





1
Unique
choice

¿Cuál de los siguientes registros tiene que ser salvaguardado (si va a modificarse) dentro de una subrutina según la convención x86-64?

User Teachers





-  a) rax
-  **b) rbx**
-  c) rcx
-  d) rdx

2
Unique
choice

[T2.2.1]

¿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 $EDX * 4 + EBX$?

User Teachers

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



Edit with WPS Office

3
Unique
choice

[P2T]
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:

User Teachers

- ☒ a) `movl lista, %ebx`
- ☐ b) `movl (lista), %ebx`
- ☐ c) `movl $lista, (%ebx)`
- ☐ d) `movl $lista, %ebx`

4
Unique
choice

¿Qué valor contendrá %edx tras
ejecutar las siguientes instrucciones?

`xor %eax, %eax` \Rightarrow `eax = 0`

`sub $1, %eax` \Rightarrow `eax = -1`

`cld` \Rightarrow Extensión de signo

`idiv %eax` \Rightarrow El cociente va a `rax` y el resto
a `rdx`. $-1/-1 = 1 \Rightarrow$ Resto = 0

User Teachers

- ☐ a) 1
- ☐ b) -1
- ☐ c) no puede saberse
con los datos del
enunciado
- ☒ d) 0



Edit with WPS Office

5
Unique
choice

[P2A2]

¿Cuál de los siguientes grupos de instrucciones IA32 sólo modifican los indicadores de estado sin almacenar el resultado de la operación?

User Teachers

- ☐ a) AND, OR, XOR
- ☐ b) ADC, SBB
- ☒ c) CMP, TEST
- ☐ d) IMUL, IDIV

6
Unique
choice

[T2.2.2]

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

User Teachers

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

Porque la orden OR
cambia el bit a 1
solo si uno o los
dos bits son 1

0x20
↳ 00100000
 ↓ ↓
 Bit 5 Bit 0



Edit with WPS Office

7
que
dice

Si RCX vale 0, la instrucción `adc`
`$-1,%rcx`

User Teachers



a) Cambia CF (si valía 0
cambiará a 1, si valía
1 cambiará a 0)



b) Pone CF=0
(independientemente
de lo que valiera
antes)



c) No cambia CF (si
valía 0 permanecerá
a 0, si valía 1
permanecerá a 1)



d) Pone CF=1
(independientemente
de lo que valiera
antes)



8
única
opción

Si el registro RAX contiene X, la
sentencia en C
`x &= 0x1;`
se traducirá a ensamblador como:
User Teachers

- ☒ a) `andq $1, %rax`
- ☐ b) `shrq %rax`
- ☐ c) `orq $0x1, %rax`
- ☐ d) `sarq %rax`

9
única
opción

[P3T]
¿En qué registro se pasa el primer
argumento a una función en Linux gcc
x86-64?

User Teachers

- ☐ a) `ecx`
- ☒ b) `edi`
- ☐ c) `edx`
- ☐ d) `esi`



10

Unique
choice

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
cld
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...

User Teachers

-  ☒ a) produce siempre el resultado correcto
-  b) fallaría con lista: .int -1,-2,-4,-8
-  c) no siempre produce el resultado correcto, pero el error no se manifiesta en los ejemplos propuestos, o se manifiesta en ambos
-  d) fallaría con lista: .int 0,1,2,3



Edit with WPS Office

11

Unique
choice

Dado el siguiente fragmento de programa:

```
.section .data
lista: .int 2,-2,0x10,3,-3
resultado: .quad 0
```

```
.section .text
main: .global main
```

```
xor %rcx,%rcx    rcx = 0
inc %cl
inc %cl          cl = 00000010
shl %cl,%rcx    => Mueve el bit de más a la izquierda en rcx, es decir, 0
mov lista,%ebx
lea (%rbx,%rcx,2),%rdx
```

El valor de %RCX después de la operación LEA es:

User Teachers

- ☐ a) 0x00000008
- ☒ b) Ninguna de las soluciones es correcta
- ☐ c) 0x00000004
- ☐ d) 0x00000002



Edit with Wondershare PDFElement

12
Unique
choice

Dado el siguiente fragmento de programa en ensamblador:

```
.section .data  
lista: .int 1,2,0x10,3  
longlista: .int .-lista  
resultado: .quad 0
```

```
.section .text  
main: .global main
```

```
xor %edx,%edx edx = 0
```

```
mov $-35,%eax
```

```
cld ⇒ Extensión de signo a RDX:RAX
```

```
mov $7,%ebx
```

*10100011
11011100 ⇒ 11011101
↓ ↓
0 0*

```
idiv %ebx
```

...

