Apellidos _____ **Nombre** _____ **DNI** _____ **A**

Examen de Fundamentos de los Computadores. *Área de Arquitectura y Tecnología de Computadores* **Septiembre: 5-09-2006**

A 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.

El último modelo de una conocida marca de automóviles ha decidido implementar un regulador de velocidad sobre un computador basado en la CPU de ejemplo. El regulador consta de tres botones, '+', '-' y 'pausa' y un interruptor on-off . El interruptor sirve para conectar / desconectar el regulador. Si se presiona '+' se aumenta la velocidad del coche y si se presiona '-' se reduce la velocidad. En caso de presionar el botón de pausa, si el regulador esta en funcionamiento, el vehículo pasa a "modo manual". Si el vehículo está en "modo manual", al pulsar pausa, el vehículo pasa a "modo regulador" adquiriendo la velocidad que tenía la última vez que pasó a "modo manual".

Para ello a ésta CPU se le conecta un interfaz de E/S, que estará mapeado en las **direcciones de memoria más bajas posibles**, con las siguientes características:

- Un registro de datos en el que se mantiene la última velocidad registrada codificada en binario natural.
- Un registro de estado / control. Los bits de este registro se utilizan para lo siguiente:
  1. Bit 0: si esta activo, indica que el regulador está conectado, en caso contrario desconectado.
  2. Bit 1: si esta activo, significa que el botón '+' se ha pulsado.
  3. Bit 7: si está activo indica que el botón '-' se ha pulsado
  4. Bit 12: si está activo indica que el botón pausa se ha pulsado y se pasa de modo regulador a modo manual.
  5. Bit 14: si está activo indica que el botón pausa se ha pulsado y se pasa de modo manual a modo regulador.

Todas las acciones del regulador son procesadas mediante interrupciones. A continuación se muestra de forma parcial el código del programa principal y de la rutina de tratamiento al pulsar el botón "pausa".

```
;Carga en diversos vectores varias rutinas de interrupción para '+', '-', ...
 [...]
;Se carga en vector 19 la rutina de interrupción para atender a la pulsación del botón pausa
                --- 1 ---

;Bucle infinito para esperar por interrupción
bucle:
    JMP bucle

PROCEDIMIENTO rut_pausa

PUSH R0
PUSH R1
PUSH R2
PUSH R3
PUSH R4

;Se carga en R0 la dirección del registro de estado/control

MOVL R1, -XX-
MOVH R1, -XX-
MOV  R0, R1

;Si lo que se hace es pasar a modo manual
MOVL R2, 01h
MOVH R2, 10h
MOV R3, [R0]
SUB R2, R3, R2
BRNZ comprueba

;Llamar al proced. manual y salir de la rutina
CALL manual;
JMP fin_rut

; Si lo que se hace es pasar a modo regulador
comprueba:
                --- 2 ---
;  Si no es ninguno de los dos casos anteriores, salir de la rutina
BRNZ fin_rut
```

```
;Apilar última velocidad registrada
                --- 3 ---
CALL automatico
POP R4

fin_rut:
POP R4
POP R3
POP R2
POP R1
POP R0

IRET
FINP
```

– ¿En qué dirección se encuentra mapeado el registro de estado/control de la interfaz (lo que falta en **–XX–**)?

0101h

– ¿Qué instrucción o instrucciones falta/n en **--- 1 ---**?

```
MOVL R0, 13h
MOVH R0, 00h
MOVL R1, BYTEBAJO DIRECCION rut_pausa
MOVH R1, BYTEALTO DIRECCION rut_pausa
MOV [R0], R1
STI
```

– ¿Qué instrucción o instrucciones falta/n en **--- 2 ---**?

```
MOVL R2, 01h
MOVH R2, 40h
MOV R3, [R0]
SUB R2, R3, R2
```

– ¿Qué instrucción o instrucciones falta/n en **--- 3 ---**?

```
MOVL R3, 00h
MOVH R3, 01h
MOV R4, [R3]
PUSH R4
```

En un momento dado de la ejecución del bucle infinito del programa anterior se recibe una interrupción debido a una pulsación en el botón '+'.

En la fase de aceptación de esa interrupción y antes de que se ejecutara ninguna instrucción de la rutina de tratamiento de la interrupción, aparecieron, **no necesariamente en ese orden**, en el bus de direcciones del sistema los siguientes valores: 3B64h, 000Ah, 3B63h. Durante la misma fase, en el bus de datos del sistema aparecieron, por orden, los siguientes datos: 0001h, 1A58h, 000Ah, 7A20h.

