**ESTRUCTURA DE COMPUTADORES (Grado II)**
**PRIMER PARCIAL (10 de noviembre de 2016)**

---

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

***a)*** *El registro Contador de Programa contiene siempre la instrucción siguiente a la que se esté ejecutando.*

Falso. El contador de programa lo que contiene es la dirección de la siguiente instrucción a ejecutar.

***b)*** *El tiempo de acceso a la memoria se indica por el parámetro MIPS.*

Falso. Los MIPS son Millones de Instrucciones Por Segundo, y son indicativo de la capacidad de proceso de la CPU. El tiempo de acceso a memoria es el tiempo que tarda la memoria en acceder a una palabra máquina. La velocidad de la memoria se puede indicar por su caudal, o cantidad de información accesible por unidad de tiempo (bytes/seg).

**2** **(2 puntos)** *En un computador con palabras y direcciones de 32 bits, con direccionamiento a nivel de byte, se ejecuta el siguiente fragmento de código en el que el tamaño de cada instrucción se indica como comentario. Considere que a partir de la dirección de memoria H'00001000 se encuentran almacenados respectivamente los siguientes datos: H'00000001, H'00000002, H'00000003, H'00000004 y H'00000005. Indique los valores sucesivos que toman los registros y posiciones de memoria afectados por la ejecución.*

```
LD .R1, #H'00001010  ; 2 palabras
LD .R2, #H'0000100C  ; 2 palabras
LD .R3, [--.R1]      ; 1 palabra
SUB #4[.R2--], [.R1] ; 2 palabras
SUB .R3, #2          ; 2 palabras
BNZ $-20             ; 1 palabra
MOVE .R1, .R2        ; 1 palabra
HALT                 ; 1 palabra
```

## SOLUCIÓN

La instrucción BNZ salta cinco palabras anteriores a donde apunta el PC, por lo que saltaría a la primera instrucción SUB. La primera vez que se ejecuta BNZ es tras restar el valor 2 a R3 que tiene un 4 y salta, pero la segunda vez resta 2 a 2 y como el resultado ya es = continua secuencialmente la ejecución. Los valores sucesivos de los registros y posiciones de memoria son los siguientes:

```
R1<- H'1010, <- H'100C
R2<- H'100C, <- H'1008 (salto),  <- H'1004 <- H'100C
R3<- 4,      <- 2      (salto),  <- 0

dir mem: contenido
      1000: H'0001
      1004: H'0002
      1008: H'0003
      100C: H'0004     (salto),  <- H'0000
      1010: H'0005, <- H'0001
```

**3 (7 puntos)** *Se dispone de un texto ASCII almacenado en memoria que finaliza con el carácter NUL (código ASCII 0x00). Cada carácter del texto se representa mediante un entero sin signo de 8 bits. Se desea escribir un programa en ensamblador del 88110 que busque una cadena de caracteres en dicho texto y determine en qué posiciones se encuentra.*

*Se supondrá que existe la función de librería* `strlen(str)` *que devuelve en r29 la longitud de la cadena de caracteres* `str` *que finaliza con el carácter NUL (excluyendo dicho carácter). El parámetro* `str` *se pasa por dirección en la pila.*

***a)*** *(**20 %**) Programe en ensamblador del MC88110 la subrutina* `strncmp(str1,str2,n)` *que compara las cadenas de caracteres* `str1` *y* `str2` *y devuelve un 0 en r29 si los* `n` *primeros caracteres son iguales y 1 si no lo son. La comparación finaliza cuando se han comparado los* `n` *caracteres o se alcanza el final de una de las cadenas. Los parámetros se le pasan por dirección en la pila, y cada cadena de caracteres finaliza con el carácter NUL.*

***b)*** *(**80 %**) Programe en ensamblador del MC88110 la subrutina* `strtok(texto,str,v)` *que busca la cadena de caracteres* `str` *en* `texto` *y almacena la posición de cada ocurrencia en el vector* `v`*. Los parámetros* `texto` *y* `str` *corresponden a dos cadenas de caracteres terminadas con el carácter NUL y se pasan por dirección en la pila.* `v` *es un vector de enteros sin signo de 16 bits que se pasa por dirección en la pila. La subrutina devuelve en r29 el número de elementos del vector que se han rellenado.*

*Un ejemplo de una invocación a esta subrutina se muestra en la figura. A* `strtok` *se la llama con los parámetros* `texto` *(en la figura se muestran las posiciones que ocupa cada carácter de la cadena) y* `str`*. Se devuelve el valor 2 en el registro r29, puesto que se han rellenado las dos primeras posiciones del vector* `v` *que se muestra en la figura.*

*Para la realización de este ejercicio se llevará a cabo el tratamiento del Marco de Pila descrito en clase y se supondrá que están definidas todas las macros que se han explicado en la parte teórica de la asignatura; que las subrutinas llamantes dejan 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 el proyecto).*