Al finalizar la ejecución de la instrucción CLTD, los valores de los registros RDX y RAX son:

User Teachers

- ☐ a) Ninguna de las soluciones es correcta
- ☒ b) RDX=0xFFFFFFFF
RAX=0xFFFFFDD
- ☐ c) RDX=0xFFFFFFFF
RAX=0xFFFFF23
- ☐ d) RDX=0xFFFFFFFF
RAX=0x00000023



Edit with WPS Office

13
Unique
choice

Dado el siguiente fragmento de programa:

```
.section .data
lista: .int 1,2,0x10,3
longlista: .int (.-lista)/4
resultado: .quad 0
```

```
.section .text
main: .global main
xor %edx,%edx edx = 0
mov $-23,%eax
cld
mov $5,%ebx
```

```
idiv %ebx
```

...

El valor de %RDX después de la división es:

User Teachers

- ☐ a) 0x00000003
- ☐ b) 0xFFFFFFFFC
- ☒ c) Ninguna de las soluciones es correcta
- ☐ d) 0xFFFFFFFF

$$\begin{array}{r} -23 \overline{) 15} \\ \underline{-3} \\ -4 \end{array}$$

RAX = cociente
RDX = resto

1011
1100
1101 → Byte
menos
significativa



Edit with WPS Office

14
Unique
choice

Dado el siguiente fragmento de programa:

```
.section .data  
lista: .int 1,2,0x10,3  
longlista: .int .-lista 16 tamaño  
resultado: .quad 0
```

```
.section .text  
main: .global main
```

```
xor %edx,%edx  edx = 0  
mov $-17,%eax  
cld  
mov longlista,%ebx
```

```
idiv %ebx
```

$$\begin{array}{r} -17 \quad | \quad 16 \\ \hline -1 \quad -1 \rightarrow RAX \\ \hline \end{array}$$

RAX

El valor de %RAX después de la división es:

User Teachers

☒ a) Ninguna de las soluciones es correcta

☐ b) 0xFFFFFFFFC

☐ c) 0x0000000F

☐ d) 0x00000004

15

Unique
choice

Dado el siguiente fragmento de programa:

```
.section .data
lista: .int 1,2,0x10,3,-3
longlista: .int .-lista 16 tamaño
resultado: .quad 0
```

```
.section .text
main: .global main
```

```
xor %edx,%edx edx = 0
mov $-12,%eax
cld
mov longlista,%ebx
```

```
idiv %ebx
```

El valor de %RDX después de la división es:

User Teachers

- ☐ a) 0x00000004
- ☐ b) 0x00000010
- ☐ c) Ninguna de las soluciones es correcta

- ☒ d) 0xFFFFFFFF4

Resto = -12
Signo
 1100
 10011
 10100 ⇒ El último byte es un 4



Edit with Wondershare PDFElement

16
Unique
choice

Dado el siguiente fragmento de programa:

```
.section .data  
lista: .int 1,2,0x10,3  
longlista: .int .-lista 16 tamaño  
resultado: .quad 0
```

```
.section .text  
main: .global main
```

```
xor %edx,%edx  
mov $-17,%eax  
cld  
mov longlista,%ebx
```

```
idiv %ebx
```

$ebx = 16 \Rightarrow \underbrace{0001}_{1} \underbrace{0000}_{0}$

El valor de %RBX después de la división es:

User Teachers

- ☐ a) 0x00000004
- ☒ b) 0x00000010
- ☐ c) 0x0000000F
- ☐ d) Ninguna de las



Edit with WPS Office

soluciones es
correcta

17
Unique
choice

Dado el siguiente fragmento de programa:

```
.section .data  
lista: .int 1,2,0x10,3  
longlista: .int (.-lista)/4 4 tamaño  
resultado: .quad 0
```

```
.section .text  
main: .global main
```

```
xor %edx,%edx  edx = 0  
mov $-35,%eax  
cld  
mov $7,%ebx  
idiv %ebx
```

...

El valor de %RAX después de la división es:

User Teachers

- ☐ a) 0xFFFFFFFFFC
- ☐ b) 0x00000005
- ☒ c) 0xFFFFFFFFFB
- ☐ d) 0xFFFFFFFFFA



Edit with WPS Office

→ Cociente = -5 → 1101 → 1010 → 1011
8