

# Ejemplos de programación en ensamblador

Estructura de Computadores

Octubre 2011

## Ejemplo 1

```
;Suma de un número variable de datos almacenados en posiciones consecutivas
;de memoria a partir de la variable SUMDOS.
;Cada dato ocupa una palabra y está representado en binario puro.
;El número de datos queda definido por la variable N y el resultado se almacena
;en la variable RESULT.
;Si existe desbordamiento se devuelve -1 en el registro r30, caso contrario se
;devuelve 0 en dicho registro.
;Se definen las siguientes macros:
LEA:    MACRO(ra,eti)
        or    ra,r0,low(eti)
        or.u  ra,ra,high(eti)
        ENDMACRO
;Carga el registro ra con la dirección efectiva definida por la etiqueta eti.
DBNZ:   MACRO(ra,eti)
        sub   ra,ra,1
        cmp   r4,ra,0
        bb0   2,r4,eti
        ENDMACRO
;Decrementa el registro ra y salta a la etiqueta eti si no es cero el resultado
;utilizando r4 como registro auxiliar.
;
;Definición de datos utilizados por el programa
        org   1000
N:       data   5
SUMDOS:  data   0x80000000,24,0x2fffffff,0,3
RESULT:  data   0
;
;Tras la ejecución del programa a partir de la dirección de memoria 1000
;deben aparecer los valores:
; 992                                05000000    00000080
;1008    18000000    FFFFFFF2F    00000000    03000000
;1024    1A0000B0    00000000    00000000    00000000
```

## Ejemplo 2

;Una matriz de M filas y N columnas está almacenada por filas a partir de la  
;variable MATRIZ. Cada elemento ocupa 1 palabra y está representado en binario  
;puro. Sumar los elementos de cada columna dejando el resultado como un vector  
;de doble precisión a partir de la variable VSUMA.

;Este vector está almacenado según el ordenamiento little-endian.

```
LEA:    MACRO (ra, eti)
        or      ra, r0, low(eti)
        or.u    ra, ra, high(eti)
```

ENDMACRO

;Carga el registro ra con la dirección efectiva definida por la etiqueta eti.

```
LOAD:   MACRO (ra, eti)
        LEA     (ra, eti)
        ld      ra, ra, r0
```

ENDMACRO

;Carga ra con el contenido de memoria que indica la etiqueta eti.

```
DBNZ:   MACRO (ra,eti)
        sub     ra,ra,1
        cmp     r5,ra,0
        bb0     2,r5,eti
```

ENDMACRO

;Decrementa el registro ra y salta a la etiqueta eti si no es cero el resultado  
;utilizando r5 como registro auxiliar.

;Datos del problema.

```
        org     1008
FILAS:   data    5, 0
COLUMNAS: data    4, 0
MATRIZ:  data    0xc000000b, 0xc000000a, 0xc0000009, 0xc0000008
        data    0xc0000007, 0xc0000006, 0xc0000005, 0xc0000004
        data    0xc0000003, 0xc0000002, 0xc0000001, 0xc0000000
        data    0xe000000b, 0xe000000a, 0xe0000009, 0xe0000008
        data    0xe0000007, 0xe0000006, 0xe0000005, 0xe0000004
VSUMA:   res     32
```

;Resultado del ejemplo:

; 1008	05000000	00000000	04000000	00000000
; 1024	0B0000C0	0A0000C0	090000C0	080000C0
; 1040	070000C0	060000C0	050000C0	040000C0
; 1056	030000C0	020000C0	010000C0	000000C0
; 1072	0B0000E0	0A0000E0	090000E0	080000E0
; 1088	070000E0	060000E0	050000E0	040000E0
; 1104	27000000	04000000	22000000	04000000
; 1120	1D000000	04000000	18000000	04000000

### Ejemplo 3

;Inserción de un elemento en una lista no ordenada.  
;La lista comienza a partir de la dirección definida por la etiqueta NOMBRE y  
;cada elemento ocupa 1 palabra, estando almacenados en posiciones consecutivas  
;de memoria. La longitud de la lista y el elemento a insertar vienen definidos  
;por las variables LONG y NUEVO.  
;Si el nuevo no se encuentra en la lista, se inserta al final de la misma y se  
;incrementa la longitud. Si el nuevo se encuentra en la lista, no se hace nada.

```
org 1000
NOMBRE: data 6,5,7,4,80,-1,0,-3,101,10
org 2000
LONG: data 10
NUEVO: data 33
```

;Tras la ejecución del programa quedarán en memoria los siguientes valores:

```
; 2000 0B000000 21000000 00000000 00000000

; 992
; 1008 07000000 04000000 50000000 FFFFFFFF
; 1024 00000000 FFFFFFFF 65000000 0A000000
; 1040 21000000
```

