq \$1, %rdx     cmpq %rsi, %rcx     jne .L2     movq -16(%rsp), %rax     ret</pre>	<p><b>Solución de gcc 9.2 con -O2:</b></p> <pre>big2little:     movq %rdi, -16(%rsp)     leaq -16(%rsp), %rsi     leaq -9(%rsp), %rax     leaq -8(%rsp), %rdx  .L2:     movzbl (%rax), %ecx     addq \$1, %rdx     movb %cl, -1(%rdx)     movq %rax, %rcx     subq \$1, %rax     cmpq %rsi, %rcx     jne .L2     movq -8(%rsp), %rax     ret</pre>	<p><b>Solución de Tacho:</b></p> <pre>big2little:     mov \$8, %ecx loop:     shl \$8, %rax     movb %dl, %al     shr \$8, %rdi #    loop loop     dec %ecx     jnz loop     ret</pre>
<p><b>Solución de gcc 9.2 con -O3:</b></p> <pre>big2little:     movq %rdi, %rax     bswap %rax     ret</pre>	<p>↔ (no se esperaba que nadie conociera bswap)</p>	<p>↑ (no se esperaba que nadie conociera loop)  Versión más eficiente, no usa memoria  Solución encontrada por Ahmed Brek y Atanasio Rubio</p>

## 2. Ensamblador(0.2 puntos).

Se puntuá **0.1p** por cada apartado (total  **$0.1 \times 2 = 0.20p$** ),

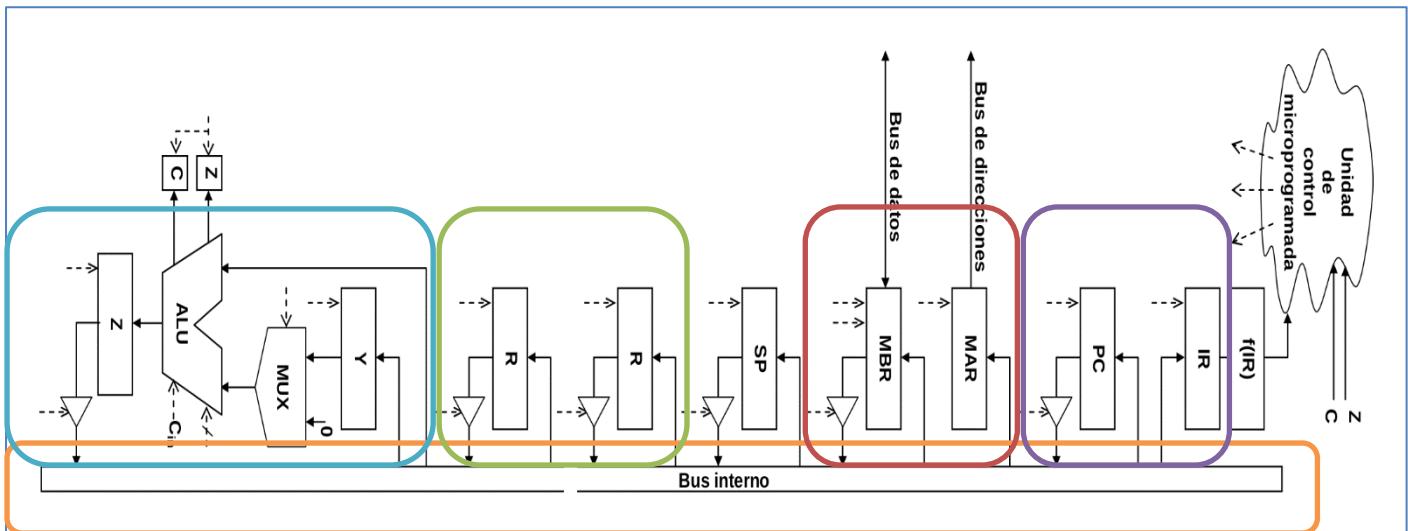
a) signed long, por imulq y el factor de escala 8

b) viene especificado por el registro rdx, es decir, se pasa como tercer parámetro de la función

## 3. Unidad de control (0.5 puntos).

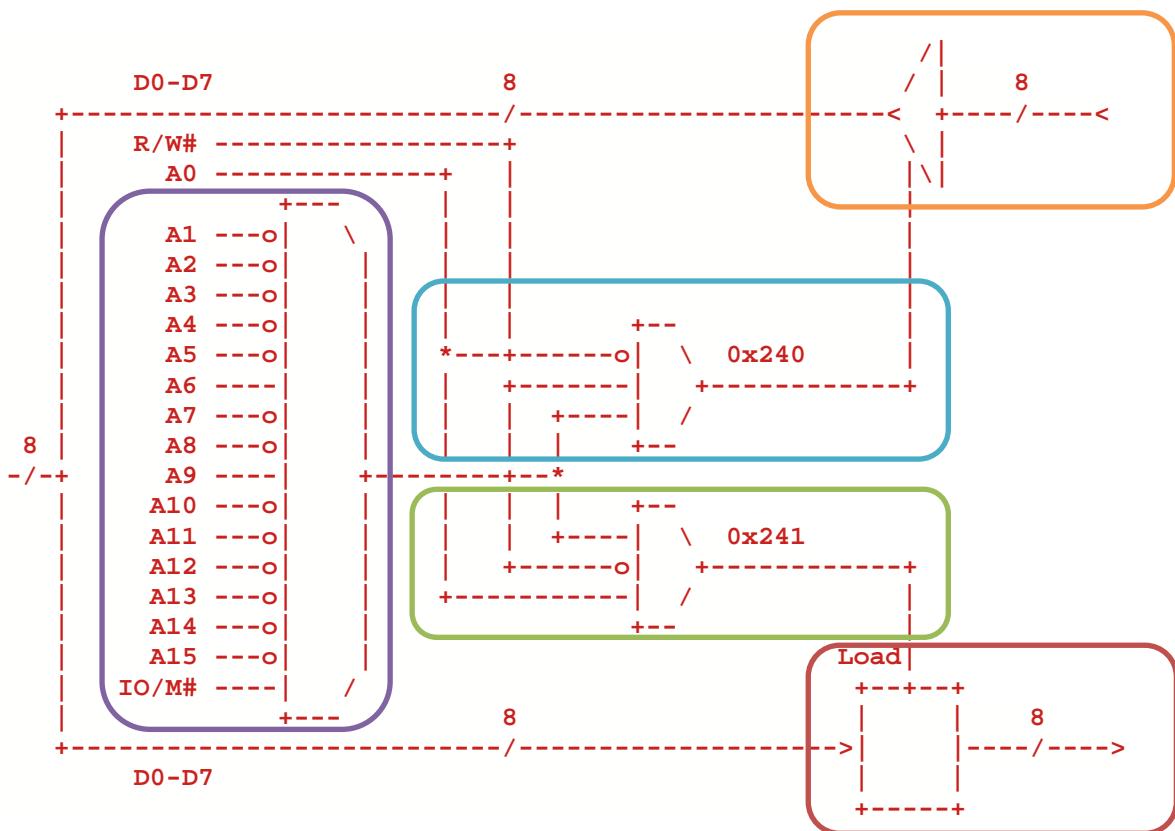
Se puntuá **0.1p** por cada zona del dibujo (total  **$0.1 \times 5 = 0.5p$** ).

El MUX no es imprescindible: ALU/Y/N/Buffer puntuán 0.1p aunque falte el MUX.



