6- Estudio de un S.E.T.I. básico I (Modelo de programación)

Ejercicios

Cuestiones/ejercicios

1. Las instrucciones se codifican con 6 bits, de modo que hay 64 códigos posibles. No obstante sólo disponemos de 31 instrucciones, por lo que hay 33 códigos que no tienen un significado para la máquina. ¿Que deberá hacer ésta en caso de encontrarse con uno de tales códigos?.

votos

votos		
(11)	1.	Activar el flag (+2+7)
(11)	2.	Pasar
(5)	3.	Parar
(3)	4.	Pedir introducir otra instrucción
(1)	5.	Definir la instrucción (pedirlo)
(2)	6.	No dejar huecos en la codificación
(6)	7.	Ejecutar otra instrucción según algún criterio

39 votos

✡ Esta opción invalida el enunciado

```
; ********************** Programa principal *********************

          lod-c 13        ; Preparación de la llamada a la subrutina con
          sto N1          ;   N1 = 13, N2 = 56, y dir_ret = volver.
          lod-c 56
          sto N2
          lod-c volver
          sto dir_ret
          jmp Multiplica  ; Llamada a la subrutina.

volver:   lod result      ; Cuando termina la subrutina, vuelve a este
                          ; punto. El resultado esta en “result”, y esta
                          ; instrucción lo pone en el acumulador.
          hlt             ; Terminación parando el computador.


dir_ret:  data            ; Dirección de retorno para la subrutina.
N1:       data            ; Un factor del producto.
N2:       data            ; El otro factor.
result:   data            ; El resultado de la subrutina ira aquí.
```

```
; ------------------- Subrutina de multiplicación -------------------
;
; Multiplica dos enteros mediante el algoritmo utilizado normalmente
; para multiplicar a mano.
; Parámetros de entrada: “N1” y “N2” los dos números a multiplicar
; Parámetro de salida: “result” el resultado.

                @100   ; Esta subrutina se carga en la posición 100.

Multiplica:     lod-c 0
                sto result

ciclo:          lod N1
                jmz fin
                shr
                sto N1
                jmf acumula
izquierda:      lod N2
                shl
                jmz fin
                sto N2
                jmp ciclo
acumula:        lod N2
                add result
                sto result
                jmp izquierda
fin:            jmp-i dir_ret
```

Macros Push y Pop
(sin control de límites)

```
Push:    macro
         sto      tmp
         lod      ap_pila
         dec
         sto      ap_pila
         lod      tmp
         sto-i    ap_pila
         mend

Pop:     macro
         lod-i    ap_pila
         sto      tmp
         lod      ap_pila
         inc
         sto      ap_pila
         lod      tmp
         mend

.....................

         @950
Pila_top:
         @1021
Pila_bot:
Ap_pila: data
Tmp:     data
```

Macro llamada a subrutina

```
Bsr:     macro
         lod-c    Ret\\
         push
         jmp      \1
Ret\\
         mend
```

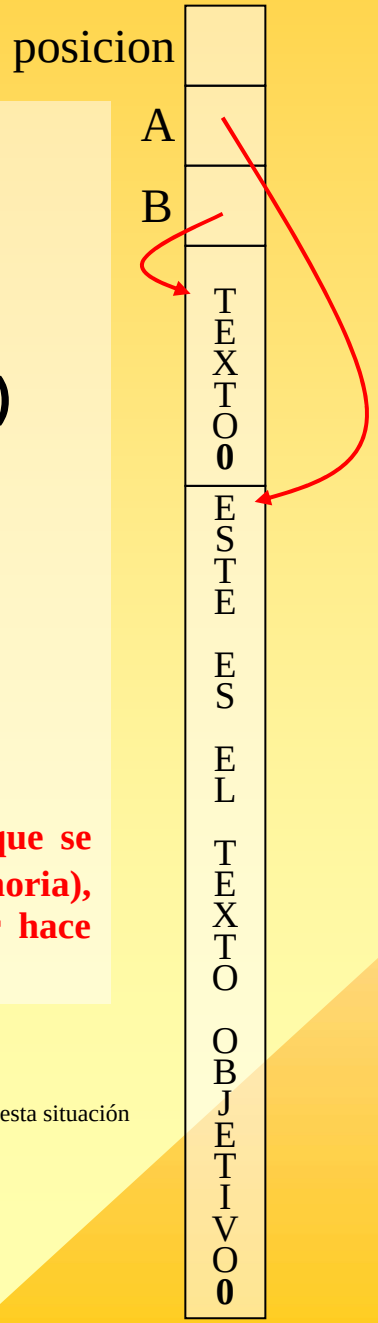
Macro retorno de subrutina

```
Rts:     macro
         pop
         jmp-i    tmp
         mend
```

Paso de parámetros y llamada rutina
(ejemplo)

```
         lod-c    13
         push
         lod-c    56
         push
         bsr      multiplica
         pop
         pop
         hlt
```

```
A=objetivo
hacer{
        B=patron
        posicion=A;
        mientras (*A!=0 y *B!=0 y (*A==*B) )
              {A++;B++;}
        Si (*B==0) ENCONTRADO;
        A=posicion+1;
} Mientras (*A!=0)
NO ENCONTRADO
```



!ojo!, "patrón" y "objetivo" son estructuras en memoria (strings), por lo que se entiende que su identificador hace mención a su referencia (dirección de memoria), mientras que "posicion", "A" y "B" son variables simples y su identificador hace mención a su valor (el contenido de la memoria).

Patologías de la solución

- Si cadena objetivo vacía, busca detrás de ella
  - Solución: inicializar posicion a cero y comenzar controlando esta situación
- Si patrón vacío sale directo
  - Solución: es necesario definir qué se pretende en tal caso