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 pretende crear un videojuego para la CPU teórica, en el que nuestro héroe (representado por el carácter arroba) se enfrenta a una serie de enemigos (representados por otro carácter). De momento, sólo se ha programado el movimiento de nuestro protagonista por cada una de las posiciones de la pantalla. El código ensamblador programado hasta el momento (excepto algunos huecos que faltan y deben ser rellenados) es mostrado a continuación de este enunciado. Antes, se procede a describir cómo funciona el programa.

Inicialmente, el héroe (carácter '@') es mostrado en el centro de la pantalla (fila 3, columna 7; teniendo en cuenta que tanto filas como columnas comienzan a numerarse con el cero). El programa principal está constantemente comprobando la última tecla pulsada (que obtiene de un valor en la sección de datos indicado por la etiqueta *tecla_pulsada*. Según la tecla pulsada, mueve a nuestra arroba una posición en la pantalla: a la izquierda (tecla 'o'), a la derecha (tecla 'p'), hacia arriba (tecla 'q') o hacia abajo (tecla 'a'). Para efectuar el movimiento, una vez detectada la nueva pulsación, llama al procedimiento *mover_personaje* y, tras retornar de éste, actualiza las variables *tecla_pulsada*, poniéndola a cero, y *pos_actual* (con la nueva posición en que se encuentra la arroba).

El procedimiento *mover_personaje* recibe dos parámetros a través de la pila que, por orden de apilado, son:

- Posición de memoria en el Espacio de Direcciones de la CPU en la que está ubicada la posición de memoria de vídeo correspondiente a la posición actual del carácter arroba de la pantalla.
- Desplazamiento a sumar al parámetro anterior para obtener la nueva posición en que imprimir el carácter @.

mover_personaje borrará la arroba de la posición actual de pantalla y la pintará en la nueva posición. Además, retorna a través del registro R1 esa nueva posición. La nueva posición no es simplemente la suma de los dos parámetros recibidos, puesto que, en el caso de que la nueva posición quede fuera de los límites de la memoria de vídeo (fuera de la pantalla), el personaje no será movido.

La variable *tecla_pulsada* será actualizada también por el procedimiento *rut_teclado*, que es la rutina de servicio del periférico teclado. Este procedimiento se encarga de leer una pulsación del buffer de teclado y llevar el valor de 16 bits asociado a la pulsación a la posición de memoria correspondiente en la sección de datos del programa.

```
ORIGEN 0FF00h
.PILA 40h

.DATOS
tecla_pulsada VALOR 0
pos_actual VALOR 0C034h

.CODIGO
; Programa principal
CLI

; Instalar rutina serv. interr. teclado
MOVL R0, 36
MOVH R0, 0
MOVL R1, BYTEBAJO DIRECCION rut_teclado
MOVH R1, BYTEALTO DIRECCION rut_teclado
MOV [R0], R1

; Imprimir el personaje en pantalla
MOVL R0, '@'
MOVH R0, 7
MOVL R1, BYTEBAJO DIRECCION pos_actual
MOVH R1, BYTEALTO DIRECCION pos_actual
MOV R2, [R1]
MOV [R2], R0

; R2 apunta a la variable tecla_pulsada
MOVL R2, BYTEBAJO DIRECCION tecla_pulsada
MOVH R2, BYTEALTO DIRECCION tecla_pulsada
; R3 apunta a la variable pos_actual
MOVL R3, BYTEBAJO DIRECCION pos_actual
MOVH R3, BYTEALTO DIRECCION pos_actual
STI

bucle_impresion:
MOV R4, [R2]
MOVH R4, 0

MOVL R5, 'o'
MOVH R5, 0
COMP R4, R5
BRNZ no_es_o

; si es la 'o', mover 1 pos a la izquierda
MOV R5, [R3]
PUSH R5
XOR R6, R6, R6
DEC R6
PUSH R6
CALL mover_personaje
            -- Hueco 4 --
MOV [R3], R1      ; Actualizar pos_actual
XOR R6, R6, R6
MOV [R2], R6      ; tecla_pulsada = 0
JMP bucle_impresion

no_es_o:
MOVL R5, 'p'
MOVH R5, 0
COMP R4, R5
BRNZ no_es_p

; si es la 'p', mover 1 pos a la derecha
MOV R5, [R3]
PUSH R5
XOR R6, R6, R6
INC R6
PUSH R6
CALL mover_personaje
            -- Hueco 4 --
MOV [R3], R1      ; Actualizar pos_actual
XOR R6, R6, R6
MOV [R2], R6      ; tecla_pulsada = 0
JMP bucle_impresion




no_es_p:
MOVL R5, 'q'
MOVH R5, 0
COMP R4, R5
```

```
BRNZ no_es_q

; si es la 'q', mover 1 pos hacia arriba
MOV R5, [R3]
PUSH R5
MOVL R6, -15
MOVH R6, 0FFh
PUSH R6
CALL mover_personaje
MOV [R3], R1      ; Actualizar pos_actual
XOR R6, R6, R6
MOV [R2], R6      ; tecla_pulsada = 0
```

