



Subrutinas

Ing. Alejandro C. Rodríguez Costello
Ing. Walter Lozano

*“Leer manuales de computación sin el hardware
es tan frustrante como leer manuales de sexo sin el software.”
Arthur C. Clarke*

Objetivos

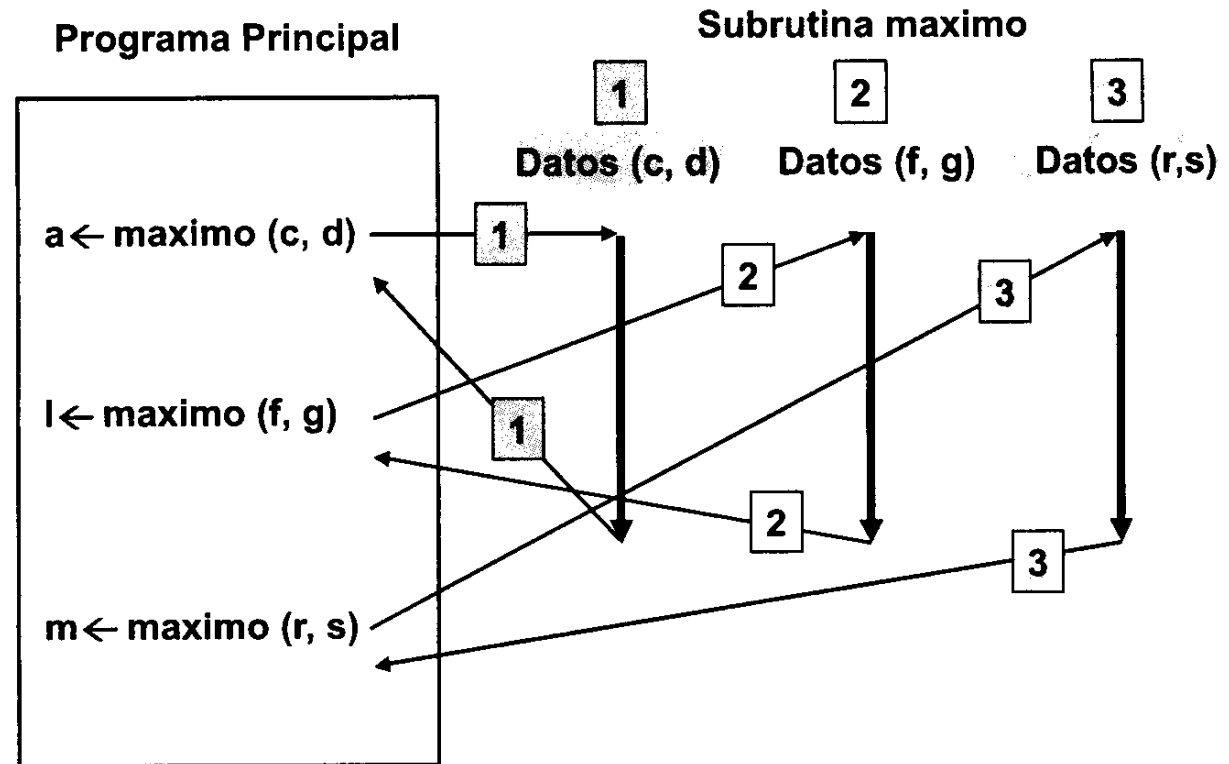
- ✍ Estudiar los tipos de subrutinas.
- ✍ Entender la relación entre parámetros y pila.
- ✍ Introducir al alumno en la programación recursiva.
- ✍ Entender las convenciones para su uso.
- ✍ Soportar la programación en assembly con HLLs.

Gestión de subrutinas

✍ Una subrutina implementa un código utilizado con mucha frecuencia


✍ **Ventajas:**

- ✍ Modularidad
- ✍ Reutilización de código
- ✍ Librerías
- ✍ Facilidad de depuración
- ✍ Flexibilidad
- ✍ Claridad en el diseño

