

APELLIDOS:

NOMBRE:

INSTRUCCIONES: Al comenzar el examen

- Crea una carpeta con tus apellidos, nombre y DNI en tu carpeta habitual de trabajo, de la forma Fernández López José – 00000000T.
- Guarda en esa carpeta todos los archivos que hagas en este examen.

ENUNCIADO

El fichero *bin2hex.ens* contiene el código fuente, incompleto, de un programa que implementa la funcionalidad de convertir números de 16 bits de base binaria a base hexadecimal en el computador teórico. El número binario se introduce en el computador a través del periférico *Luces*. Una vez convertido a hexadecimal, el resultado se muestra en el periférico *Pantalla*. El evento que desencadena la tarea de conversión es una interrupción generada por el periférico *Luces*.

La secuencia de 16 bits inicial dará lugar a 4 dígitos hexadecimales. El algoritmo utilizado para convertir un número de base 2 a base 16 se basa en una técnica de desplazamiento de bits. Lo que se hace es ir obteniendo, a través del bit de carry del registro de estado de la CPU, uno a uno los bits que forman parte de cada cuarteto de la secuencia inicial. Dado que cada vez que se obtiene un bit se conoce su “peso” en el número, en caso de dicho bit sea ‘1’ se suma su “peso” al dígito hexadecimal final.

Para imprimir cada dígito hexadecimal en la pantalla lo que se hace es obtener el código ASCII correspondiente al dígito hexadecimal, diferenciando cuándo se trata de un número entre 0 y 9, o entre 10 y 15 (el código ASCII del carácter ‘0’ es 30h; el código ASCII del carácter ‘A’ es 41h).

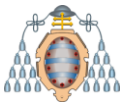
El programa tiene las siguientes características:

- La dirección de la memoria de vídeo en la que se escribirá el siguiente carácter en la pantalla se almacena en la variable *ptr_pantalla*.
- La dirección a partir de la cual se mapea el periférico *Luces* se almacena en la variable *luc*.
- El vector de interrupción asociado al periférico *Luces* es el 255.

El programa consta de los siguientes procedimientos:

- *rut_luc*: Rutina de tratamiento de la interrupción del periférico *Luces*. Lee el estado de los interruptores del periférico *Luces*, actualiza el estado de los leds con la combinación leída e invoca al procedimiento *bin2hex* para convertir la secuencia de bits leída a hexadecimal. A continuación escribe un espacio en blanco en la pantalla para separar el resultado de la conversión actual de posibles resultados posteriores.
- *bin2hex*: Procedimiento que implementa el algoritmo de conversión de base 2 a base 16 descrito anteriormente. Recibe la secuencia de bits a convertir a través de la pila y envía uno a uno el dígito resultante de convertir cada cuarteto de bits a la pantalla.
- *print_dig_hex*: Procedimiento que muestra en la pantalla el carácter correspondiente a un número hexadecimal comprendido entre 0 y 15. El número hexadecimal a representar lo recibe a través de la pila.
- *print_espacio*: Imprime un espacio en blanco en la pantalla.





TAREAS

[1 PUNTO] Completar la instrucción o instrucciones necesarias en el – Hueco 1 --.

[1 PUNTO] Completar la instrucción o instrucciones necesarias en el – Hueco 2 --.

[1 PUNTO] Completar la instrucción o instrucciones necesarias en el – Hueco 3 --.

[1 PUNTO] Completar la instrucción o instrucciones necesarias en el – Hueco 4 --.

[1 PUNTO] Completar la instrucción o instrucciones necesarias en los huecos 5a a 5d. Sustituir las secuencias 'XXX' por el número adecuado en cada hueco.

[1 PUNTO] Completar la instrucción o instrucciones necesarias en el – Hueco 6 --.

[1 PUNTO] Configurar el simulador del computador elemental con la cantidad mínima de memoria RAM necesaria para ejecutar correctamente el programa. Conectar el periférico Luces y el periférico Pantalla. Ensamblar el programa y cargarlo en el simulador. En este punto, guardar el estado de la simulación en un fichero de nombre *bin2hex.sim*.

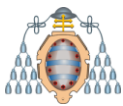
[0.5 PUNTOS] Seleccionar mediante los interruptores del periférico *Luces* la secuencia binaria 1100 1010 1111 1110. Generar una interrupción a partir de dicho periférico. ¿Qué valor se muestra en la pantalla?

[2.5 PUNTOS] Tal cual está implementado el programa, solo se puede convertir de base 2 a base 16 una cantidad limitada de números, hasta que la pantalla se llene. Se pide implementar la funcionalidad que permita determinar cuándo se ha llegado al final de la pantalla y, en ese caso, borrar el contenido de la misma para comenzar a escribir desde la esquina superior izquierda. Para ello, se deberían de sustituir los grupos de instrucciones en el programa cuyos comentarios comienzan con el símbolo '*' por la invocación a un procedimiento que realice la tarea de incrementar el puntero de pantalla, comprobar si se ha llegado al final de la misma (es conveniente declarar una nueva variable para realizar esta tarea), y borrar la pantalla y colocar el puntero al inicio de la misma en caso necesario. El nombre del nuevo procedimiento será *actualiza_ptr_pantalla*.

