



Instrucciones generales para la realización de este examen

La respuesta debe escribirse en el hueco existente a continuación de cada pregunta **con letra clara**.

Cada respuesta correcta suma un punto Cada respuesta incorrecta, ilegible o vacía no suma ni resta. El total de puntos se dividirá entre el total de preguntas y se multiplicará por 10 para obtener la nota del examen.

Se ha creado un programa en ensamblador para los procesadores Intel x86 que es capaz de separar una lista de números enteros en dos listas con los elementos que ocupan la posición par e impar de la lista original. La lista origen será de tamaño **byte**, la lista par será de tamaño **doble byte**, y la lista impar será de tipo **doble palabra**.

El programa principal invocará al procedimiento ‘ParImpar’, al cual le pasará **cuatro parámetros** a través de la pila:

- La dirección de la lista de datos origen.
- La dirección de la lista_orden par.
- La dirección de la lista_orden impar.
- El número de elementos de la lista origen.

```
.386
.MODEL FLAT

EXTERN ExitProcess:PROC

.DATA
lista_origen DB 1,2,-3,-4,5,6,-7,-8
lista_orden_par DW 8 DUP(1)
lista_orden_impar DD 8 DUP(2)

.CODE
inicio:
    PUSH OFFSET lista_origen
    PUSH OFFSET lista_orden_par
    PUSH OFFSET lista_orden_impar
    PUSH 8

    CALL ParImpar
```

```
PUSH 0
CALL ExitProcess
NOP

ParImpar PROC
    PUSH EBP
    MOV EBP, ESP

    ;Almacenamos registros que usaremos
    PUSH EAX
    PUSH EDI
    PUSH ESI
    PUSH ECX
    PUSH EBX

    ;Recuperamos los parámetros
    ;nº elementos en ECX
    ;lista origen en EAX
    ;lista_orden_impar en ESI
    ;lista_orden_par en EDI

    ----- HUECO 1 -----

    ;usamos EBX para alternar posiciones
    ;par(EBX==1),impar (EBX==0) y como v.
    ;temporal.

    XOR EBX,EBX
bucle:
    CMP EBX,0
    JE SHORT impar

    ;es par
    MOVSX EBX,BYTE PTR [EAX]

    ;escribimos en la lista par y avanzamos
    ;por la lista par

    ----- HUECO 2 -----

    ;Marcamos el próximo como impar

    ----- HUECO 3 -----

    impar:
        MOVSX EBX,BYTE PTR [EAX]

    ;escribimos en la lista impar y
    ;avanzamos ;por la lista impar

    ----- HUECO 4 -----
```

```
;Marcamos el próximo como par

    MOV EBX,1

fin_impar:

    INC EAX
    LOOP bucle

    ;Recuperamos de la pila los registro
    ;usados

    POP EBX
    POP ECX
    POP ESI
    POP EDI
    POP EAX

    POP EBP

    ----- HUECO 5 -----

END inicio
```

– ¿Qué instrucción o instrucciones faltan en el **HUECO 1**?

```
MOV EAX, [EBP + 20]
MOV EDI, [EBP + 16]
MOV ESI, [EBP + 12]
MOV ECX, [EBP + 8]
```

– ¿Qué instrucción o instrucciones faltan en el **HUECO 2**?

```
MOV [EDI],BX
ADD EDI,2
```

– ¿Qué instrucción o instrucciones faltan en el **HUECO 3**?

```
XOR EBX,EBX
JMP SHORT fin_impar
```



- ¿Qué instrucción o instrucciones faltan en el **HUECO 4**?

```
MOV [ESI],EBX
ADD ESI,4
```

- ¿Qué instrucción o instrucciones faltan en el **HUECO 5**?

```
RET 16
ParImpar ENDP
```

- Cuál es el valor inicial del puntero de la pila?
Contestar en hexadecimal.

0012FFC4

- ¿Cuántos bytes se almacenan en la pila en el momento en el que más información contiene?.
Responder en decimal.

44

- ¿Sobre qué posición de la memoria se encuentra la etiqueta lista_origen? **Contestar en hexadecimal.**

00402000h

```

:00400FFE 0000          add    [eax],al      eax 00000000  c=0
:00401000 6800204000    ◆ PUSH OFFSET lista_origen  ebx 7FFDF000  z=1
:00401005 6808204000    ◆ PUSH OFFSET lista_orden_par  ecx 00000101  s=0
:0040100A 6818204000    ◆ PUSH OFFSET lista_orden_impar  edx FFFFFFFF  o=0
:0040100F 6A08          ◆ PUSH 8             esi 0012F1C0  p=1
:00401011 E808000000    ◆ CALL ParImpar        edi 0012EF88  a=0
:00401016 6A00          ◆ PUSH 0             ebp 0012FFF0  i=1
:00401018 E842000000    ◆ CALL ExitProcess      d=0
:0040101D 90            ◆ NOP
#examen_junio2006#ParImpar
:0040101E 55            ◆ PUSH EBP
:0040101F 8BEC          ◆ MOV EBP, ESP
:00401021 50            ◆ PUSH EAX            ss 0023
:00401022 57            ◆ PUSH EDI            cs 001B
:00401023 56            ◆ PUSH ESI
:00401024 51            ◆ PUSH ECX
:00401025 53            ◆ PUSH EBX
:00401FD8 00 00 00 00 00 00 00 00
:00401FE0 00 00 00 00 00 00 00 00
:00401FE8 00 00 00 00 00 00 00 00
:00401FF0 00 00 00 00 00 00 00 00
:00401FF8 00 00 00 00 00 00 00 00
:0012FFC8 0012EF88
:0012FFC4 79478989
:0012FFC0 00402000
:0012FFBC 00402008
:0012FFB8 00402018
:0012FFB4 00000008
:0012FFB0 00401016
```