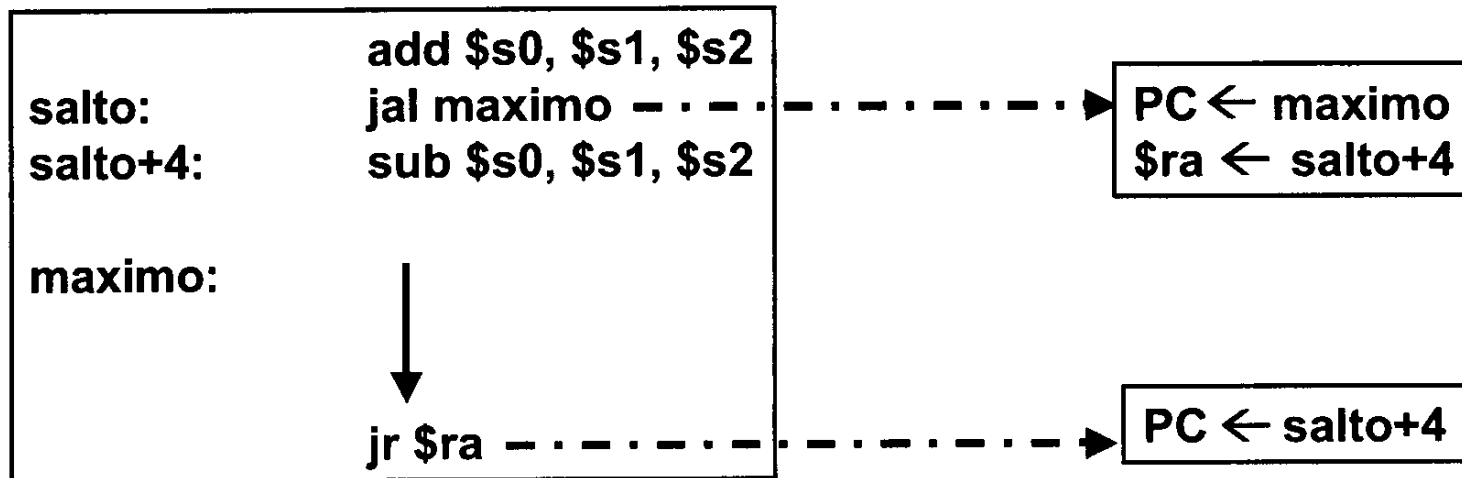


Gestión de subrutinas

Llamadas a subrutinas (programa invocado)

 Transferir el control a la subrutina
 Almacenar la dirección de retorno }  jal etiqueta

 Devolver el control al programa que hace la llamada (programa invocador)  jr \$ra



Gestión de subrutinas

Parámetros

- ↳ Entrada a la subrutina
- ↳ Salida a la subrutina

Paso de parámetros


- ↳ Mediante registros. El ensamblador establece por convenio
 - ↳ \$a0-\$a3: parámetros de entrada
 - ↳ \$v0-\$v1: parámetros de salida
- ↳ Mediante la pila

Tipos de parámetros

- ↳ valor: el parámetro es el dato
- ↳ referencia: el parámetro es la dirección de memoria donde está almacenado el dato.

Gestión de subrutinas

Llamada a una subrutina. Parámetros por valor

 Programa invocador:

1. Cargar el valor de los parámetros en los registros \$a0-\$a3
2. Llamada a la subrutina
3. Leer los parámetros de salida de los registros \$v0-\$v1