`texto`

| e | s | t | o | | e | s | | u | n | a | | e | s | t | u | f | a | NUL |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 | 11 | 12 | 13 | 14 | 15 | 16 | 17 | |

`str`

| e | s | t | NUL |
|---|---|---|---|

`v`

| 0 |
|---|
| 12 |

# SOLUCIÓN

**a)** Se cargan las dos direcciones de comienzo de las cadenas en r20 y r21. Se comparan los caracteres apuntados por los dos punteros y si no son iguales se retorna de la función inmediatamente retornando el valor 1. Si son iguales y son el carácter NUL, las cadenas son iguales (se retorna de la función devolviendo el valor 0). Si son iguales y no son el carácter NUL, se sigue comparando hasta que se hayan comparado `n` caracteres. En este caso se retorna de la función devolviendo el valor 0, puesto que en la especificación se indica que la comparación está acotada a los `n` primeros caracteres. Por último, cabe destacar que en esta subrutina no se ha salvaguardado la dirección de retorno ni se ha creado el marco de pila porque esta subrutina no llama a ninguna otra y todas las variables locales de la subrutina se pueden albergar en los registros del procesador.

```
strncmp: ld r20,r30,r0          ; r20 y r21 contienen los dos punteros a
         ld r21,r30,4           ; str1 y str2
         ld r7,r30,8            ; r7 contiene el valor n
         or r29,r0,1            ; El valor de retorno se inicializa a 1
         or r4,r0,r0            ; r4 contiene el desplazamiento al comienzo de las cadenas
b_cmp:   ld.bu r2,r20,r4
         ld.bu r3,r21,r4
         cmp r5,r2,r3           ; Si los dos caracteres no son iguales se acaba
         bb1 ne, r5, no_iguales
         cmp r5,r2,r0           ; Si son iguales al carácter NUL, las cadenas son iguales
         bb1 eq, r5, iguales
```

```
        addu r4,r4,1
        subu r7,r7,1
        cmp r5,r7,r0          ; Si hemos comparado n caracteres, las cadenas son iguales
        bb1 eq, r5, iguales
        br b_cmp
iguales: subu r29,r29,1       ; Se devuelve 0
no_iguales: jmp(r1)
```

**b)** Para rellenar el vector `v` se recorre la cadena de caracteres texto caracter a carácter comparando con la cadena `str` acotando la comparación a la longitud de `str` (llamando a `strncmp`). Si el resultado de la llamada es 0 se rellena una entrada del vector. Esta operación se realiza hasta que se alcanza el carácter NUL de `texto`. Para realizar esta operación se crearán tres variables locales:

- Longitud de la cadena `str`. Se invocará a `strlen` en la fase de inicialización y se almacenará el resultado en esta variable para no tener que invocarla en cada iteración del bucle.
- Puntero a `texto`. Contiene la dirección del carácter de texto que se está evaluando.
- Desplazamiento al comienzo del vector `v`. Se utilizará para cargar la posición del carácter en el caso de que una llamada a `strncmp` devuelva un 0.

Cuando se finaliza la subrutina, el valor de retorno será el desplazamiento al vector `v` dividido entre dos bytes que ocupa cada entrada del vector.

```
strtok: PUSH(r1)
        PUSH(r31)
        or r31,r30,r30
        subu r30,r30,12
        ld r20,r31,12
        PUSH(r20)
        bsr strlen
        addu r30,r30,4
        st r29,r31,-4     ; Longitud del patrón que se pasa como parámetro
        ld r20,r31,8      ; Puntero al texto
        st r0,r31,-12     ; Se inicializa el desplazamiento a vector a 0
bucle_tok:
        ld.bu r3,r20,r0
        cmp r7,r3,r0      ; Si el carácter es NUL se acaba
        bb1 eq,r7,fin_tok
        ld r2,r31,-4      ; Longitud
        PUSH(r2)
        st r20,r31,-8     ; Puntero al texto
        PUSH(r20)
        ld r21,r31,12     ; Patrón
        PUSH(r21)
        bsr strncmp
        ld r20,r31,-8     ; Puntero
        addu r30,r30,12
        cmp r7,r29,r0
        bb1 ne, r7, cont
        ld r2,r31,-12     ; Desplazamiento al vector
        ld r21,r31,16     ; Vector
        ld r22,r31,8      ; Dirección del principio del texto
        subu r22,r20,r22  ; Posición que hay que almacenar en la tabla
        st.h r22,r21,r2
        addu r2,r2,2
        st r2,r31,-12
cont:   addu r20,r20,1    ; Se incrementa el puntero en una unidad
        br bucle_tok
```

```
fin_tok: ld r29,r31,-12   ; Se recupera el desplazamiento y se divide por 2
        ext r29,r29,0<1>
        or r30,r31,r31
        POP(r31)
        POP(r1)
        jmp(r1)
```



---

**NOTAS: 25 de noviembre de 2016** **DURACIÓN: 1 hora y 40 minutos**
**REVISIÓN: 29 de noviembre de 2016** **PUNTUACIÓN: 10 puntos**