



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.

A continuación se muestra el listado de un programa cuyo cometido es procesar una cadena de caracteres. El procesamiento de la cadena es realizado por un procedimiento denominado `convierte`. Este procedimiento recibe dos parámetros a través de la pila. El primer parámetro es la dirección de la cadena a procesar, y el segundo, el código del carácter que actúa como terminador de la cadena. El procedimiento procesa la cadena hasta que alcanza el terminador. El procesamiento realizado por el procedimiento es la conversión de la cadena a letras mayúsculas. Se supone que la cadena original sólo contiene letras minúsculas y el terminador.

En la tabla siguiente se muestran los registros utilizados por el procedimiento, así como el cometido de cada uno de ellos.

R1	Puntero a la cadena de caracteres
R2	Almacena el terminador de cadena
R3	Auxiliar para almacenar temporalmente los caracteres de la cadena
R4	Almacena la máscara
R6	Acceso a los parámetros de la pila

Datos adicionales:

ASCII de la 'A' = 41h

ASCII de la 'a' = 61h

```
ORIGEN 200h
INICIO primera
.PILA 10h
.DATOS
```

```
cadena VALOR "abc$"

.CODIGO

PROCEDIMIENTO convierte
push R6
mov R6, R7
; Salvaguardar registros utilizados
push R1
push R2
push R3
push R4

; Recuperar parametros

inc R6
inc R6
mov R2, [R6]
inc R6
mov R1, [R6]

; Cargar mascara

movl R4, 0DFh
movh R4, 0FFh

; Procesamiento de la cadena
bucle:
mov R3, [R1]
comp R3, R2
brz fuera
and R3, R3, R4

mov [R1], R3
inc R1
jmp bucle

fuera:

; Restaurar registros y retornar
pop R4
pop R3
pop R2
pop R1
pop R6
ret
FINP

primera:
movl R0, BYTEBAJO DIRECCION cadena
movh R0, BYTEALTO DIRECCION cadena
push R0
movl R0, '$'
```

```
movh R0, 0
push R0

call convierte
dec R6
dec R6

jmp -1

FIN
```

— ¿Que instrucciones son necesarias en el cuadro 1? Ten en cuenta el comentario inmediatamente superior.

```
inc R6
inc R6
mov R2, [R6]
inc R6
mov R1, [R6]
```

— ¿Que instrucciones son necesarias en el cuadro 2? Ten en cuenta el comentario inmediatamente superior.

```
movl R4, 0DFh
movh R4, 0FFh
```

— ¿Qué instrucciones son necesarias en el cuadro 3?

```
mov [R1], R3
inc R1
jmp bucle
```

— Se sabe que el código máquina de la instrucción `call convierte` es D0DF. ¿En qué dirección de memoria se encuentra ubicada dicha instrucción?

0224

— Codifica la instrucción marcada con el símbolo ◀◀◀.

2802

La CPU elemental está ejecutando el siguiente fragmento de



código, que no tiene ningún cometido en particular:

```
Alli:
  MOVL R1, 04h
  MOVH R1, 25h
  MOV  R2, R1
  MOV  R3, [R1]
  DEC  R3
  BRZ  Alli
  CALL rutina
```

La unidad de control se halla a punto de comenzar el ciclo 1 de una de estas instrucciones. A partir de este momento, observando el bus interno (IB) vemos aparecer los valores siguientes, en ciclos sucesivos: 2502, 2503, 0A20. A partir de esta información responde a las siguientes preguntas.

— ¿Qué valores aparecerán en el bus interno (IB) en los tres ciclos siguientes?

2504, 2503, 2504

— Codificar la instrucción de salto condicional.

F4FA

— Cuando va a ejecutarse la instrucción de salto condicional, ¿qué valor hay en el registro R3?

92FF

— La codificación de la instrucción CALL del listado anterior es D005, y cuando esta instrucción es ejecutada, IP toma el valor 250C, ¿qué valor tomará IP cuando el procedimiento retorne?