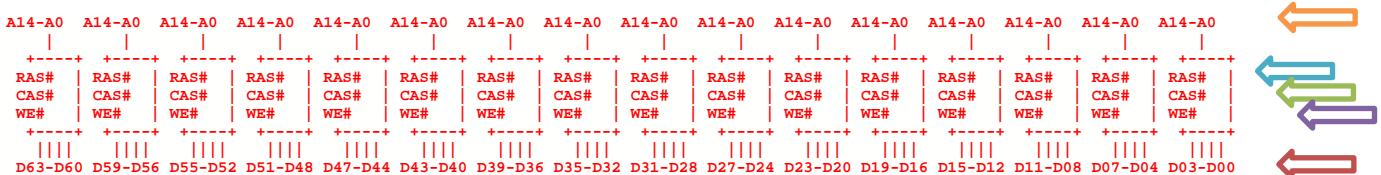
## 4. Entrada/Salida (0.5 puntos).

Se puntuá **0.1p** por cada zona del dibujo (total  **$0.1 \times 5 = 0.5p$** ).



## 5. Configuración de memoria (0.5 puntos).

Se puntuá 0.05p por cada zona del dibujo (total  $0.05 \times 10 = 0.50$ p).



## 6. Cache (0.5 puntos).

Se puntuá 0.1p por cada zona del dibujo con todos los (números/flechas) correct@s (total  $0.1 \times 5 = 0.5$ p).

$$64 \text{ GB} = 2^{36} \text{ B}$$

$$2^{36} \text{ B} / 2^6 \text{ B/block} = 2^{30} \text{ blocks}$$

$$(1\ 073\ 741\ 824)$$

$$12 \text{ MB} = 3 * 2^{22} \text{ B}$$

$$3 * 2^{22} \text{ B} / 2^6 \text{ B/line} = 3 * 2^{16} \text{ lines}$$

$$(196\ 608) (-12 = 196\ 596)$$

$$12\text{-way} = 3 * 2^2 \text{ lines/set}$$

$$3 * 2^{16} \text{ lines} / (3 * 2^2 \text{ lines/set}) = 2^{14} \text{ sets}$$

$$(16\ 384)$$

De los conjuntos (0/1/último=31) se dibujan las dos líneas (0/1, 2/3, 62/63)

Pasan a ser conjuntos (0/1/16383), primera/última líneas (0/11, 12/23, 196 596/196 607)

De memoria se dibujan los bloques que van a conj. 0/1 con etiqueta Tag=0/2, es decir, bloques (0/1) (Tag=0), bloques (64/65) (Tag=64/32=2), pasan a ser bloques (0/1) Tag=0, bloques (32 768/32 769)

...

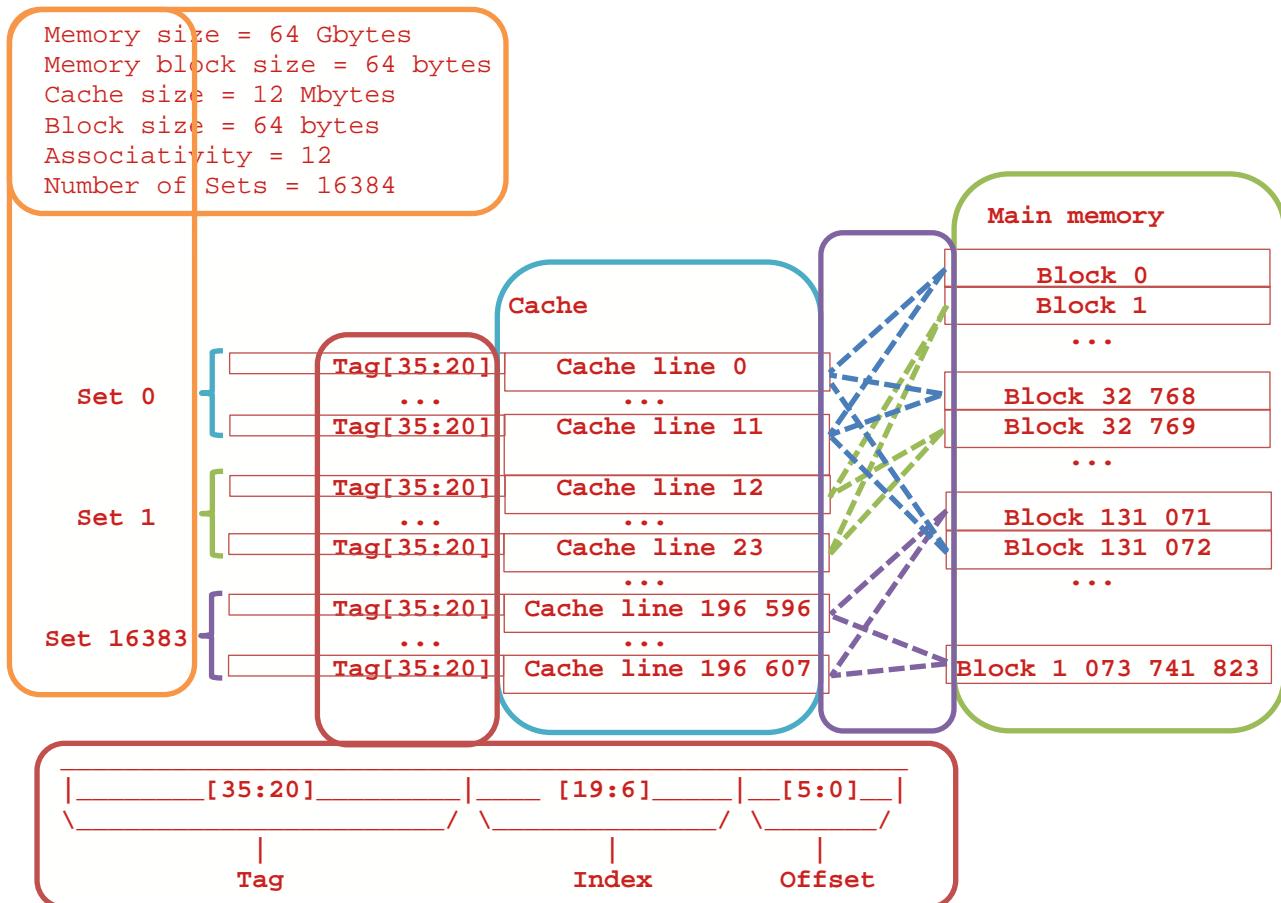
un bloque (256) que va al conj. 0 con Tag=256/32=2^3=8

pasa a ser bloque (131 072), conj.0 Tag=8

...

y también el anterior y el último (255/4095), que van al conj.31

pasan a ser (131 071/1 073 741 823)







Nombre:

DNI:

Grupo:

### Test de Teoría (3.0p)

**Todas las preguntas son de elección simple sobre 4 alternativas.**

**Cada respuesta vale 0.1p si es correcta, 0p si está en blanco o claramente tachada, -0.03p si es errónea.**  
**Anotar las respuestas (a, b, c ó d) en la siguiente tabla.**

1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30

1. En el direccionamiento inmediato el operando reside en:
    - a. en un registro del procesador
    - b. en la instrucción tras el código de operación
    - c. en memoria, en la dirección indicada
    - d. en la pila

---

  2. Una instrucción máquina del tipo "Add M,R" podría formar parte del repertorio de
    - a. una máquina pila
    - b. una máquina de acumulador
    - c. una máquina con arquitectura R/R
    - d. una máquina con arquitectura M/M

---

  3. La traducción a ASM de una función C con prototipo **void fun(<tipo> arg1, <tipo> arg2);** por parte de gcc empieza con el siguiente texto:
 

```
fun:
  movl (%rdi), %eax
  movl (%rsi), %edx
  ...
```

¿Cuál es el posible <tipo> mencionado en el prototipo?