Con estos datos, contestar a las siguientes preguntas:

- ¿En que dirección de memoria se encuentra la rutina de tratamiento para la interrupción generada por la pulsación de '+'?

7A20

- ¿Cuál será el valor del registro R7 al inicio de la ejecución de la instrucción JMP bucle?

3B65

- ¿Cuál es el número de interrupción que se ha generado? Contestar en decimal.

10

- ¿Cuál es el valor de los flags del registro de estado Z y S?

Z= 0 S= 0

- En una ALU de 8 bits se introducen los siguientes valores y se realiza una **SUMA**:
  - En el **operando A** el número -10d codificado en exceso Z=15.
  - En el **operando B** los 8 bits más altos del número -7,125d codificado en IEE-754.

¿Cuál es el valor obtenido en **la salida en decimal interpretando el resultado como Complemento a 2**?.

Resultado: -59d

- Durante la ejecución de una instrucción de un programa en la CPU teórica se produce una petición de interrupción de un periférico la cual es aceptada. Durante la fase de desencadenamiento (antes que se ejecute ninguna instrucción de la rutina asociada a la interrupción) aparecen en la pila los siguientes datos ordenados de izquierda -el más antiguo-, a derecha -el más reciente- terminado en el dato apuntado por R7 ([R7]): E001h, C001h, 0002h, 0300 h. ¿Cuál era la dirección de la instrucción que se estaba ejecutando en el programa cuando se produjo la interrupción?. (responder en la forma XXXX h)

02FFh

- En un computador similar a la CPU teórica, se pretende reservar la mitad del espacio direccionable, correspondiente con las direcciones más bajas, para dispositivos de Entrada/Salida. Asimismo a partir de la dirección de memoria 8000h se mapea un dispositivo de memoria RAM de tamaño de 24K. Para finalizar, se pretende añadir al computador un dispositivo de memoria ROM que cubra todo el espacio direccionable libre. ¿Cuál es el rango de direcciones que puede ocupar dicho dispositivo?.(responder en la forma XXXX h-YYYY h).

E000 h – FFFF h

- Se pretende diseñar dicho dispositivo de memoria a partir de chips de 2Kx4. ¿Qué tipo de decodificador sería necesario en dicho dispositivo de memoria?. (responder en la forma X/Y siendo X entradas Y salidas).

2/4

- Se pretende ampliar el conjunto de instrucciones de la CPU teórica con una nueva instrucción que permita decrementar en una unidad el contenido de la posición de memoria apuntada por Ri. El mnemónico de la nueva instrucción es:

DEC [Ri]

Completar la siguiente tabla con las señales de control necesarias en cada paso para implementar esta instrucción.

4	Ri_1-IB, IB-MAR, LEER
5	Ciclo de espera
6	MDR-IB, TMPE-SET, ADD, ALU-TMPS, ALU-SR
7	TMPS-IB, IB-MDR, ESCRIBIR
8	FIN

— ¿Cuáles de las siguientes afirmaciones son CIERTAS? (Puedes responder "ninguna" si así lo consideras.)

a) Con n bits no es posible representar la misma cantidad de números enteros utilizado el convenio signo-magnitud que utilizando el convenio complemento a dos.
b) En un circuito secuencial la salida depende en todo momento de los valores que adopten las variables de entrada.
c) La codificación F507 corresponde a la instrucción BRZ 7.
d) Un compilador permite traducir un programa escrito en un lenguaje de programación a código máquina.

A, D

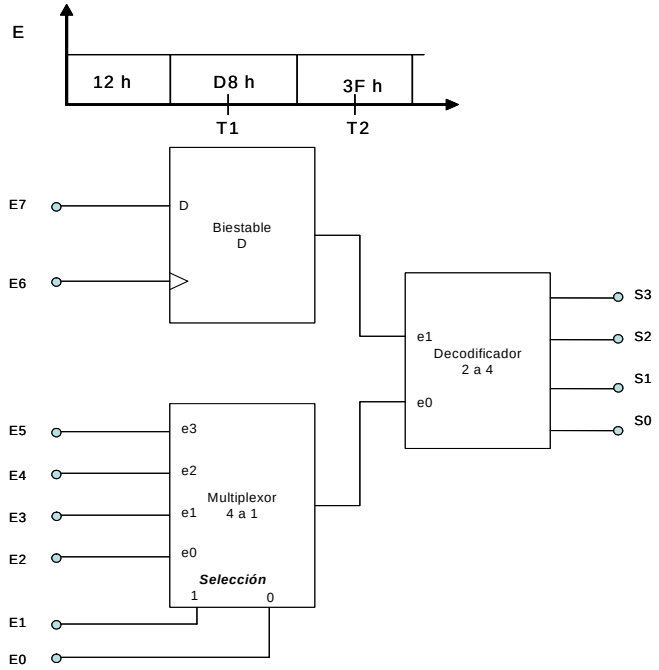
— ¿Cuáles de las siguientes afirmaciones son **CIERTAS**? (puedes responder "ninguna" si así lo consideras)