INSTRUCCIONES Al finalizar el examen

- Comprueba que en la carpeta con tu nombre están los ficheros *bin2hex.ens* y *bin2hex.sim*.
- Desde el Explorador de Windows pulsa con el botón derecho sobre la carpeta con tu nombre y elige la opción para comprimirla a un archivo .zip.
- En la barra de direcciones del Explorador de Windows escribir <ftp://156.35.151.58:31511>
En nombre de usuario escribir 'alumno' y como contraseña 'mayo2fc'.
- Copia en la carpeta que se ha abierto el fichero .zip con tu examen. Cerciórate de que es el fichero correcto porque una vez copiado no lo puedes modificar.





```

ORIGEN 1000
INICIO main
.PILA 20h

.DATOS
ptr_pantalla VALOR 0F000h
luces VALOR 0E000h

.CODIGO
main: MOVL R0, BYTEBAJO DIRECCION rut_luces
      MOVH R0, BYTEALTO DIRECCION rut_luces
      -- Hueco 1 -- ; Instalar rutina interrupción

      JMP -1

PROCEDIMIENTO rut_luces
      PUSH R0
      PUSH R1
      MOVL R0, BYTEBAJO DIRECCION luces
      MOVH R0, BYTEALTO DIRECCION luces
      MOV R0, [R0]
      MOV R1, [R0]
      MOV [R0], R1
      -- Hueco 2 -- ; Invocar a bin2hex

      CALL print_espacio
      -- Hueco 3 --- ; Retornar de la rutina

FINP

PROCEDIMIENTO bin2hex
      PUSH R6
      MOV R6, R7
      PUSH R0
      PUSH R1
      PUSH R2
      PUSH R3
      -- Hueco 4 --- ; R0 = núm binario a convertir

      XOR R1, R1, R1
      XOR R2, R2, R2
      MOVL R3, 04h ; num cuartetos en 16 bits
      MOVH R3, 00h

bucle_bin2hex: ; para cada cuarteto...
      ADD R0, R0, R0
      BRNC dig_peso_2
      -- Hueco 5a --
      MOVL R1, XXX ; dig peso 3 del cuarteto a 1

dig_peso_2:
      ADD R0, R0, R0
      BRNC dig_peso_1
      -- Hueco 5b --
      MOVL R2, XXX; dig peso 2 del cuarteto a 1
      ADD R1, R1, R2

dig_peso_1:
      ADD R0, R0, R0
      BRNC dig_peso_0
      -- Hueco 5c --
      MOVL R2, XXX; dig peso 1 del cuarteto a 1
      ADD R1, R1, R2

dig_peso_0:
      ADD R0, R0, R0
      BRNC 2
      -- Hueco 5d --
      MOVL R2, XXX; dig peso 0 del cuarteto a 1
      ADD R1, R1, R2
      PUSH R1
      CALL print_dig_hex
      INC R7
      XOR R1, R1, R1
      -- Hueco 6 --

      POP R3
      POP R2
      POP R1
      POP R0
      POP R6
      RET

```

```

PROCEDIMIENTO print_dig_hex
      PUSH R6
      MOV R6, R7
      PUSH R0
      PUSH R1
      PUSH R2
      PUSH R3
      INC R6
      INC R6
      MOV R1, [R6] ; cuarteto de bits
      MOVL R2, 09h
      MOVH R2, 00h
      COMP R2, R1
      BRC es_letra
      MOVL R2, 30h
      ADD R2, R2, R1 ; sumar ASCII '0'
      JMP print

es_letra:
      MOVL R2, 37h
      ADD R2, R2, R1 ; sumar (ASCII 'A')-10

print:
      MOVL R0, BYTEBAJO DIRECCION ptr_pantalla
      MOVH R0, BYTEALTO DIRECCION ptr_pantalla
      MOV R3, [R0]
      MOVH R2, 07h ; atributos de color
      MOV [R3], R2
      INC R3 ; * actualizar puntero
      MOV [R0], R3 ; * de pantalla
      POP R3
      POP R2
      POP R1
      POP R0
      POP R6
      RET

FINP

PROCEDIMIENTO print_espacio
      PUSH R0
      PUSH R1
      MOVL R0, BYTEBAJO DIRECCION ptr_pantalla
      MOVH R0, BYTEALTO DIRECCION ptr_pantalla
      MOV R1, [R0]
      MOVL R2, ' ' ; espacio en blanco
      MOVH R2, 00h ; atributos de color
      MOV [R1], R2
      INC R1 ; * actualizar puntero
      MOV [R0], R1 ; * de pantalla
      POP R1
      POP R0
      RET

FINP
FIN

```

FINP