    - a. long \*
    - b. unsigned int \*
    - c. short \*
    - d. unsigned char \*

---

  4. Se puede describir paso a paso la ejecución de la instrucción **add (%rbx, %rdx, 4), %eax;** de la siguiente manera (marcar la opción **FALSA**):
- movq \$-1, %rdx**  
**movl \$1, %edx**
- a. 0xFFFF FFFF FFFF 0001
  - b. 0xFFFF FFFF 0000 0001
  - c. 0xFFFF 0000 0000 0001
  - d. 0x0000 0000 0000 0001
- 
5. ¿Qué valor contendrá el registro rdx tras ejecutar las dos instrucciones siguientes?
- ```
400544: e8 07 00 00 00 callq 400550 <f>
400549: 48 89 03          mov %rax,(%rbx)
```
- la instrucción call suma al contador de programa la cantidad:
- a. 0x00000007
  - b. 0x00400549
  - c. 0x400544
  - d. 0x48
- 
7. Una función C declarada como **int get\_var\_digit(size\_t index, size\_t digit)** genera como código ensamblador

```

movq var(,%rdi,8), %rax
movl (%rax,%rsi,4), %eax
ret

```

Se puede adivinar que:

- var es un array multi-nivel (punteros a enteros) de cuatro filas
  - var es un array multi-nivel pero no se pueden adivinar las dimensiones
  - var es un array bidimensional de enteros, con ocho columnas
  - var es un array bidimensional de enteros, con cinco columnas
- 
- En una unidad de control microprogramada se tiene un campo de 14 señales de control de las cuales sólo se activaría una o ninguna en un ciclo de reloj, nunca dos o más en el mismo ciclo de reloj. Sería entonces posible...
    - codificarlas con 4 bits, y sobraría un código que quedaría sin uso
    - codificarlas con 5 bits, y sobrarían dos códigos que quedarían sin uso
    - solaparlas en un solo campo de 5 bits, ahorrando por tanto 9 bits
    - solaparlas en un solo campo de 7 bits, ahorrando por tanto 6 bits
- 
- En el contexto de microprogramación, el control residual...
    - intenta disminuir la cantidad de "bits residuales", usando las técnicas de codificación y/o solapamiento de campos, como opuestas a la microprogramación directa o "inmediata".
    - se refiere a que cuanto más codificación y/o solapamiento se use, menos capacidad para expresar paralelismo se tiene, siendo ese menor control un "residuo" o consecuencia no deseada de dichas técnicas no "inmediatas"
    - clasifica las microinstrucciones del microcódigo según formen parte de micropogramas ("microinstr. inmediatas") o no ("microinstr. residuales")
    - consiste en almacenar señales de control en un "registro de control residual" para usarlas en ciclos posteriores, a diferencia del "control inmediato", en donde los bits se utilizan inmediatamente

**10.** Suponer que un procesador ideal que ejecuta cada instrucción en T segundos se segmenta en cuatro etapas ideales de duración T/4. ¿Cuál razonamiento es correcto?

- Se espera una reducción de prestaciones porque además de ejecutar las instrucciones hay que segmentarlas (coste de la segmentación)
  - Se espera una reducción de prestaciones porque la duración del ciclo de reloj vendrá impuesta por la etapa más lenta
  - Se espera un aumento de prestaciones debido al efecto de los riesgos (hazards) sobre el avance de las instrucciones en el cauce
  - Se espera un aumento de prestaciones debido a que las cuatro etapas solapan su funcionamiento, con una aceleración ideal de 4x
- 

**11.** Un salto condicional del tipo "delayed branch", o salto retardado, ejecuta la(s) instrucción(es) siguiente(s)...

- sólo si el salto se produce (las ignora si NO se produce), de manera que instrucción(es) en el destino del salto podrían adelantarse tras la propia instrucción de salto
  - sólo si el salto NO se produce (las ignora si se produce), de manera que instrucción(es) en el destino del salto podrían adelantarse tras la propia instrucción de salto
  - siempre, de manera que instrucción(es) anterior(es) al salto podrían colocarse tras la propia instrucción de salto
  - nunca, de manera que instrucción(es) anterior(es) al salto no podrían colocarse tras la propia instrucción de salto
- 

**12.** Un computador con 20 líneas de dirección y memoria de bytes tiene 640KB de RAM, 128KB de ROM, y utiliza E/S mapeada en memoria. ¿Cuál es el número máximo de periféricos que pueden conectarse, si cada uno de ellos utiliza 32 direcciones?

- $2^{10}$
  - $2^{11}$
  - $2^{12}$
  - $2^{13}$
- 

**13.** La consulta de estado que se puede llevar a cabo en una operación de salida mediante E/S programada sirve para...

- a. consultar si el dispositivo tiene algún dato de salida disponible
  - b. consultar si el dispositivo está aún ocupado, por ejemplo con alguna operación de salida anterior
  - c. consultar si el dispositivo funciona correctamente
  - d. ninguna de las respuestas anteriores es correcta
- 

14. ¿Qué conjunto de componentes permite construir una memoria 256Mx32? (sin que sobren componentes)

- a. 16 chips 64Mx4
  - b. 32 chips 64Mx4
  - c. 16 chips 64Mx16
  - d. Ninguna de las anteriores
- 

15. ¿Cuántas líneas de dirección (patillas) son necesarias para direccionar un chip de memoria DRAM de 256K x 4?

- a. 9
  - b. 13
  - c. 18
  - d. 22
- 

16. ¿Qué arquitectura es típica en procesadores RISC?

- a. registro-registro
  - b. registro-memoria
  - c. memoria-registro
  - d. memoria-memoria
- 

17. ¿Cuál de las siguientes características es posterior a la segunda generación de computadores?

- a. Memoria de núcleos de ferrita
  - b. Compilador
  - c. Memoria cache
  - d. Lenguaje ensamblador
- 

18. Si el registro rax contiene **x**, la sentencia en **C x &= 0x1;** se traducirá a ensamblador como:

- a. andq \$1, %rax
  - b. orq \$0x1, %rax
  - c. shrq %rax
  - d. sarq %rax
- 

19. Para crear espacio en la pila para variables locales sin inicializar suele realizarse la siguiente operación:

- a. Restar una cantidad positiva a rbp.
  - b. Sumar una cantidad positiva a rbp.
  - c. Restar una cantidad positiva a rsp.
  - d. Sumar una cantidad positiva a rsp.
- 

20. Si la estructura struct a ocupa un espacio de 26 bytes en memoria, ¿cuántos bytes ocupa la siguiente estructura struct b cuando se compila en 64 bits?

```
struct b {  
    struct a a1;  
    int i;  
    struct a a2;  
};
```

- a. 24
  - b. 58
  - c. 60
  - d. 64
- 

21. En la secuencia de programa siguiente:

```
628: e8 cd ff ff ff  callq  5fa <suma>  
62d: 48 83 c4 20      add    $0x20,%rsp
```

¿cuál es el valor que introduce en la pila la instrucción callq?

- a. 0xffffffffcd
  - b. 0x5fa
  - c. 0x628
  - d. 0x62d
- 

22. Un archivo .o que contiene código objeto reubicable:

- a. Contiene instrucciones máquina binarias.
  - b. Contiene instrucciones máquina y directivas en ensamblador.
  - c. Puede ejecutarse directamente.
  - d. Contiene las direcciones definitivas de las variables globales.
- 

23. ¿Cuál de las siguientes sentencias sobre la unidad de control es **FALSA**?