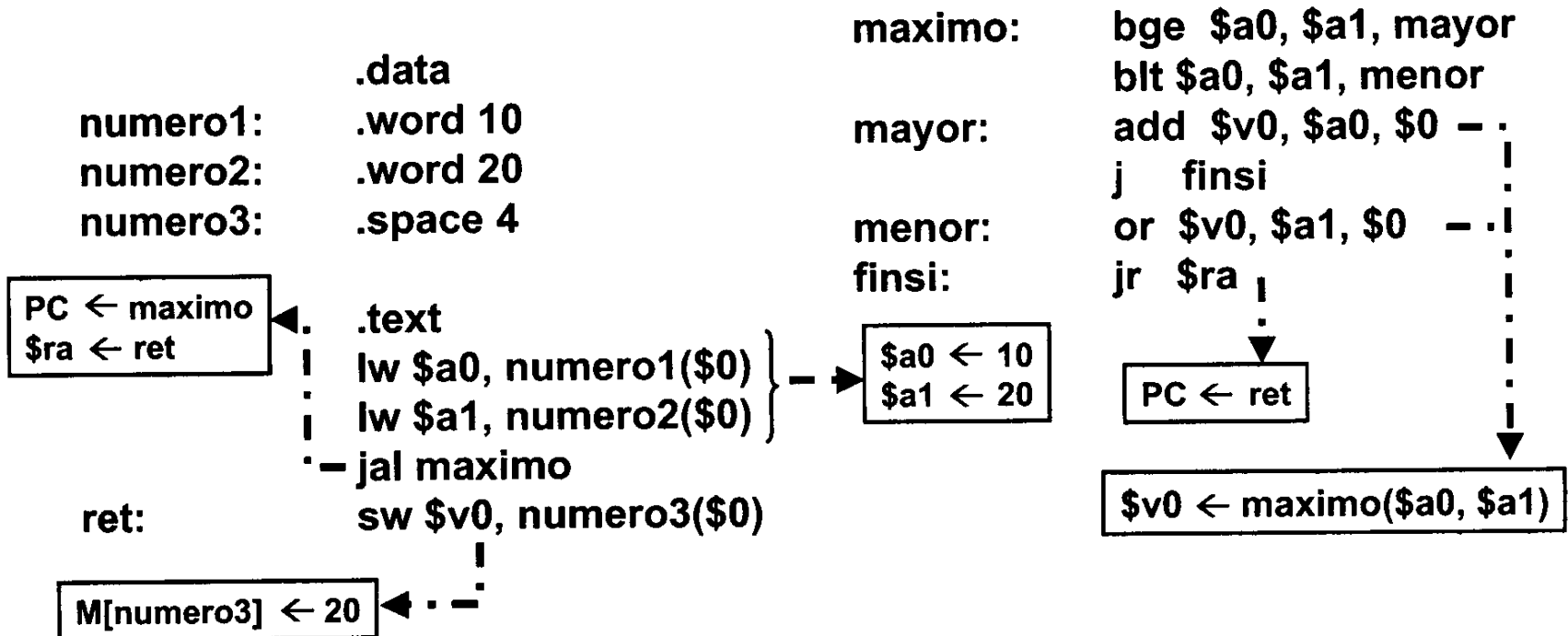
 Programa invocado:

- a. Leer los parámetros de entrada desde los registros \$a0-\$a3
- b. Realizar tarea
- c. Devolver el valor de los parámetros de salida sobre registros \$v0-\$v1
- d. Devolver el control al programa invocador

Gestión de subrutinas


✍ Ejemplo de llamada a una subrutina pasando los parámetros por valor

↳ Subrutina que calcula el máximo de dos números



Gestión de subrutinas

Llamada a una subrutina. Parámetros por referencia

 Programa invocador:

1. Carga la dirección de los parámetros de entrada y salida en los registros \$a0-\$a3.
2. Llamada a la subrutina

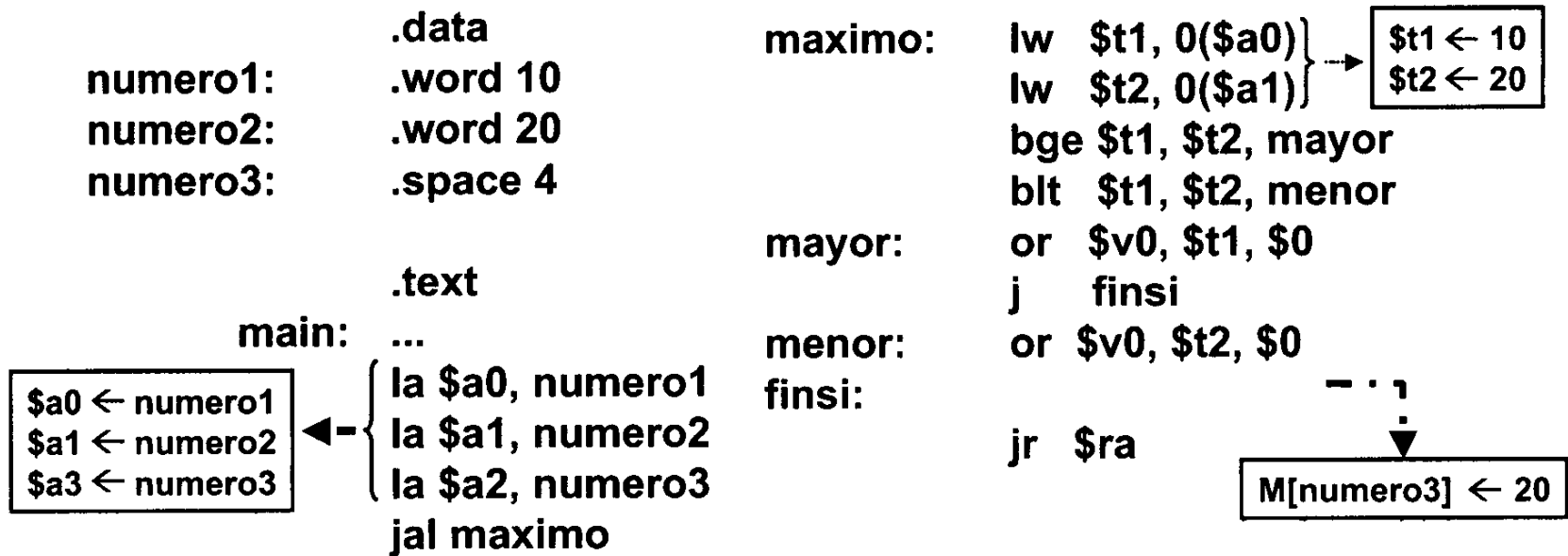
 Programa invocado:

- a. Leer la dirección de los parámetros de los registros \$a0-\$a3
- b. Leer los parámetros de las posiciones de memoria referenciadas
- c. Realizar tarea
- d. Almacenar los parámetros de salida en las direcciones de memoria correspondientes.
- e. Devolver el control al programa invocador

Gestión de subrutinas

✍ Ejemplo de llamada a una subrutina pasando los parámetros por referencia

↳ Subrutina que calcula el máximo de dos números



↳ **Problema:** ¿Qué ocurre con el contenido de los registros $\$t1$ y $\$t2$ cuando el control es devuelto al invocador?

Gestión de subrutinas

- ✍ **Solución: aquellos registros que van a ser modificados en la subrutina y que el invocador desea mantener después de la llamada deben salvarse en la pila (\$s0-\$s7, \$t0-\$t9)**
- ✍ **Convenio que establece el ensamblador del MIPS para salvar los registros en la pila**
 - ↳ El invocador:
 - ↳ Salva los registros **\$t0-\$t9** cuyo contenido le interesa mantener después de la llamada.
 - ↳ El invocado:
 - ↳ Salva los registros **\$s0-\$s7** que utiliza

Gestión de subrutinas

✍ Ejemplo de llamada a una subrutina pasando los parámetros por referencia y salvando el contenido de los registros **\$t1**, **\$t2** en la pila.

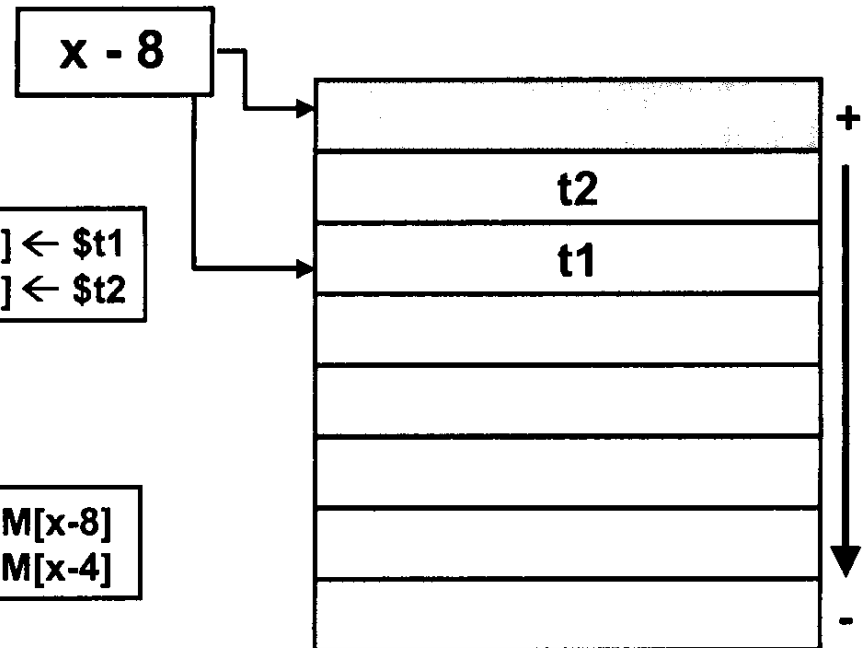
↳ Subrutina que calcula el máximo de dos números

main: ...

```
addi $sp, $sp, -8  
sw $t2, 4($sp)  
sw $t1, 0($sp)  
la $a0, numero1  
la $a1, numero2  
la $a2, numero3  
jal maximo  
lw $t1, 0($sp)  
lw $t2, 4($sp)  
addi $sp, $sp, 8
```