-- Hueco 4 --

```
JMP bucle_impresion


no_es_q:
MOVL R5, 'a'
MOVH R5, 0
COMP R4, R5
BRNZ bucle_impresion ; no es ‘o’,’p’,’q’ó ’a’

; si es la 'a', mover 1 pos hacia abajo
MOV R5, [R3]
PUSH R5
MOVL R6, 15
MOVH R6, 0
PUSH R6
CALL mover_personaje
MOV [R3], R1      ; Actualizar pos_actual
XOR R6, R6, R6
MOV [R2], R6      ; tecla_pulsada = 0
```

-- Hueco 4 --

```
JMP bucle_impresion
; Fin del programa principal

PROCEDIMIENTO mover_personaje
PUSH R6
MOV R6, R7
PUSH R0
PUSH R2
PUSH R3
PUSH R4
PUSH R5

; Hacer R0 = desplazamiento del personaje
INC R6
INC R6
MOV R0, [R6]

; Hacer R1 = posición actual del personaje
```

-- Hueco 1 --

```
; Copiamos la posición actual a R2
MOV R2, R1

; Sumamos pos. actual y desplazamiento
ADD R1, R1, R0

; R4 apunta a la esquina superior
; izquierda de la pantalla
MOVL R4, 0
MOVH R4, 0C0h

; R5 apunta a la esquina inferior derecha
; de la pantalla
MOVL R5, 77h
MOVH R5, 0C0h

; Si se exceden los limites de la pantalla
; no se debe hacer nada
COMP R1, R4
BRS retorna_pos_vieja
COMP R5, R1
BRS retorna_pos_vieja  ; ***********

; Borrar personaje de posición actual
MOVL R3, ' '
MOVH R3, 0
MOV [R2], R3

; Imprimir personaje en nueva posición
MOVL R3, '@'
MOVH R3, 7
MOV [R1], R3
JMP salir

; Retornar el valor adecuado
retorna_pos_vieja:
MOV R1, R2

salir:
POP R5
POP R4
POP R3
POP R2
```

-- Hueco 2 --

```
FINP



; Rutina serv. interrupc. interfaz teclado
PROCEDIMIENTO rut_teclado
PUSH R0
PUSH R1
MOVL R0, 80h
MOVH R0, 0C0h
MOV R1, [R0]

MOVL R0, BYTEBAJO DIRECCION tecla_pulsada
MOVH R0, BYTEALTO DIRECCION tecla_pulsada
MOV [R0], R1
```

-- Hueco 3 --

```
FINP

FIN
```

– ¿A qué vector de la tabla de vectores de interrupción está asociada la interfaz de teclado? **Contesta en decimal**.

36

– ¿A qué rango de direcciones de memoria responde la interfaz de vídeo? ¿Y la de teclado? **Responde en hexadecimal.** Ejemplo de un rango de direcciones: **00FFh-1245h**. Pista: El número de direcciones a los que responden ambas interfaces es una potencia de 2.