2507

Se sabe que la CPU teórica emplea 5 milisegundos en ejecutar el siguiente fragmento de código:

```
XOR R3, R3, R2
MOV  R2, [R4]
MOVL R3, FFh
MOV  R5, R2
```

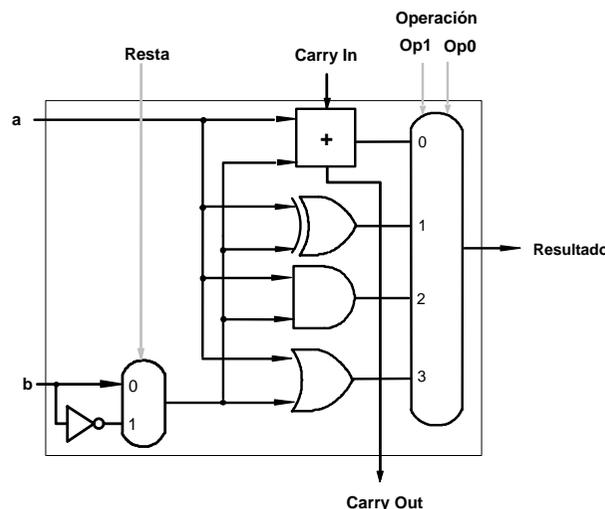
— ¿Cuántos milisegundos dura un ciclo?

0,25

Se quiere implementar una nueva instrucción en la CPU teórica que sume un dato inmediato de 8 bits a un registro. El dato inmediato iría codificado en el byte de menor peso.

— Indicar la secuencia de pasos de control (pasos 4 y posteriores) necesarios para ejecutar ADD R2, Inmd8. Utiliza los pasos que necesites (Nota: necesitarás usar la señal JUMP.)

4	R2-IB,IB-TMPE
5	JUMP, ADD, ALU-TMPS, ALU-SR
6	TMPS-IB, IB-R2,FIN



— Se ha construido una ALU de 8 bits con ALUs de 1 bit como la mostrada en la figura superior. Si el operando A es la cantidad -7 expresada en complemento a 2, el operando B es 43 expresado en signo-magnitud y las señales de control de la ALU tienen el siguiente valor: Op0=1 Op1=0 y, por un error la señal Resta= 1 y Cin=0 ¿Cuál es el resultado generado por la ALU de 8 bits?. Dar el resultado en HEXADECIMAL.

2D

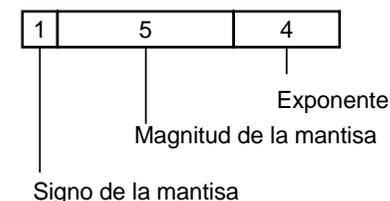
— ¿Cuál es la codificación del mayor número positivo que



se puede representar en el formato IEEE-754, sin considerar el +infinito? Contesta en hexadecimal

7F7F FFFF

A continuación se muestra un formato de coma flotante. La mantisa se normaliza toda fracción, pero el bit más significativo de la mantisa no se representa (es decir, hay bit oculto). El exponente se codifica en exceso a nueve.



— Indica cuáles son el máximo y el mínimo positivos representables en este formato. Indica el máximo en decimal y expresa el mínimo como potencia de dos.

MAX: 63

MIN: 2^{-10}

— Piensa si incrementando el número de bits de la magnitud de la mantisa del formato anterior, sería posible representar el número 51,375 sin cometer error alguno. Si crees que es posible contesta con el número de bits de incremento, en caso contrario contesta IMPOSIBLE.

3

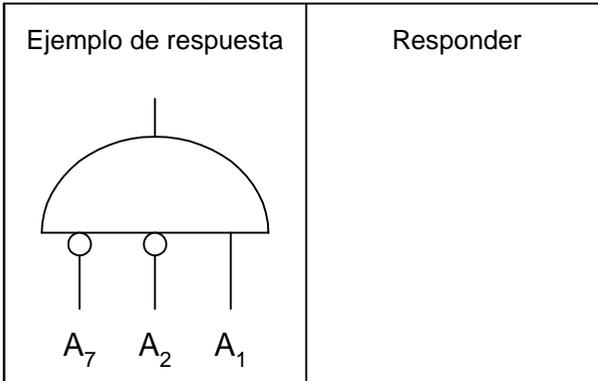
— Se ha diseñado un dispositivo de memoria utilizando un decodificador de tres entradas y varios chips de organización 64Kx8. Además, se sabe que el ancho de la palabra de datos del dispositivo es 32. Determinar la organización del dispositivo (MxN) y el número de chips utilizados en su diseño.