- a. Cuanto más horizontal es la microprogramación, más largas son las microinstrucciones
- b. Debido al pequeño número de operaciones simples, la sección de control de un procesador RISC puede ser cableada en lugar de microprogramada
- c. El programador de lenguaje ensamblador necesita conocer la microarquitectura del ordenador

- d. Es posible realizar el diseño físico de una unidad de control cableada de manera semiautomática
- 

24. En una unidad de control microprogramada con formato de microinstrucciones vertical, un subcampo que deba especificar 16 señales de control, codificadas de tal forma que pueda activarse sólo una o ninguna señal de control, habrá de tener una anchura mínima de:

- a. 4 bits
  - b. 5 bits
  - c. 16 bits
  - d. 17 bits
- 

25. La técnica de "adelanto de registros" (register forwarding) en un cauce segmentado se usa para limitar el impacto de los riesgos...

- a. estructurales
  - b. organizativos
  - c. de control
  - d. de datos
- 

26. Un sistema no segmentado tarda 10 ns en procesar una tarea. La misma tarea puede ser procesada en un cauce (pipeline) de 4 segmentos con un ciclo de reloj de 4 ns. Cuando se procesan muchas tareas, la ganancia máxima de velocidad que se obtiene se acerca a:

- a. 10
  - b. 4
  - c. 40
  - d. 2,5
- 

27. ¿Cuál de las siguientes afirmaciones es cierta?

- a. La E/S en memoria emplea la patilla IO/M#.
  - b. En E/S independiente, las instrucciones de acceso a memoria suelen ser más largas que las de E/S.
  - c. La E/S en memoria facilita la protección.
  - d. En la E/S en memoria las instrucciones de E/S son fácilmente reconocibles.
- 

28. La instrucción máquina di (Disable Interrupts), conocida como cli (Clear Interrupt Flag) en x86:

- a. Desactiva todas las interrupciones enmascarables.

- b. Desactiva las interrupciones de inferior o igual prioridad a una dada.
  - c. Desactiva determinados niveles de interrupción de forma selectiva.
  - d. Desactiva las interrupciones software.
- 

29. ¿Cuál de las siguientes afirmaciones sobre la jerarquía de memoria es cierta?

- a. Para aumentar la eficiencia se transfieren bloques completos.
  - b. Toda la información que el procesador necesita está en el nivel 1.
  - c. Si una palabra no se encuentra en el tercer nivel entonces se busca en el segundo nivel.
  - d. A medida que nos alejamos del procesador, el tamaño de memoria disminuye.
- 

30. ¿Cuántas líneas de dirección (patillas) son necesarias para direccionar un chip de memoria DRAM de 4096 x 4?

- a. 6
  - b. 10
  - c. 11
  - d. 12
-

**Nombre:**
**DNI:**
**Grupo:**

## Test de Prácticas (4.0p)

**Todas las preguntas son de elección simple sobre 4 alternativas.**

**Cada respuesta vale 0.2p si es correcta, 0 si está en blanco o claramente tachada, -0.06p si es errónea.**

**Anotar las respuestas (a, b, c ó d) en la siguiente tabla.**

|   |   |   |   |   |   |   |   |   |    |    |    |    |    |    |    |    |    |    |    |
|---|---|---|---|---|---|---|---|---|----|----|----|----|----|----|----|----|----|----|----|
| 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 | 11 | 12 | 13 | 14 | 15 | 16 | 17 | 18 | 19 | 20 |
|   |   |   |   |   |   |   |   |   |    |    |    |    |    |    |    |    |    |    |    |

1. En la práctica "media" se pide como ejercicio previo sumar una lista de N enteros sin signo en doble precisión acumulando acarreos. Para aplicar esta técnica se sugiere utilizar la(s) instrucción(es)...

- a. ADC \$0, ...
  - b. CLTD, ADD y ADC
  - c. JNC y INC
  - d. CLTQ, CQTO y IDIV
- 

2. En la práctica "media" se pide como ejercicio previo sumar una lista de N enteros sin signo en doble precisión extendiendo con ceros. Para aplicar esta técnica se sugiere utilizar la(s) instrucción(es)...

- a. ADC \$0, ...
  - b. CLTD, ADD y ADC
  - c. JNC y INC
  - d. CLTQ, CQTO y IDIV
- 

3. En la práctica "media" se pide como ejercicio previo sumar una lista de N enteros en doble precisión extendiendo el signo. Para aplicar esta técnica se sugiere utilizar la(s) instrucción(es)...

- a. ADC \$0, ...
  - b. CLTD, ADD y ADC
  - c. JNC y INC
  - d. CLTQ, CQTO y IDIV
- 

4. La práctica "popcount" debía calcular la suma de bits (peso Hamming) de los elementos de un array. Un estudiante entrega la siguiente versión de popcount5:

```
int popcount5(int* array, size_t len){
    size_t i,j,val;
    long x;
    int result=0;
    for (i=0; i<len; i++){
        x = array[i];
        val = 0;
        for (j=0; j<8; j++){
            val += x & 0x0101010101010101L;
            x >>= 1;
        }
        val += (val >> 32);
        val += (val >> 16);
        val += (val >> 8);
        result+= val & 0xFF;
    }
    return result;
}
```

Esta función no sigue la adaptación a 32 bits recomendada en clase como versión "oficial", y conserva muchos vestigios de la versión de 64 bits, como los tipos de **array**, **x**, **val**, la máscara **0x...01L** y el desplazamiento **>>32**.

Esta función popcount5 fallaría:

- a. con array={0x80000000, 0x00400000, 0x00000200, 0x00000001}
  - b. con array={0, -1, -2, -3}
  - c. con ambos ejemplos
  - d. con ninguno de los dos ejemplos
- 
5. La práctica "popcount" debía calcular la suma de bits (peso Hamming) de los elementos de un array. Un estudiante entrega la siguiente versión de popcount6:
- ```
int pc6(unsigned* array, size_t len){
    size_t i;
    unsigned x;
```

```

int result=0;

const unsigned m1 = 0x55555553;
const unsigned m2 = 0x33333333;
const unsigned m4 = 0x0f0f0f0f;
const unsigned m8 = 0x00ff00ff;
const unsigned m16 = 0x0000ffff;

for (i=0; i<len; i++){
    x = array[i];

    x = (x & m1) + ((x >> 1) & m1);
    x = (x & m2) + ((x >> 2) & m2);
    x = (x & m4) + ((x >> 4) & m4);
    x = (x & m8) + ((x >> 8) & m8);
    x = (x & m16) + ((x >> 16) & m16);

    result+= x;
}
return result;
}

```

Esta función sólo se diferencia de la versión "oficial" recomendada en clase en algún dígito de alguna máscara.

Esta función `popcount6` fallaría:

- a. con `array={0x80000000, 0x00400000, 0x00000200, 0x00000001}`
  - b. con `array={0x10000000, 0x00200000, 0x00000400, 0x00000008}`
  - c. con ambos ejemplos
  - d. con ninguno de los dos ejemplos
- 

6. En una bomba como las estudiadas en prácticas, del tipo...

```

400746: lea    0xc(%rsp),%rsi
40074b: lea    0x1ab(%rip),%rdi
                # 4008fd <stdin_u+0x10d>
400752: mov    $0x0,%eax
400757: callq  400590 <scanf@plt>
40075c: cmpl   $0x400600,0xc(%rsp)
400764: je     40076b <main+0x8c>
400766: callq  400697 <boom>
40076b: callq  4006bb <defused>

```

...el código numérico (pin) es...