$M[x-8] \leftarrow \$t1$
 $M[x-4] \leftarrow \$t2$

$\$t1 \leftarrow M[x-8]$
 $\$t2 \leftarrow M[x-4]$

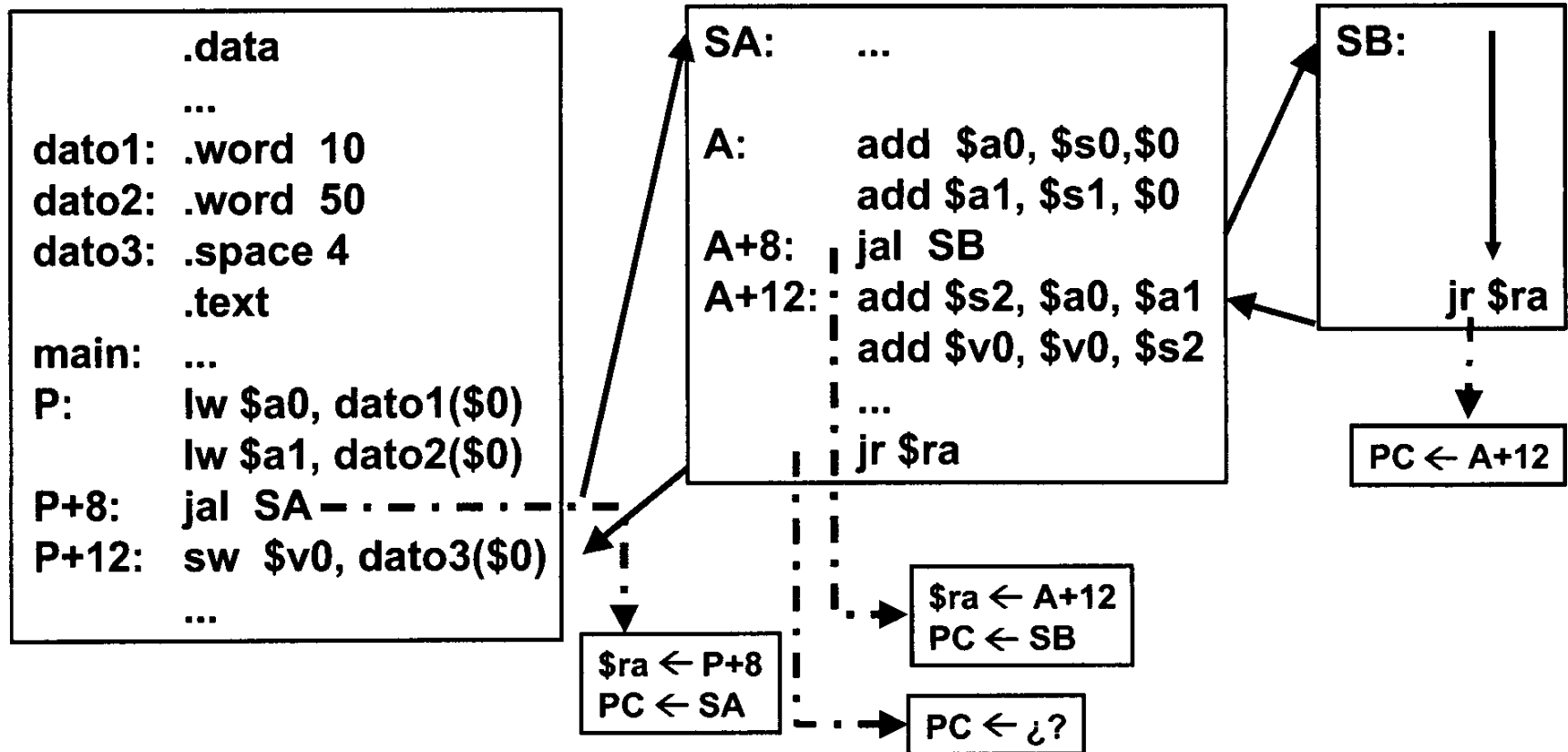


Gestión de subrutinas

Anidamiento de subrutinas.

