1 **(1 puntos)** Indique, justificando la respuesta, si las siguientes afirmaciones son verdaderas o falsas.

a) La unidad de control necesita como entrada el registro de estado y el registro de direcciones de memoria.

Falso. La unidad de control necesita como entrada el registro de estado y el registro de instrucción.

b) La entrada/salida programada permite que la operación en el dispositivo se realice más rápidamente.

Falso. El tiempo de operación del dispositivo solo depende del dispositivo, no de la técnica de E/S utilizada por la CPU.

c) En un computador con modelo de ejecución registro-registro se puede ejecutar `ADD .R1,[.R2 ]`

Falso. Puesto que el objeto del segundo direccionamiento está en memoria, la instrucción aritmética ADD no pertenece al modelo de ejecución registro-registro.

2 **(3 puntos)** Sea un computador de palabra de 32 bits, de dos direcciones, con direccionamiento a nivel de byte, con modelo de ejecución registro-registro cuyos únicos modos de direccionamiento son: inmediato, directo a registro e indirecto a registro. Programe el código equivalente a las instrucciones que se indican y usando, como máximo, tres registros temporales, `T1`, `T2` y `T3`:

a) `ADD [.R3], /2000, [#8[.R7--]]`

```
; M(R3) <--- M(2000) + M(M(R7+8))
; R7 <--- R7 - 4;

LD  .RT1, #2000
LD  .RT1, [.RT1]
LD  .RT2, #8
ADD .RT2, .R7
LD  .RT2, [.RT2]
LD  .RT2, [.RT2]
ADD .RT1, .RT2
ST  .RT1, [.R3]
SUB .R7 , #4
```

b) `MOVE [/1000], #64[--.R4]`

```
; R4 <--- R4 - 4
; M(R4+64) <--- M(M(1000))

SUB .R4 , #4
LD  .RT1, #1000
LD  .RT1, [.RT1]
LD  .RT1, [.RT1]
LD  .RT2, #64
ADD .RT2, .R4
ST  .RT1, [.RT2]
```

c) `CALL[#4[.R7++]]`

```
; Salvar PC
; PC <--- M(R7 + 4)
; R7 <--- R7 + 4

LD  .RT1, #4
ADD .RT1, R7
LD .RT1, [.RT1]
ADD .R7, #4
CALL [.RT1]
```

3 *(**6 puntos**) Una tabla (`STOCK`) contiene elementos que indican las existencias en un almacén de productos. Cada entrada de la tabla contiene dos enteros: el primero es un entero con el código de un producto existente en almacén (Código); el segundo entero es el número de unidades existentes (Existencias). La tabla **no está ordenada**, sus elementos se almacenan en posiciones consecutivas de memoria, y finaliza cuando el campo Código contiene el valor 0 (NULL). Existe otra tabla con la ficha de productos (`PRODUCTOS`) que contiene, por cada código de producto (Código), un valor mínimo de existencias a partir del cual hay que hacer pedido a proveedor (Mínimo), la cantidad de unidades que se solicitan siempre en cada pedido de ese producto (Cantidad), y el precio unitario de cada unidad (Precio). La tabla se supone que **no está ordenada** y finaliza cuando el campo Código contiene el valor 0 (NULL). Véase las tablas al final.*

*Se desea generar una lista en forma tabla (`PEDIDOPROVEEDOR`) que deberá tener una entrada de dos palabras por cada producto que hay que pedir a proveedor. En cada elemento de esta lista deberá estar el Código del producto y su cantidad a pedir en ese pedido. La tabla **no es necesario que esté ordenada** y finaliza cuando el campo código contiene el valor 0.*

Se pide realizar las siguientes subrutinas en ensamblador del MC88110:

***a) (20 %)** `BUSCAR(Código, ListaProductos)` busca en la tabla de fichas de `ListaProductos` un código de producto `Código`. Los parámetros de entrada se pasan en la pila, `Código` por valor, y `ListaProductos` por dirección. La subrutina devuelve en r29 la dirección del elemento buscado o NULL si no se ha encontrado. No es necesario que esta subrutina implemente el marco de pila.*

***b) (10 %)** Indique brevemente qué modificaciones habría que hacer en la subrutina del apartado anterior si la tablas `STOCK` y `PRODUCTOS` estuvieran ordenadas por el campo `Código`.*

***c) (70 %)** `GENERARPEDIDO (Stock, Productos, PedidoProveedor)` genera la lista `PedidoProveedor` a partir de las tablas `Stock` y `Productos`. Todos los parámetros se pasan en la pila por dirección. Esta rutina hará uso de la subrutina `BUSCAR`, y en caso de no encontrar el `Código` en la tabla `Productos`, no incorporará ese `Código` al `PedidoProveedor`.*

Se supondrá que están definidas todas las macros que se han explicado en la parte teórica de la asignatura, que la subrutina llamante deja disponibles todos los registros excepto r1, r30 (SP) y r31 (FP); que la pila crece hacia direcciones de memoria decrecientes y el puntero de pila apunta a la última posición ocupada (de la misma forma que se ha utilizado en la teoría). A modo de ejemplo a continuación se muestran las tablas `STOCK` y `PRODUCTOS` y la lista `PEDIDOPROVEEDOR` que se genera como resultado de la ejecución de la subrutina `GENERARPEDIDO`.