- a. el entero 0x400590
  - b. el entero 0x400600
  - c. el entero almacenado a partir de 0xc(%rsp)
  - d. el entero alm. a partir de 0x1ab+0x400752
- 

7. En la práctica "blink" de Arduino, un estudiante muestra a los profesores lo que le han enseñado en algún otro lugar: conectando un led directamente entre las patillas *Digital pin 13* (`LED_BUILTIN`) y `GND` (justo al lado) también parpadea. Cabe esperar la siguiente respuesta de los profesores:

- a. Exactamente esa solución viene en el guión, no hace falta aprenderla fuera de clase
  - b. Nosotros conectamos los led en serie con una resistencia como dice el guión
  - c. Hay que revisar ese equipo en concreto, debe tener algún defecto, ese led no debe parpadear
  - d. Es cierto, es un error de diseño del guión, es más sencillo conectarlo así
- 

8. Suponga una memoria cache con las siguientes propiedades: Tamaño: 32 KB. Política de reemplazo: LRU. Estado inicial: vacía (todas las líneas inválidas). Tamaño de línea 64 B. Dado el siguiente fragmento de código:

```

int x[262144], y[262144], a=0;
for (i = 0; i < 262144; i++)
    a += x[i] * y[i];

```

¿Cuál será la tasa de fallos aproximada que se obtiene en la ejecución del bucle anterior? Suponer que el tamaño de los tipos de datos es como en x86-64, y que el compilador optimiza el acceso a las variables `a/i` en un registro.

- a. 1/2 (un fallo por cada 2 accesos)
  - b. 1/4 (un fallo por cada 4 accesos)
  - c. 1/8 (un fallo por cada 8 accesos)
  - d. 1/16 (un fallo por cada 16 accesos)
- 

9. En el programa `line.cc` de la práctica "cache", se accede al vector saltando...

- a. de byte en byte
  - b. de 64 en 64 bytes
  - c. de 1 KB en 1 KB
  - d. de line en line bytes, donde line barre los valores desde 1 hasta 1K en un bucle for
- 

10. En la práctica de cache hemos hecho una gráfica con el código `line.cc` ¿Qué forma tiene la gráfica que se puede obtener?

- a. Forma de U (o V), con un tramo descendente y otro ascendente
  - b. Forma de /, una gráfica siempre creciente
  - c. Forma de media U seguida de \, es decir, un tramo descendente suave, un pequeño tramo horizontal, y un tramo descendente lineal
  - d. Una escalera con varios tramos horizontales
- 

11. ¿Cuál de las siguientes instrucciones es incorrecta?

- a. `shr $1,%rdx`
- b. `shr %rdx`
- c. `shr %cl,%rdx`

d. shr %rcx,%rdx

12. En la práctica “media” se pide sumar una lista de 16 enteros CON signo de 32 bits en una plataforma de 32 bits sin perder precisión, esto es, evitando overflow. ¿Cuál es el mayor valor negativo (menor en valor absoluto) que repetido en toda la lista de 16 enteros causaría overflow con 32bits?

PISTA: Sumar un número 16 veces == multiplicarlo por 16 == desplazarlo 4 bits a la izquierda

- a. 0xffff ffff
- b. 0xfc00 0000
- c. 0xf7ff ffff
- d. 0xf000 0000

13. La función siguiente, que aparece en el esqueleto suma\_09 presentado en la práctica “popcount”

```
int suma3(int* array, int len){  
    asm("mov $0, %%eax\n"  
        "    mov $0, %%rdx\n"  
        "bucle:\n"  
        "    add (%%rdi,%%rdx,4), %%eax\n"  
        "    inc %%rdx\n"  
        "    cmp %%rdx,%%rsi\n"  
        "    jne bucle\n"  
        "    : : : // output\n"  
        "    : : : // input\n"  
        "    : \"cc\", // clobber\n"  
        "    \"rax\", \"rdx\"\n";  
}
```

no incluye la instrucción **ret** dentro de la sentencia **asm** porque:

- a. porque la función no devuelve nada
- b. porque suma3 es un puntero a función
- c. porque la incluye automáticamente el compilador
- d. porque suma3 no se llamará con call sino con jmp

14. Respecto a la inicialización de los elementos del array en el esqueleto suma\_09 de la práctica “popcount”:

```
#define SIZE (1<<16)  
for (i=0; i<SIZE; i++)  
    lista[i]=i;
```

- a. Cualquiera de las tres funciones suma del programa suma\_09\_Casm irán más rápido con esta inicialización que con números aleatorios.
- b. Las tres funciones suma del programa suma\_09\_Casm irán igual de rápido con esta inicialización que con números aleatorios.

- c. **popcount1** (bucle interno for) irá más rápido con esta inicialización que con números aleatorios
- d. **popcount2** (bucle interno while) irá más lento con esta inicialización que con números aleatorios.

15. Un estudiante entrega la solución de una bomba y en ella aparece el comando de gdb “**p(char\*)\$rdi**”. Este comando:

- a. muestra únicamente el contenido de %dil
- b. es erróneo sintácticamente
- c. muestra el valor de la dirección de memoria contenida en %rdi y el string almacenado a partir de esa dirección de memoria
- d. muestra el contenido de %rdi interpretado como un string de 8 caracteres

16. En la realización de la práctica de la bomba digital compilada en 32 bits, una parte del código máquina es el siguiente:

```
0x080486e8: call 0x8048524 <strncmp>  
0x080486ed: test %eax,%eax  
0x080486ef: je 0x80486f6 <main+134>  
0x080486f1: call 0x8048604 <boom>
```

¿Cuál de los siguientes comandos de gdb cambiaría el salto condicional por un salto incondicional?

- a. set \$0x080486ef=0xeb
- b. set \*(char\*)0x080486ef=0xeb
- c. set \*(char\*)0x080486f6=jmp
- d. set %0x080486ef=0xeb

17. ¿Cuál de las siguientes patillas (pin) no se menciona (no se pide usar/no se debe usar) en la 1ª práctica del Theremín de luz de Arduino? (Proyecto p06 / sólo fotocélula y zumbador)

- a. Analog In A0
- b. AREF
- c. ledPin = 13
- d. GND

18. En la práctica del Theremín de luz de Arduino se pide combinar el código del Proyecto p06 (sólo fotocélula y zumbador) con el de la Lección 26 (fotocélula y leds). Para conseguir que se enciendan y apaguen correctamente todos los led, lo mejor sería:

- a. Reutilizar la variable pitch calculada mediante map() como la nueva numLEDSLit

- b. Fusionar las variables `sensorValue` y `reading` de forma que sólo se llame una vez a `analogRead()`
  - c. Calcular `numLEDSLit` con otra función `map()` de 0 a 8, en lugar de barrer de 50 a 4000
  - d. Copiar uno y otro código en las secciones correspondientes, no se puede fusionar nada
- 

19. El código de `size.cc` accede al vector saltando de 64 en 64. ¿Por qué?

- a. Porque cada elemento del vector ocupa 64 bytes.
  - b. Para recorrer el vector más rápidamente.
  - c. Porque el tamaño de cache L1 de todos los procesadores actuales es de 64 KB.
  - d. Para anular los aciertos por localidad espacial, es decir, para que sólo pueda haber aciertos por localidad temporal.
- 

20. Un servidor tiene dos procesadores Intel Xeon E5-2620 v3@ 2.40GHz (2,4 Ghz, 6 núcleos, 12 hebras, 15MiB de caché L3, reloj DDR4 a 1866 MHz). Suponga que un proceso, que se ejecuta en un único núcleo, tiene que ordenar un vector de datos de usuarios accediendo repetidamente a sus elementos. Cada elemento es una estructura de datos y tiene un tamaño de 4 KB. Según <http://www.cpu-world.com/> los tamaños de cache son:

Cache: L1 data    L1 instr.    L2    L3  
Size: 6 x 32KB  6 x 32KB  6 x 256KB  15MB

Si representamos en una gráfica las prestaciones en función del número de usuarios a ordenar, ¿para qué límites teóricos en el número de usuarios se observarán saltos en las prestaciones debidos a accesos a la jerarquía de memoria?