### Ejemplo 4

;Inserción de un elemento en una lista ordenada.  
;La lista está almacenada a partir de la dirección definida por LISTOR y está  
;constituida por elementos de 1 palabra de longitud expresados en binario puro.  
;Están ordenados de menor a mayor y están almacenados en posiciones de memoria  
;consecutivas. El tamaño de la lista está almacenado en la dirección  
;LONLIS y el elemento a insertar está almacenado en la variable NUEVO.  
;La lista no almacena elementos repetidos.

```
org 1000
LISTOR: data 1,3,7,45,56,67,101,200
org 2000
LONLIS: data 8
NUEVO: data 90
```

;Tras la ejecución del programa los contenidos de memoria son:

```
; 992 01000000 03000000
; 1008 07000000 2D000000 38000000 43000000
; 1024 5A000000 65000000 C8000000 00000000
;
; 2000 09000000 5A000000
```

## Ejemplo 5

```
;Inserción de un elemento en una lista con punteros.
;Cada elemento de la lista consta de dos palabras consecutivas en memoria, la
;primera contiene la dirección del siguiente elemento de la lista y la
;segunda el valor numérico del elemento, que está representado en binario puro.
;La lista está ordenada de menor a mayor.
;La variable CABECERA contiene la dirección del primer elemento de la lista y
;la lista termina cuando el campo de dirección de un elemento contiene el valor
;0x00000000. Por lo tanto, si la variable CABECERA contiene el valor cero es
;que la lista está vacía.
;La variable NUEVO contiene la dirección del elemento nuevo a insertar en la
;lista.
```

```
    ;DATOS
        org      1000
CABECERA: data    2000
NUEVO:   data     2200
    ;EJEMPLO PARA COMPROBACIÓN
        org      2000
        data     2024,2
        org      2024
        data     2048,5
        org      2048
        data     2072,10
        org      2072
        data     2096,13
        org      2096
        data     0,15
        org      2200
        data     2112,7
;Tras la ejecución del programa el contenido de memoria queda:
;      2000      E8070000      02000000      00000000      00000000
;              (2024)      (a1)
;      2016      00000000      00000000      98080000      05000000
;              (2200)      (a2)
;      2032      00000000      00000000      00000000      00000000
;      2048      18080000      0A000000      00000000      00000000
;              (2072)      (a4)
;      2064      00000000      00000000      30080000      0D000000
;              (2096)      (a5)
;      2080      00000000      00000000      00000000      00000000
;      2096      00000000      0F000000      00000000      00000000
;              (fin)      (a6)
;      2112      00000000      00000000      00000000      00000000
;      2128      00000000      00000000      00000000      00000000
;      2144      00000000      00000000      00000000      00000000
;      2160      00000000      00000000      00000000      00000000
;      2176      00000000      00000000      00000000      00000000
;      2192      00000000      00000000      00080000      07000000
;              (2048)      (a3)
```

## Ejemplo 6

```
;Recorrido de una lista encadenada de cadenas de caracteres (strings)
;con objeto de determinar su longitud y el número total de caracteres
;de la lista.
;
;La lista está almacenada a partir de la dirección almacenada en la
;posición de memoria definida por LISTAENC. Cada elemento de la lista
;consta de una cadena de caracteres seguida de una palabra que contiene
;la dirección de comienzo del siguiente elemento de la lista. La cadena
;contiene un número indefinido de caracteres y se termina con el carácter
;null (0x00, '\0').
;La dirección del siguiente elemento está en la siguiente palabra de
;memoria, es decir, en la siguiente dirección múltiplo de cuatro. Un
;cero almacenado en esta dirección indica el final de la lista encadenada.
;Un cero almacenado en la dirección LISTAENC indica que la lista está vacía.
;
;Determinar la longitud de cada cadena de la lista, incluyendo el carácter
;de fin de cadena (null), y almacenarla en los correspondientes elementos
;de una lista encadenada que comienza a partir de la dirección de memoria
;almacenada en la posición definida por LISTALONG. Cada elemento de esta
;lista se compone de dos palabras; en la primera debe almacenarse la
;longitud de la cadena correspondiente en la lista LISTAENC. La segunda
;palabra contiene la dirección del siguiente elemento de la lista.
;Un cero en esta palabra indica el final de la lista.
;
;El programa finalizará prematuramente si se detecta alguna inconsistencia
;en los datos. En todo caso, en la dirección NELEM se almacenará el número
;de elementos de la lista procesados correctamente y en la dirección NCAR
;se almacenará el número total de caracteres de las cadenas procesadas
;correctamente.
```

```
        ; DATOS
        org 1000
LISTAENC: data 2000
LISTALONG: data 0x4000
NELEM:   res 4
NCAR:    res 4

        ; EJEMPLO PARA COMPROBACIÓN
        org 1200
        data "Hola, amigo\0"
        data 1500
        org 1500
        data "Qué tal estás?\0"
        data 0
        org 2000
        data "Soy 88110\0"
        data 1200
```

```
org 0x1400
data 0, 0x4008
org 0x4000
data 0, 0x1400, 0, 0
```

```
;Tras la ejecución del programa los contenidos de memoria son:
;      1008      03000000      25000000      00000000      00000000
;
;      0x1400     0C000000     08400000     00000000     00000000
;
;      0x4000     0A000000     00140000     0F000000     00000000
```