Rango direcciones interfaz de vídeo: C000h - C07Fh

Rango direcciones interfaz de teclado: C080h – C081h

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

INC R6

MOV R1, [R6]

A

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

Examen de Fundamentos de los computadores. *Área de Arquitectura y Tecnología de Computadores* **Convocatoria de Junio: 06-07-2005**

– ¿Qué instrucción o instrucciones falta/n en el **-- Hueco 2 --** y en el **-- Hueco 3 --**?

```
-- Hueco 2 --

POP R0

POP R6RET

-- Hueco 3 --

POP R1

POP R0

IRET
```

– Rellena el espacio de direcciones con dispositivos de memoria para que el programa funcione correctamente. Dispones de dispositivos de tamaño 4K, 8K, 16K, 32K y 64K. El espacio de direcciones deberá estar tan lleno de dispositivos de memoria como sea posible y deberá ser usado el mínimo número posible de éstos. Ten en cuenta el rango de direcciones ocupado por las interfaces de los dispositivos de E/S (pantalla y teclado). **ATENCIÓN: Hay un hueco para usar hasta 8 dispositivos de memoria, pero no quiere decir que no se puedan usar menos. Como ayuda, se ofrece también el diálogo de configuración de memoria del simulador VonNeumann. Ejemplo de tamaño: 4K. Ejemplo de rango: 0000h-7FFFh.**

Disp. 1 → Tamaño: 32K Rango dir: 0000h-7FFFh

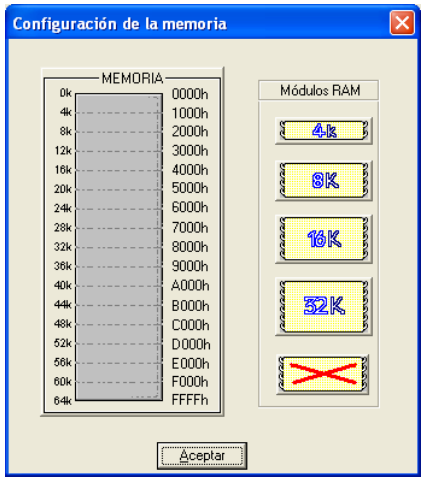
Disp. 2 → Tamaño: 16K Rango dir: 8000h-BFFFh

Disp. 3 → Tamaño: 4K Rango dir: D000h-DFFFh

Disp. 4 → Tamaño: 8K Rango dir: E000h-FFFFh

Disp. 5 → Tamaño: Rango dir:

Disp. 6 → Tamaño: Rango dir:



– Hay varios huecos en el código ensamblador que aparecen como **-- Hueco 4 --**. Esto se debe a que en todos ellos falta/n la/s misma/s instrucción o instrucciones. ¿De qué instrucción o instrucciones se trata?

```
INC R7

INC R7
```

– Como puedes comprobar, el programador ha puesto un tamaño de pila grande para evitar problemas. Pero, ¿cuál sería el tamaño mínimo de pila con el que funcionaría el programa? **Responde en hexadecimal con dos dígitos**.

0Dh

– ¿Cuál es la codificación de la última de las instrucciones **BRS retorna_pos_vieja** del procedimiento *mover_personaje* (la que aparece a la izquierda de un comentario con varios astericos)? **Responde en hexadecimal**.

F607h

– ¿Qué señales están activas en el último paso de ejecución de la última de las instrucciones **BRS retorna_pos_vieja** del procedimiento *mover_personaje* (la que aparece a la izquierda de un comentario con varios astericos), si sabemos que cuando se ejecutó la instrucción anterior R1 valía C070h? ¿De qué número de paso se trata?

Señales activas: FIN

Número de paso: 4

– Justo cuando se está ejecutando el paso 4 de la última instrucción del programa principal (**JMP bucle_impresion**), se pulsa la tecla M del teclado, ¿qué señales de control estarán activadas tres ciclos de reloj más tarde?