- a. 4 / 32 / 512 usuarios
  - b. 8 / 64 / 3840 usuarios
  - c. 48 / 384 / 23040 usuarios
  - d. 32 / 256 / 8192 usuarios
-



**Nombre:**

**DNI:**

**Grupo:**

## Examen de Problemas (3,0 p)

**1. Ensamblador.** (0.7 puntos). Compilando con gcc -Os la siguiente función C:

```
unsigned ndof (unsigned y) {
    return fórmula-oculta;
}
```

se obtiene este código ensamblador de x86-64:

**ndof:**

```
    movl $28, %eax
    testb $3, %dl
    jne .L1
    movl $100, %ecx
    movl %edi, %eax
    xorl %edx, %edx
    divl %ecx
    movl $29, %eax
    testl %edx, %edx
    jne .L1
    xorl %edx, %edx
    movl $400, %ecx
    movl %edi, %eax
    divl %ecx
    cmpl $1, %edx
    sbbl %eax, %eax
    notl %eax
    addl $29, %eax
```

; Pista: como la instrucción sbb src, dst  
; ; ; ; ; ; ; ; ;  
; ; ; ; ; ; ; ; ; ; ;  
; ; ; ; ; ; ; ; ; ; ;  
; ; ; ; ; ; ; ; ; ; ;  
; ; ; ; ; ; ; ; ; ; ;  
; ; ; ; ; ; ; ; ; ; ;  
; ; ; ; ; ; ; ; ; ; ;  
; ; ; ; ; ; ; ; ; ; ;  
; ; ; ; ; ; ; ; ; ; ;  
; ; ; ; ; ; ; ; ; ; ;

**.L1:**

```
    ret
```

- (0.1p) ¿Qué valor devuelve la función ndof al usar como argumento y=2020?
- (0.5p) Escriba en C el cuerpo completo de la función ndof (sustituya “return fórmula-oculta;” por una sentencia o varias sentencias en C que realicen la misma función que el código ensamblador anterior).
- (0.1p) ¿Qué calcula la función?

**2. Ensamblador** (0.4 puntos). La siguiente función realiza una serie de operaciones y cálculos sobre la siguiente estructura C.

```
struct rec {
    num_t1 i;
    num_t2 a[M][N];
    num_t1 j;
};

int fun(struct rec* r){
    return r->a[r->i][r->j];
}
```

num\_t1 y num\_t2 son tipos declarados con typedef. M y N son constantes enteras positivas declaradas con #define.

La traducción a ensamblador con gcc -Og de esta función es:

```
fun:
    movslq    16(%rdi), %rdx
    movslq    (%rdi), %rax
    leaq      (%rdi,%rax,4), %rax
    movsbl   4(%rdx,%rax), %eax
    ret
```

a) (0.2p) ¿Qué tipos son num\_t1 y num\_t2?

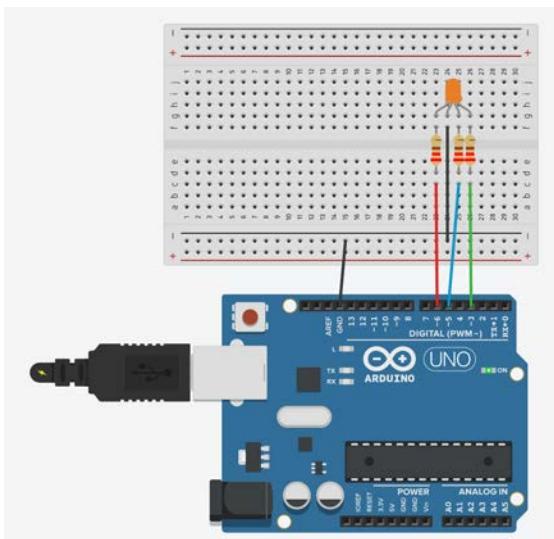
b) (0.2p) ¿Qué valores tienen las constantes M y N?

**3. Unidad de control** (0.6 puntos). La máquina de Tanenbaum estudiada en clase se implementó con un diseño horizontal de 79 microinstrucciones de 32 bits, y con un diseño vertical de 160 microinstrucciones de 12 bits. A ambos diseños se les podría haber aplicado nanoprogramación, pero un simple vistazo al microcódigo del diseño horizontal hace pensar que habrá del orden de 10-20 microinstrucciones repetidas como mucho. En el diseño vertical se ven más repeticiones, pero tampoco se acercan ni por asomo al 50% de microinstrucciones repetidas. Para fijar conceptos y entender por qué Tanenbaum no se planteó realizar nanoprogramación en su máquina:

- a) (0.3p) calcular qué porcentaje de microinstrucciones del diseño horizontal deberían estar repetidas para que la nanoprogramación fuera más ventajosa que el diseño vertical (en el sentido de ahorrar memoria de control)
- b) (0.3p) calcular qué porcentaje de microinstrucciones del diseño vertical deberían estar repetidas para que resultara ventajoso aplicarle nanoprogramación (en el mismo sentido)

Notar que incluso si se diera ese porcentaje de repetición, la nanoprogramación implicaría una lectura adicional a la nanomemoria que retardaría la máquina aún más.

**4. Entrada/Salida** (0.4 puntos). Disponemos del siguiente circuito que conecta una placa Arduino con un led RGB. El led RGB dispone de tres patillas R, G y B conectadas a los pines 6, 3 y 5 de la placa Arduino. Complete el programa de Arduino para que el led RGB pase gradualmente de rojo a verde en 10 segundos aproximadamente, luego de nuevo de rojo a verde en otros 10 segundos, y así sucesivamente.



```
#define RED 6
#define GREEN 3
#define BLUE 5

void setup() {}

void loop() {
    /* Escriba aquí el código */
}
```

Ayuda:

`analogWrite(pin, value)`

*Escribe un valor analógico entre 0 y 255 en un pin. Puede usarse para variar el brillo de un led. No es necesario llamar a `pinMode()` para fijar un pin como salida antes de llamar a `analogWrite()`.*

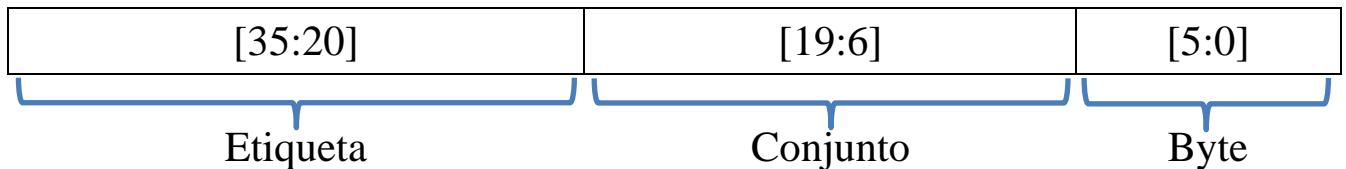
`delay(ms)`

*Pausa el programa la cantidad de milisegundos especificada como parámetro.*

## 5. Configuración de memoria (0.5 puntos).

- (0.4p) Partiendo de circuitos SRAM de 2Mx8, dibuje un sistema de memoria direccionable por palabras de 32 bits, con un bus de direcciones de 22 bits, un bus de datos de 32 bits, una señal Write y una señal Read.
- (0.1p) Indique la primera y la última dirección en hexadecimal de cada fila de circuitos.