a) El código ASCII estándar permite representar 256 caracteres.
b) La dirección de memoria 0100h corresponde con un vector de la tabla de vectores de interrupción.
c) En Intel la instrucción **ADD ESP, 4** es equivalente a **RET 12**.
d) En la CPU teórica se pueden direccionar datos de **memoria** del mismo números de bits que es posible direccionar en Intel.

Ninguna

– El circuito de la figura representa un circuito secuencial de 8 entradas y 4 salidas. En el instante t=0 todas las salidas Si se encuentran a 0. A partir de ese instante las entradas evolucionan según el cronograma que se adjunta. ¿Qué valor tendrá la salida en los instantes T1 y T2?

T1:04h T2: 08h



Se ha creado un programa en ensamblador para los procesadores Intel x86 que es capaz de separar una lista de números enteros en otra lista en la que estarán los números positivos de la lista original invirtiendo el orden en el que aparecían inicialmente.

Así si la *lista_origen* es 1, 2, -3, -4, 5, 6,-7, -8, la lista resultado *lst_orden_inverso_negativo* será -8, -7, -4,-3

La lista origen será de tamaño **byte** y la lista resultado será de tamaño **palabra**.

El programa principal invocará al procedimiento '`InvierteOrden`', al cual le pasará **tres parámetros** a través de la pila:

- La dirección de la lista de datos origen.
- La dirección de la lst_orden_inverso_negativo.
- 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
lst_orden_inverso_negativo    DW 8 DUP(0)


.CODE

inicio:
   PUSH OFFSET lista_origen
   PUSH OFFSET lst_orden_inverso_negativo
   PUSH 8

    CALL InvierteOrden


    PUSH 0
    CALL ExitProcess
    NOP


InvierteOrden PROC
    PUSH EBP
    MOV EBP, ESP

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

    ;Recuperamos los parámetros
         ;lista origen en EAX
         ;lista_orden_inverso en EDI
         ;nº elementos en ECX
------- HUECO 1 -------
    MOV EAX, [EBP + 16]
    MOV EDI, [EBP + 12]
    MOV ECX, [EBP + 8]
```

```
    ;Nos pondremos al final de la
;lista_origen para recorrerla en sentido
;contrario
    ------- HUECO 2 -------
    DEC ECX
    ADD EAX,ECX
    INC ECX

    XOR EBX,EBX
    ;EBX irá conteniendo cada número de ;la
lista origen

bucle:

    ;Leemos el numero y comprobamos si ;es
negativo
    MOV BL,[EAX]
    ------- HUECO 3 -------
    CMP BL,0
    JG  SHORT positivo

    ;es negativo lo copiamos en la lista
;invertida
    ------- HUECO 4 -------
    MOVSX BX,BL
    MOV [EDI],BX
    ADD EDI,2

positivo:

    ;pasamos al siguiente numero de la
;lista origen y finalizamos el bucle
    DEC EAX
    LOOP bucle

    ;Recuperamos de la pila los registro
;usados
    POP EDX
    POP EBX
    POP ECX
    POP EDI
    POP EAX
    POP EBP

;Limpiamos los parámetros de la pila
    ------- HUECO 5 -------
    RET 12

InvierteOrden ENDP
END inicio
```

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

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

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

```
DEC ECX
ADD EAX,ECX
INC ECX
```

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

```
CMP BL,0
JG  SHORT positivo
```

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

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

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

```
RET 12
```

– Usando la información que se muestra en el pantallazo de la última figura. ¿Cuál es el valor inicial del puntero de la pila? **Contestar en hexadecimal**.

0013FFC4h

– ¿Cuántos dobles palabras se almacenan en la pila en el momento en el que más información contiene?. Responder en decimal.

10

– Usando la información que se muestra en el pantallazo de la última figura. ¿Sobre qué posiciones de memoria se encuentra el valor -3 en `lista`_origen y en la lista *lst_orden_inverso_negativo* al terminar la ejecución del programa? **Contestar en hexadecimal.**

Lista_origen: 00402002h

Lst_orden_inverso_negativa: 0040200Eh