R7_IB, TMPE_SET, ADD, ALU_TMPS

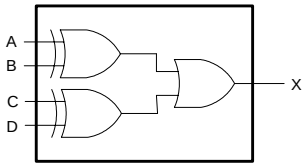
Se sabe que la CPU teórica emplea 0,5 microsegundos en ejecutar el siguiente fragmento de código:

```
MOVH R1, 0ABh
MOVL R1, 0CDh
MOV R2, [R1]
MOV [R2], R1
```

– ¿Qué frecuencia tiene la señal de reloj que llega a la unidad de control de la CPU teórica? Contesta en MHz

40 MHz

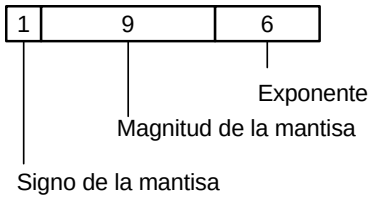
Se pretende rediseñar el circuito de la figura adjunta utilizando un array lógico programable (PLA).



– ¿Cuál es el número mínimo de puertas AND y OR debe tener dicho array lógico programable?

AND: 12 OR: 1

A continuación se muestra un formato de coma flotante. La mantisa se normaliza todo fracción en formato signo-magnitud, y el exponente se codifica en exceso central.



– Indica cuáles son el máximo y el mínimo **positivos** representables en este formato. **Indica el máximo y el mínimo como potencia de dos**.

MAX: $2^{31} - 2^{22}$ MIN: 2^{-33}

— Determina cuál o cuáles de las siguientes afirmaciones son **CIERTAS**. Si ninguna es cierta contesta NINGUNA.

A) Trabajando con 8 bits y exceso a Z = 189, el número 67 no es representable.
B) 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.
C) En la CPU elemental, la instrucción PUSH Rx requiere más pasos de ejecución que la instrucción POP Rx
D) 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.

A, C

En la figura adjunta se muestra el estado de algunos registros y de un conjunto de posiciones de memoria del computador elemental justo cuando acaba de ejecutarse el **paso 5º** de una instrucción.

MEMORIA		REGISTROS	
0810	28C0	PC	0813
0811	20FF		
0812	3000	R0	C0FF
0813	4000	R1	0000
0814	3900	R2	0000
0817	0000		
0818	0000		
0819	0000		
081A	0000	R7	081A

A partir de la información de la figura contesta a las preguntas A, B y C.

— A) ¿cuál será el contenido del registro MDR en el instante representado en la figura?

COFFh o 0819h

— B) Determina el valor de los bits del registro de estado justo después de la ejecución de la instrucción 4000 (almacenada en la dirección 0813).

CF = 1 OF = 0 SF = 1 ZF = 0

— C) ¿Qué valores contendrá el BUS interno durante los tres primeros pasos de ejecución de la instrucción (almacenada en la dirección 0814)? Contesta con 4 dígitos hexadecimales para cada valor del BUS.

Paso1: 0814h

Paso 2: 0815h

Paso 3: 3900h

Se dispone de un computador con un bus de direcciones de 16 líneas y un bus de datos de 8 líneas.

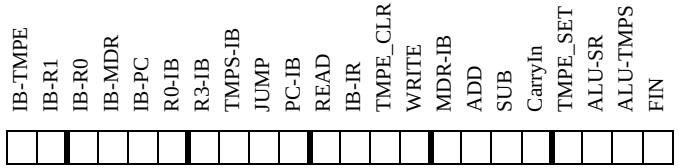
— Se desea conectar al computador indicado un dispositivo de memoria que ocupará el 12,5 % del espacio de direcciones. Para ello se dispone de chips de memoria de 10 líneas de dirección y 2 de datos. ¿Cuántos chips serán necesarios para construir este dispositivo?

32

— Indica el rango de direcciones en el que se ubica el banco 4 del dispositivo de memoria al que se hace referencia en la pregunta anterior, teniendo en cuenta que la dirección A4D6h pertenece a dicho dispositivo de memoria. (Los bancos están numerados como banco 0, banco 1…). (Formato de solución para el rango: XXXXh – YYYYh)

B000h – B3FFh

Una unidad de control microprogramada para la CPU elemental produce palabras de control de 61 bits, cuyos 22 bits inferiores se interpretan en la forma siguiente:



— En un instante dado, la palabra de control toma el valor ....210000h y, en ese mismo instante, IR contiene el valor 400Ch. ¿Cuál será la próxima palabra de control que emitirá la unidad de control? **Expresa el resultado en hexadecimal utilizando 6 cifras.**

……008046h