## 6. Cache (0.4 puntos). Las direcciones físicas de memoria principal del procesador Intel Core i7-10710U tienen el siguiente formato desde el punto de vista de la cache L3 de 12 MB:



Calcule:

- (0.1p) Tamaño de memoria principal en bytes
- (0.1p) Tamaño de línea en bytes
- (0.1p) Tamaño de conjunto en bytes
- (0.1p) Número de vías de la cache L3

### Test de Teoría (3.0p)

<b>1</b>	<b>2</b>	<b>3</b>	<b>4</b>	<b>5</b>	<b>6</b>	<b>7</b>	<b>8</b>	<b>9</b>	<b>10</b>	<b>11</b>	<b>12</b>	<b>13</b>	<b>14</b>	<b>15</b>	<b>16</b>	<b>17</b>	<b>18</b>	<b>19</b>	<b>20</b>	<b>21</b>	<b>22</b>	<b>23</b>	<b>24</b>	<b>25</b>	<b>26</b>	<b>27</b>	<b>28</b>	<b>29</b>	<b>30</b>
b	d	b	b	d	a	b	a	d	d	c	d	b	b	a	a	c	a	c	c	d	a	c	b	d	d	b	a	a	

### Test de Prácticas (4.0p)

<b>1</b>	<b>2</b>	<b>3</b>	<b>4</b>	<b>5</b>	<b>6</b>	<b>7</b>	<b>8</b>	<b>9</b>	<b>10</b>	<b>11</b>	<b>12</b>	<b>13</b>	<b>14</b>	<b>15</b>	<b>16</b>	<b>17</b>	<b>18</b>	<b>19</b>	<b>20</b>
c	a	b	c	b	b	b	d	d	c	d	c	c	b	c	b	b	c	d	b

### Examen de Problemas (3.0p)

#### 1. Ensamblador (0.7 puntos).

a) puntúa **0.1p**, b) puntúa **0.5p**, c) puntúa **0.1p** (total **0.70p**).

a) La función devuelve **29** cuando le pasamos 2020 como argumento.

b) Algunas alternativas posibles

Traducción directa:

```
unsigned ndof (unsigned y) {
    if (y % 4 != 0)
        return 28;
    if (y % 100 != 0)
        return 29;
    return 29 - (y % 400 != 0);
}
```

Función original:

```
unsigned ndof (unsigned y) {
    return y % 4 == 0 ? 
           y % 100 == 0 ? 
               y % 400 == 0 ? 
                   29 : 28 : 29 : 28 ;
```

c) La función “ndof” calcula el **número de días de febrero** (number days of february) del año **y**.

[https://es.wikipedia.org/wiki/A%C3%B3\\_bisiesto#Algoritmo\\_computacional](https://es.wikipedia.org/wiki/A%C3%B3_bisiesto#Algoritmo_computacional)

Material adicional:

<pre>ndof: movl \$28, %eax      # eax = 28 testb \$3, %dl       # y % 4 ? jne .L1             # y % 4 != 0 ==&gt; return 28 (no bisiesto)  movl \$100, %ecx     # ecx = 100 movl %edi, %eax     # eax = y xorl %edx, %edx     # edx = 0 divl %ecx            # edx:eax / ecx &lt;=&gt; y / 100 movl \$29, %eax      # eax = 29 testl %edx, %edx     # y % 100 ? jne .L1             # y % 100 != 0 ==&gt; return 29 (bisiesto)  xorl %edx, %edx     # edx = 0 movl \$400, %ecx     # ecx = 400 movl %edi, %eax     # eax = y divl %ecx            # edx:eax / ecx &lt;=&gt; y / 400 cmpl \$1, %edx       # edx=0? =&gt; CF== 1 // edx&gt;0? =&gt; CF == 0 sbbl %eax, %eax      # edx=0? =&gt; eax== -1 // edx&gt;0? =&gt; eax== 0 notl %eax            # edx=0? =&gt; eax== 0 // edx&gt;0? =&gt; eax== -1 addl \$29, %eax      # y % 400 == edx == 0 ==&gt; return 29 (bisiesto) .L1:                # y % 400 == edx != 0 ==&gt; return 29+(-1) (no) ret</pre>	
---	--

## 2. Ensamblador(0.4 puntos).

Se puntúa **0.1p** por cada pregunta (total **0.1 x 4 = 0.40p**)

- a) (0.2p) num\_t1 es **int**, por cualquiera de los movslq, corroborado por desplazamiento 4 en movsbl 4(...)  
num\_t2 es **char**, por movsbl, corroborado por cálculo lea ( $\&r + i*4 + j*4 = \&r + 4 + (i*4 + j)*1$ )
- b) (0.2p) N=4, por el factor leaq i\*4 + j  
M=3, por el desplazamiento movslq 16(%rdi) para j (i 4B, a 12B, j 4B)

Material adicional:

	<pre>fun:     movslq    16(%rdi), %rdx      # r[offset 16] j? 4B signo =&gt; rdx     movslq    (%rdi), %rax       # r[offset 0] i 4B signo =&gt; rax     leaq      (%rdi,%rax,4), %rax # calcular &amp;r + i*4 y seguir calculando     movsbl    4(%rdx,%rax), %eax # r[offset 4+j+i*4] a[i][j] 1B signo     ret</pre>	
	<pre>typedef int num_t1; typedef char num_t2;  #define M 3 #define N 4</pre>	<pre>r: +-----+  00         i(4B)      +-----+ --\  04         a[0] \      +-----+    08         a[1]   a(12B = 3*4 B)      +-----+    12         a[2] /      +-----+ --/  16         j(4B)      +-----+</pre>

## 3. Unidad de control (0.6 puntos).

Cada solución (a,b) requiere una ecuación, un dato x (únicas/repet.), un resultado r, puntúa **0.1p** cada uno (total **0.1 x 6 = 0.6p**).

Solución: Los diseños originales ocupaban: Horizontal 79ui \* 32b/ui = 2528b, Vertical 160ui \* 12b/ui = 1920b  
se reduce al  $1920/2528 = 0.7595 = 75.95\%$ , ahorro del 24.05%, aunque ese dato no se pide

### a) (0.3p)

Para acercarse al 50% repetición, de 79 microinstrucciones se deberían repetir unas 40, y otras x=40 deberían ser únicas.  
Haciendo prueba y error alrededor de esa cantidad x=40ui únicas, el tamaño de las memorias de control sería

	micro:	nano:	
x	$[\lg_2 x]$	$79 * [\lg_2 x]$	$x * 32$ Total bits
64	6	474	2048 2522 ~ 2528 por 6bits no merece la pena diferenciar de 79ui 64 únicas
40	6	474	1280 1754 < 1920 40ui únicas ahorra más que diseño vertical
32	5	395	1024 1419 << 32ui únicas usarían micromemoria de $79*5$ en lugar de $79*6$

El límite estaría entre 40 y 64 microinstrucciones únicas, en concreto:

$$474 + 32x \leq 1920 ; x \leq (1920 - 474)/32 = 1446/32 = 45.19 ; x \leq 45 \text{ únicas}$$

Solamente por comprobar que el cálculo es correcto:

	micro:	nano:	
x	$[\lg_2 x]$	$79 * [\lg_2 x]$	$x * 32$ Total bits
46	6	474	1472 1946 > 1920
45	6	474	1440 1914 < 1920

Empezamos buscando un 50% repetición porque el ejemplo de los apuntes es todavía más repetido (640ui, 280 únicas < 320), pero vale cualquier método (incluso fuerza bruta por prueba y error) con tal de localizar que  $32 < x < 64$ , para poder plantear la ecuación que puntúa ( $474 + 32x \leq 1920$ ) evitando el redondeo hacia arriba  $[\lg_2 x]$ .

