

Subprogramas

Tutoría académica nº6
Principio de Computadores

- Los subprogramas (procedimientos y funciones) son la herramienta fundamental para la abstracción y la reutilización de código. En esta presentación, de forma general, haremos referencia a funciones.
- La pila (**stack**) es una zona de memoria dinámica que cobra especial relevancia en el uso de las funciones.
- Cosas que intervienen en la llamada a una función:
 - Los argumentos que necesita la función para ejecutarse.
 - Los valores que devuelve la función.
 - El lugar donde tiene que regresar la ejecución al finalizar la función.
 - Los posibles datos locales que requiera la función.

De forma general, se usarán siempre los registros específicos, limitando el uso de la pila para cuando sea necesario.

REGISTROS INVOLUCRADOS EN EL USO DE FUNCIONES

Paso de argumentos:

- Enteros: \$a0,\$a1,\$a2,\$a3
- Simple precisión: \$f12,\$f13,\$f14,\$f15 (cuidado con el solapamiento con doble precisión)
- Doble precisión: \$f12,\$f14 (cuidado con el solapamiento con doble precisión)

Valores de retorno:

- Enteros: \$v0 y \$v1
- Simple precisión: \$f0,\$f1,\$f2, \$f3 (cuidado con el solapamiento con doble precisión)
- Doble precisión: \$f0, \$f2 (cuidado con el solapamiento con simple precisión)

Dirección de retorno: \$ra

Puntero de pila: \$sp

Marco de pila: \$fp

LLAMADA A UNA FUNCIÓN.

Las funciones se llaman con la instrucción:


`jal etiqueta_funcion`

Esto hace dos cosas:

1. $\$ra \leftarrow PC+4$
2. $PC \leftarrow \text{etiqueta_funcion}$

Antes de ejecutar `jal minvect` los valores de PC y $\$ra$ eran:

PC	=	400064
R31 [ra]	=	400018



esta dirección corresponde con la de la instrucción posterior a la llamada `jal main`.

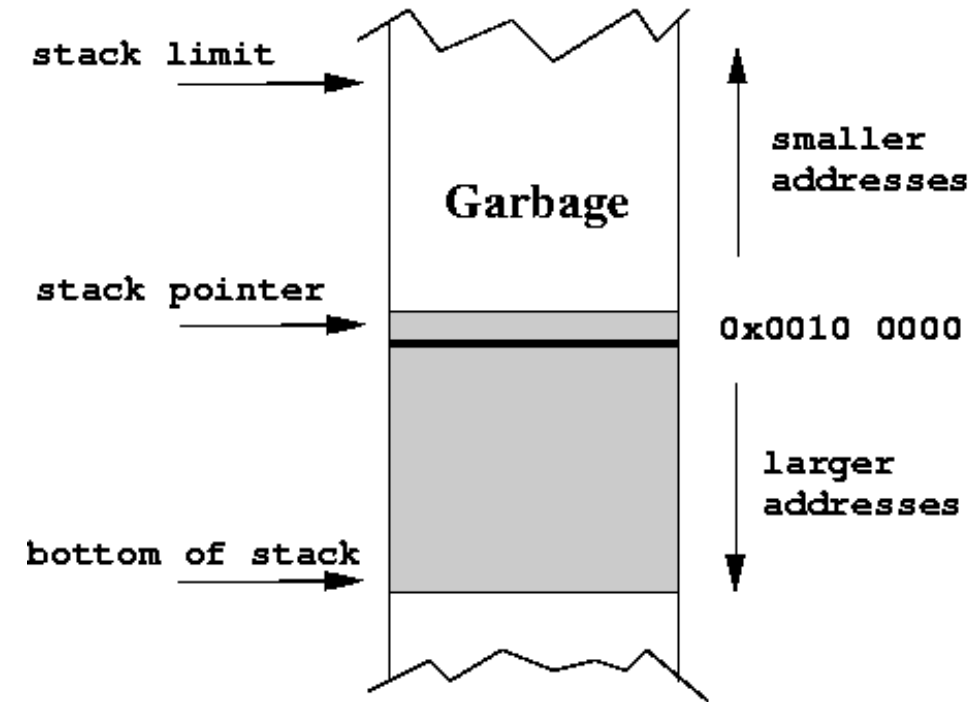
Después de ejecutar `jal minvect`:

PC	=	400024
R31 [ra]	=	400068

```
1      .data
2      vector: .word    -1, 1, 3, 4, -2, 8, 10, 12, 6, -6
3      dim:    .word    10
4
5
6      minvect: # funcion que calcula el minimo de un vector
7                # de enteros, y lo devuelve en $v0
8                # Argumentos: $a0 direccion base del vector.
9                #                $a1 numero de elementos del vector
10     [00400024] move $t0,$a0
11                addi $t1,$a1,-1
12                lw $v0,0($t0) # tomo minimo el primero
13                loop: ble $t1,$zero,finloop
14                    addi $t0,4 # desplazo al siguiente
15                    lw $t2,0($t0)
16                    bge $t2,$v0,next
17                    move $v0,$t2
18                next: addi $t1,-1
19                    j loop
20                finloop: jr $ra
21
22     main:
23         la $a0,vector
24         lw $a1,dim
25     [00400064] jal minvect
26                # el minimo queda en $v0. Se imprime por pantalla
27
28     [00400068] move $a0,$v0
29                li $v0,1
30                syscall
31                li $v0,10
32                syscall
```

Stack

- La pila crece de direcciones mayores hacia direcciones más pequeñas.
- El puntero de pila \$sp (stack pointer) contiene la dirección válida más pequeña de la pila. Cualquier dirección de la pila más pequeña contendrá basura (garbage).
- La base de la pila (bottom of stack) contiene la dirección más alta que puede tener la pila. Cuando la pila se inicializa el \$sp apunta a esta dirección.
- El límite de la pila (stack limit) contiene la dirección mínima válida que puede tener la pila. Si \$sp va por debajo de esta dirección se producirá el error stack overflow.



Operaciones del stack.

- **push**: coloca algo en la base de la pila y actualiza el stack pointer.

Ej: salvar \$s0 en la pila

```
addi    $sp,$sp,-4  
sw      $s0,0($sp)
```

- **pop**: extrae el elemento colocado en la cima (la cima es la dirección apuntada por el stack pointer) de la pila y actualiza el stack pointer.

Ej: recuperar \$s0 de la pila.

```
lw      $s0,0($sp)  
addi    $sp,$sp,4
```

- Como es lógico puedes introducir varios registros y actualizar el \$sp una sola vez.

Ejemplo de un push de tres registros:

```
addi    $sp,$sp,-12  
sw      $t0,0($sp)  
sw      $t1,4($sp)  
sw      $t2,8($sp)
```

¿Sabrías decir qué registro está en la cima del stack?

CASOS EN LOS QUE SERÁ NECESARIO UTILIZAR EL STACK.

- Cuando para pasar argumentos no son suficientes los registros estándar.
- Cuando para devolver valores no son suficientes los registros estándar.
- Cuando una función, llama a su vez a otra función. En este caso concreto siempre deberá guardarse en la pila la dirección de retorno y aquellos argumentos que tenga que ser sobrescritos.
- Cuando haya que utilizar registros salvados dentro de la función, ya que según convenio deben dejarse restaurados antes de finalizar.

En cualquier otro supuesto recuerda que usar registros es siempre más eficiente que usar la memoria y por lo tanto se debe minimizar el uso del stack.