MxN = 512K x 32

Nº chips =32

— Un micro-computador tiene un espacio direccionable de 512 Kilobytes. Se desea instalar en dicho computador un dispositivo de memoria RAM de 32 Kilobytes. Como dato se conoce además que la dirección 693Dh pertenece a dicha

memoria. Dibuja la puerta AND que debe actuar como selector del dispositivo. En el ejemplo de respuesta se muestran tres líneas de dirección, pero puede ser cualquier otro número, indicar su peso y si entran negadas o no a la puerta.



— Se desea diseñar un dispositivo de memoria que ocupe el 25 % del espacio de direcciones del computador elemental. Para ello se utilizan chips que reciben 12 líneas de dirección y 2 de datos. ¿Cuántos chips de este tipo serán necesarios?

32

— Indica cuál o cuáles de las siguientes afirmaciones con CIERTAS. Contesta NINGUNA si crees que ninguna lo es.

- En las operaciones de resta llevadas a cabo por la CPU elemental, cuando el sustraendo es igual al minuendo se activa el bit CF del registro de estado.
- En la CPU elemental, la instrucción INC Rx requiere 6 pasos de ejecución.
- En la CPU elemental, las peticiones de interrupción se detectan en el último paso de la fase de búsqueda e incremento del contador de programa de la instrucción en curso.
- En las operaciones de suma realizadas sobre números expresados en complemento a 2, el carry y el overflow nunca se activan simultáneamente.

NINGUNA

— Define los siguientes conceptos relativos a una unidad de control microprogramada:

Palabra de control: es una palabra formada por tantos bits como señales de control hay en la CPU. Cada bit de la palabra se hace corresponder con una señal de control.

Microprograma: es el conjunto de palabras de control necesarias para ejecutar todas las instrucciones de una CPU.

— Explica brevemente cómo funciona el mecanismo de gestión de prioridad de interrupciones por encadenamiento.

El encadenamiento se realiza sobre la línea INTA.

Las señales de reconocimiento de interrupción generadas por la CPU se propagan a través de la línea INTA, hasta que alcanzan al interfaz que se solicitó la interrupción.

Así el orden de colocación de las interfaces en la cadena INTA determina la prioridad de sus correspondientes periféricos.

Se ha conectado a la CPU teórica un dispositivo lector de tarjetas de acceso que permite controlar la apertura de puertas a edificios. El interfaz de este dispositivo está compuesto por un registro de estado, uno de control y uno de datos, mapeados en este orden a partir de la dirección C400h y todos de tamaño 16 bits. Cuando se pasa una tarjeta por el dispositivo, éste pone un 1 en el bit 3 de su registro de estado

y escribe el número de tarjeta en el registro de datos. Cuando la CPU lee el registro de datos, se pone a 0 el bit 3 del registro de estado. Cuando la CPU escribe un 1 en el bit 0 del registro de control y un 0 en el resto de los bits, el dispositivo abre la puerta. Cuando pone un 1 en el bit 1 del registro de control, el dispositivo no abre la puerta, enciende una luz roja y emite un pitido para avisar al usuario de que se le ha denegado el acceso. Completar el siguiente fragmento de código para que se compruebe si se ha pasado una tarjeta por el dispositivo.

```

MOVl R0, 00h
MOVH R0, 0C4h
MOVl R4, 08h
MOVH R4, 00h
no_hay_tarjeta: MOV R3, [R0]
                AND R3, R3, R4
                BRZ no_hay_tarjeta

```

— Suponiendo que se sabe que se ha pasado una tarjeta, escribir las instrucciones necesarias para leer el número de tarjeta en R5.

```

MOVl R2, 02h
MOVH R2, 0C4h
MOV R5, [R2]

```

— Suponiendo que en R1 está la dirección del registro de control, escribir las instrucciones necesarias para abrir la puerta.

```

MOVl R5, 01h
MOVH R5, 00h
MOV [R1], R5

```