El porcentaje de repetición debería ser entonces mayor o igual que

$$79 - 45 = 34 \text{ microinstrucciones repetidas} ; 34/79 = 43.04\% ; \text{rep} \geq 43.04\%$$

### b) (0.3p) mismo procedimiento

	micro:	nano:	
x	$[\lg_2 x]$	$160 * [\lg_2 x]$	$x * 12$ Total bits
128	7	1120	1536 2656 > 1920
64	6	960	768 1728 < 1920

El límite estaría entre 64 y 128 microinstrucciones únicas, en concreto:

$$1120 + 12x \leq 1920 ; x \leq (1920 - 1120)/12 = 800/12 = 66.67 ; x \leq 66 \text{ únicas}$$

Solamente por comprobar que el cálculo es correcto:

		micro:	nano:	Total bits
x	[lg2 x]	$160 * [\lg 2 x]$	$x * 12$	
67	7	1120	804	$1924 > 1920$
66	7	1120	792	$1912 < 1920$

El porcentaje de repetición debería ser entonces mayor o igual que  
 $190-66=94$  microinstrucciones repetidas ;  $94/160 = 58.75\%$  ;  $\text{rep} \geq 58.75\%$

#### 4. Entrada/Salida (0.4 puntos).

Nota completa **0.40p** si el programa funciona. Las soluciones ocupan unas 4-8 líneas, por eso en modo “rescate” se puntúa **0.05-0.1p** por cada línea “acertada” (total **0.05 x 8 = 0.1 x 4 = 0.4p**). Algunas alternativas posibles:

##### Versión de 4 líneas:

```
#define RED 6
#define GREEN 3
#define BLUE 5

void setup() {}

void loop() {
    /* Escriba aquí el código */

    // digitalWrite(BLUE, 0); se puede obviar
    for (int i=0; i<=255; i++){
        analogWrite(RED,255-i);
        analogWrite(GREEN, i);
        delay(10000/256);
    } // 10000/256ms por iteración x 256 its = 10s
}
```

##### Versión de 7 líneas:

```
#define RED 6
#define GREEN 3
#define BLUE 5

void setup() {}

void loop() {
    int redValue = 255;
    int greenValue = 0;

    analogWrite (BLUE, 0);
    for(int i = 0; i < 256; i++) {
        analogWrite (RED , redValue-- );
        analogWrite (GREEN, greenValue++ );
        delay (39); // 256*39ms = 9984ms ~ 10s
    }
}
```

#### 5. Configuración de memoria (0.5 puntos).

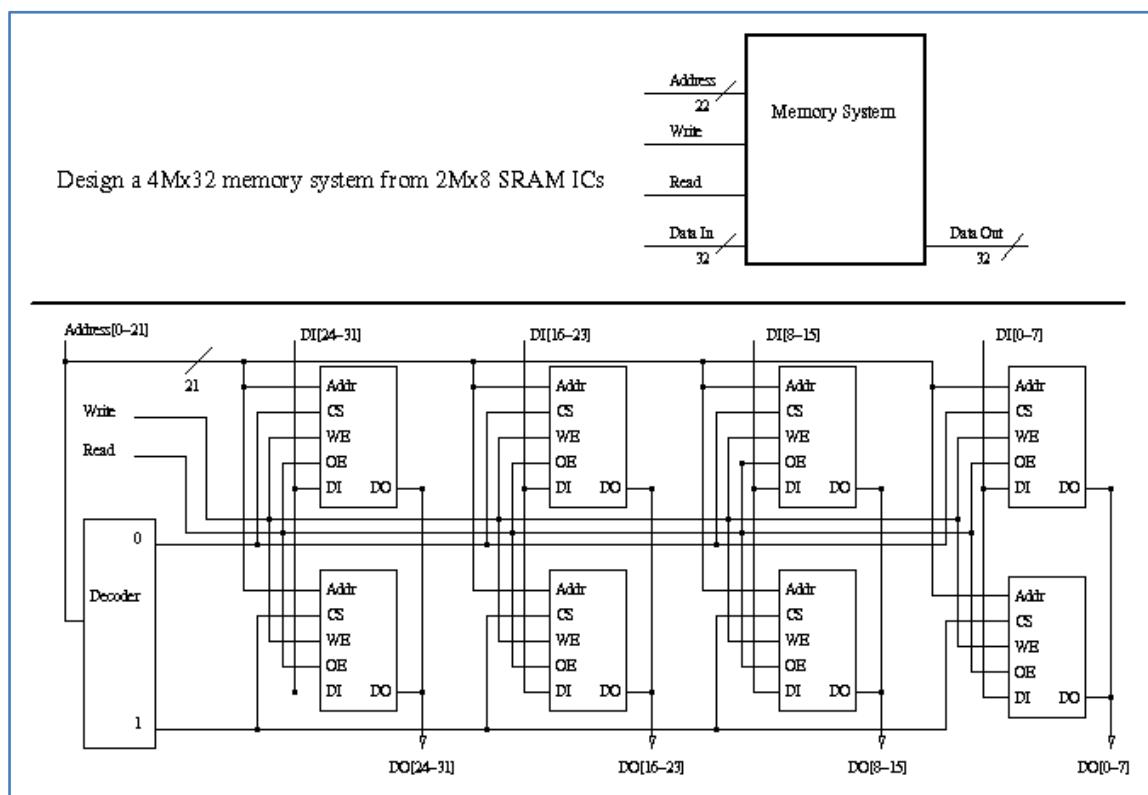
Se puntúa **0.1p** por cada detalle de la lista (total **0.1 x 5 = 0.50p**):

Apartado a): 2x4 chips, conexión R/W, decod.CS y conexión direcciones, conexión datos

Apartado b): los 4 números correctos

Solución:

a)



b)

Fila 0:	Fila 1:
000000	200000
...	...
1FFFFF	3FFFFF

## 6. Cache (0.4 puntos).

Se puntuá **0.1p** por cada número correcto (total **0.1 x 4 = 0.4p**).

Solución:

a)  $2^{36} = 64 \text{ GB}$

b)  $2^6 \text{ B/línea} = 64 \text{ B/línea}$

c) L3:  $12 \text{ MB} = 3 \times 4 \times 2^{20} \text{ B} = 3 \times 2^{22} \text{ B}$

L3:  $3 \times 2^{22} \text{ B} / 2^{14} \text{ conjuntos} = 3 \times 2^8 \text{ B/conjunto} = 768 \text{ B/conjunto}$

d) L3:  $3 \times 2^8 \text{ B/conjunto} / 2^6 \text{ B/línea} = 3 \times 2^2 \text{ líneas/conjunto} = 12 \text{ líneas/conjunto} = 12 \text{ vías}$