Tabla `STOCK`

Código	*Existencias*
452	*2*
312	*10*
581	*50*
324	*32*
215	*10*
0	*??*

Tabla `PRODUCTOS`

Código	*Mínimo*	*Cantidad*	*Precio*
123	*10*	*50*	*3*
215	*15*	*50*	*4*
312	*5*	*20*	*16*
320	*20*	*80*	*2*
324	*10*	*50*	*15*
452	*5*	*15*	*60*
480	*10*	*40*	*32*
581	*20*	*60*	*12*
0	*??*	*??*	*??*

Tabla `PEDIDOPROVEEDOR`

Código	*Cantidad*
452	*15*
215	*50*
0	*??*

SOLUCIÓN

a) Como no se llama a ninguna subrutina, no es necesario salvar dirección de retorno, y tampoco manejo de marco de pila. El algoritmo para buscar consiste en recorrer la lista con un puntero que se inicializará al parámetro `Productos` hasta fin de lista (NULL o 0). El mejor registro puntero para recorrerla es el propio r29, que también es parámetro de salida. Por cada Código de la lista, hay que comprobar si es igual al buscado, en cuyo caso se ha encontrado. Si se llega al fin de lista es que no se ha encontrado y hay que devolver en r29 un 0. A continuación se muestra el código:

```
BUSCAR:   ld r29, r30, 4        ; r29 contiene el puntero a la tabla Productos
          ld r4, r30, r0          ; r4 contiene el Código a buscar
BUC_BUS:  ld r5, r29, r0         ; si fin de lista, terminar (no encontrado)
          cmp r3, r5, r0
          bb1 eq, r3, NOENC
          cmp r3, r5, r4         ; comparar código de la lista con el que se busca
          bb1 eq, r3, ENC
          add r29,r29,16          ; avanzar puntero las 4 palabras de una entrada de Productos
          br BUC_BUS
NOENC:    add r29,r0,r0          ; Se devuelve no encontrado
ENC:      jmp(r1)                ; Se devuelve la entrada encontrada
```

b) El código anterior funciona correctamente, pero se podría optimizar para el caso de que no se encuentre el Código en la tabla de Productos. Si la lista de Productos está ordenada de menor a mayor, se puede averiguar que un Código no se encuentra antes de llegar al final de la lista. Al comparar el Código buscado con el de la lista, si el de la lista es mayor, es que no se encuentra. Bastaría añadir una sola instrucción tras la comparación de salto condicional a no encontrado en ese caso:

```
          ...
          cmp r3, r5, r4         ; comparar código a buscar con el de la lista
          bb1 gt, r3, NOENC
          bb1 eq, r3, ENC
          ...
```

c) Esta subrutina se resuelve recorriendo la tabla Stock utilizando un puntero (r20). El recorrido de la tabla finalizará cuando se encuentre el valor 0 (NULL) en el campo `Código`. En cada iteración se busca la entrada Código en la lista con las fichas de `Productos` llamando a la subrutina `BUSCAR`. Como hay variables importantes que puede modificar `BUSCAR`, hay que manejar marco de pila, y almacenar dichas variables antes en la zona de variables locales.

Si no está se salta a fin de bucle (avanzar punteros) para iniciar con otro `Código`. Pero si se encuentra, hay que comparar las `Existencias` de ese `Código` con el `Mínimo`. En caso de ser menor las existencias, habrá que escribir una nueva entrada en la tabla de PedidoProveedores, que se irá actualizando según indique otro puntero (r21). Es importante mantener siempre el puntero con el que se recorre la tabla de Stock y el puntero a la tabla PedidoProveedores, por lo que deben estar en variables locales. El Código se puede obtener a partir de conocer el puntero a Stock, por lo que no hace falta almacenarlo.

```
GENPEDIDO:  PUSH(r1)
            PUSH(r31)
            or r31,r30,r30
            subu r30,r30,8
            ;acceso a los parámetros e inicialización de punteros
            ld r20,r31,8  ; puntero a tabla Stock
            ld r21,r31,16  ; puntero a tabla Pedido
            ld r22,r31,12  ; puntero a tabla Productos
bucle:      ld r5,r20,r0 ; r5<-código de Stock
            cmp r7,r5,r0 ; si fin de lista terminar
            bb1 eq,r7,fingen
            ;BUSCAR Código en Productos
            ;guardar variables locales r20,r21
            st r20,r31,-4  ; Puntero a tabla Stock
            st r21,r31,-8  ; Puntero a tabla Pedidos
            PUSH(r22)
            PUSH(r5)
            bsr BUSCAR
            addu r30,r30,8 ; limpiar parámetros de la pila
            ld r20,r31,-4 ; restaurar variables locales
            ld r21,r31,-8
            cmp r3,r29,r0  ; Si buscar no ha tenido éxito
            bb1 ne,r3,otro ; salta a otro
```

```
            ; comparar existencias con mínimo
            ld r6,r20,4    ; Existencias
            ld r7,r29,4    ; Mínimo
            cmp r3,r6,r7  ;
            bb1 gt,r3,otro ; si hay más existencias salta a otro
            ; escribir entrada de ese producto en Pedido
            ld r5,r20,0  ; r5 <- Codigo
            st r5,r21,0
            ld r8,r29,8  ; r8 <- Cantidad
            st r8,r21,4
            addu r21,r21,8
otro:       addu r20,r20,8 ; avanzar punteros
            br bucle
fingen:     st r0, r21, 0 ; escribir fin en Pedidos
            or r30,r31,r31
            POP(r31)
            POP(r1)
            jmp(r1)
```