Se pretende desarrollar un sistema automático para detectar sin errores si un balón ha traspasado en más de la mitad de su volumen, la línea de gol de una portería de fútbol y que pueda reflejarlo directamente en el marcador como gol válido. Con este sistema se evitarían polémicas y los llamados “goles fantasma”.

El sistema se realizará mediante un nuevo periférico el cual estará compuesto por distintos sensores colocados en ambas líneas de portería, que se conectarán a un modelo de CPU basado en la CPU teórica que hemos visto en clase. Un primer prototipo se probará durante las semifinales del mundial de Alemania 2006. El periférico generará una interrupción asociada al vector 16 cada vez que se produce un gol. La CPU atiende a las peticiones de interrupción por medio de la rutina de servicio Marcador.

El interfaz del periférico consta de tres registros, uno de control y dos de datos (RegP1, RegP2) mapeados en este orden a partir de la dirección 8000 h. Dichos registros realizarán las siguientes funciones:

-
- Registro de Control: el bit menos significativo activado (con el valor 1) de este registro indica que el gol se ha producido en la Portería1. El bit más significativo (con el valor 1) de este registro indica que el gol se ha producido en la Portería2.
- Registro RegP1: almacena el número de tantos (goles) en la portería denominada Portería 1.
- Registro RegP2: almacena el número de tantos (goles) en la portería denominada Portería 2.

Cuando se produce una interrupción, la rutina de tratamiento comprueba, en primer lugar, donde se ha producido el tanto y a continuación en función de esta información actualiza uno de los registros (RegP1, RegP2).

```
ORIGIN 500h
INICIO ini
.PILA 7h
.CODIGO

ini:
;Instalar la rutina de tratamiento de la
;interrupción
----- HUECO 1 -----
; activar interrupciones
```

```
STI
JMP -1

PROCEDIMIENTO Marcador
PUSH R0
PUSH R1
PUSH R2
; leer registro de control
MOVL R0, 00h
MOVH R0, 80h
MOV R1, [R0]
; si el gol no se ha producido en la
Porteria 1 entonces ir Porteria_2 y
actualizar número de tantos
----- HUECO 2 -----

Porteria_1:
INC R0
; actualizar el número de tantos
Porteria 1
----- HUECO 3 -----
JMP actualizado
Porteria_2:
INC R0
INC R0
;actualizar el número de tantos
----- HUECO 4 -----
actualizado:
; desapilar y salir de la rutina
----- HUECO 5 -----
FINP
```

– ¿Qué instrucción o instrucciones faltan en el **HUECO 1?**

```
MOVL R0, BYTEBAJO DIRECCION Marcador
MOVH R0, BYTEALTO DIRECCION Marcador
MOVL R1,10h
MOVH R1,00h
MOV [R1], R0
```

– ¿Qué instrucción o instrucciones faltan en el **HUECO 2?**

```
MOVH R2, 00h
MOVL R2, 01h
AND R2, R2, R1
BRZ Porteria_2
```

– ¿Qué instrucción o instrucciones faltan en el **HUECO 3?** **MOV R2,[R0]**

```
INC R2
MOV [R0],R2
```

– ¿Qué instrucción o instrucciones faltan en el **HUECO 4?**

```
MOV R2, [R0]
INC R2
MOV [R0],R2
```

– ¿Qué instrucción o instrucciones faltan en el **HUECO 5?**

```
POP R2
POP R1
POP R0
IRET
```

– Justo antes de ejecutarse la instrucción **MOV R1, [R0]**, parte del contenido de la pila era el siguiente (de direcciones de memoria mayores a menores terminando en el valor de [R7]):0001h, 0507h, 0507h , 0010h, 0000h. ¿Cuál era el valor del registro R0 en el programa principal justo antes de saltar al procedimiento?. **Contestar en hexadecimal.**

0507h

– Las direcciones del E.D. no ocupadas por el interfaz se pretende cubrir utilizando únicamente módulos de memoria de 16k. ¿Qué rango o rangos de direcciones ocupará el sistema de memoria?. Ejemplo 0000h-1000h.

0000h-7FFFh y C000h-FFFFh

– Se pretende optimizar el tamaño de la pila de tal manera que ocupe el mínimo espacio posible ¿cuál debe ser el valor mínimo a colocar en la directiva **.PILA** de tal manera que el programa funcione correctamente? **Contestar en hexadecimal.**

5h