↳ Una subrutina llama a otra

↳ Una subrutina se llama a si misma (recursividad)



Gestión de subrutinas

Problemas:

- ↳ Cuando se retorna de la subrutina SB a la rutina SA ¿Qué valores continen los registros \$a0-\$a1?
- ↳ ¿Cómo se retorna desde la subrutina SA al programa main?

Solución:

- ↳ La subrutina SA apila los registros **\$a0-\$a1** y **\$ra** antes de modificarlos y hacer la llamada a la subrutina SB
- ↳ Además apilará los registros **\$s0-\$s2** antes de modificar su contenido.

Problema:

- ↳ ¿Donde se almacenan las variables locales de una subrutina, que no se pueden almacenar en registros?

Solución:

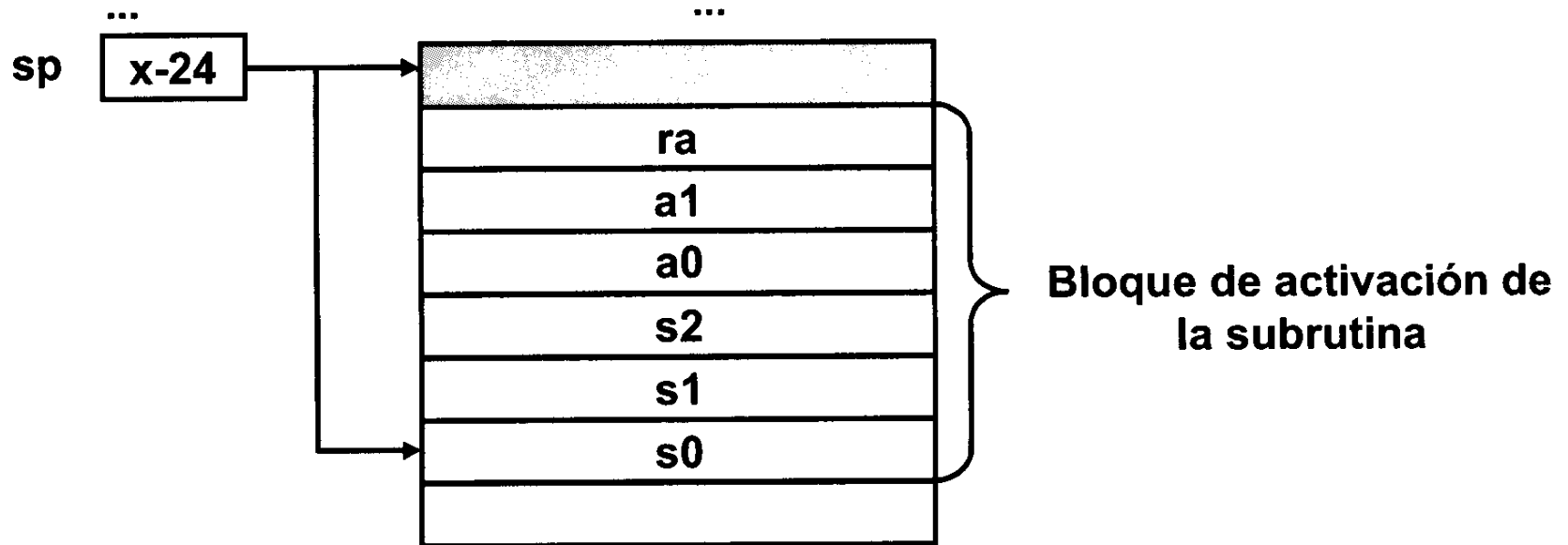
- ↳ En la pila después de los registros almacenados en ella

Gestión de subrutinas

SA: **addi \$sp, \$sp, -24**
 sw \$s0, 0(\$sp)
 sw \$s1, 4(\$sp)
 sw \$s2, 8(\$sp)
 sw \$a0, 12(\$sp)
 sw \$a1, 16(\$sp)
 sw \$ra, 20(\$sp)

A: **add \$a0, \$s0, \$0**
 add \$a1, \$s1, \$0
A+8: **jal SB**
A+12: **lw \$a0, 12(\$sp)**
 lw \$a1, 16(\$sp)
 add \$s2, \$a0, \$a1
 add \$v0, \$v0, \$s2

lw \$ra, 20(\$sp)
lw \$s0, 0(\$sp)
lw \$s1, 4(\$sp)
lw \$s2, 8(\$sp)
addi \$sp, \$sp, 24
jr \$ra

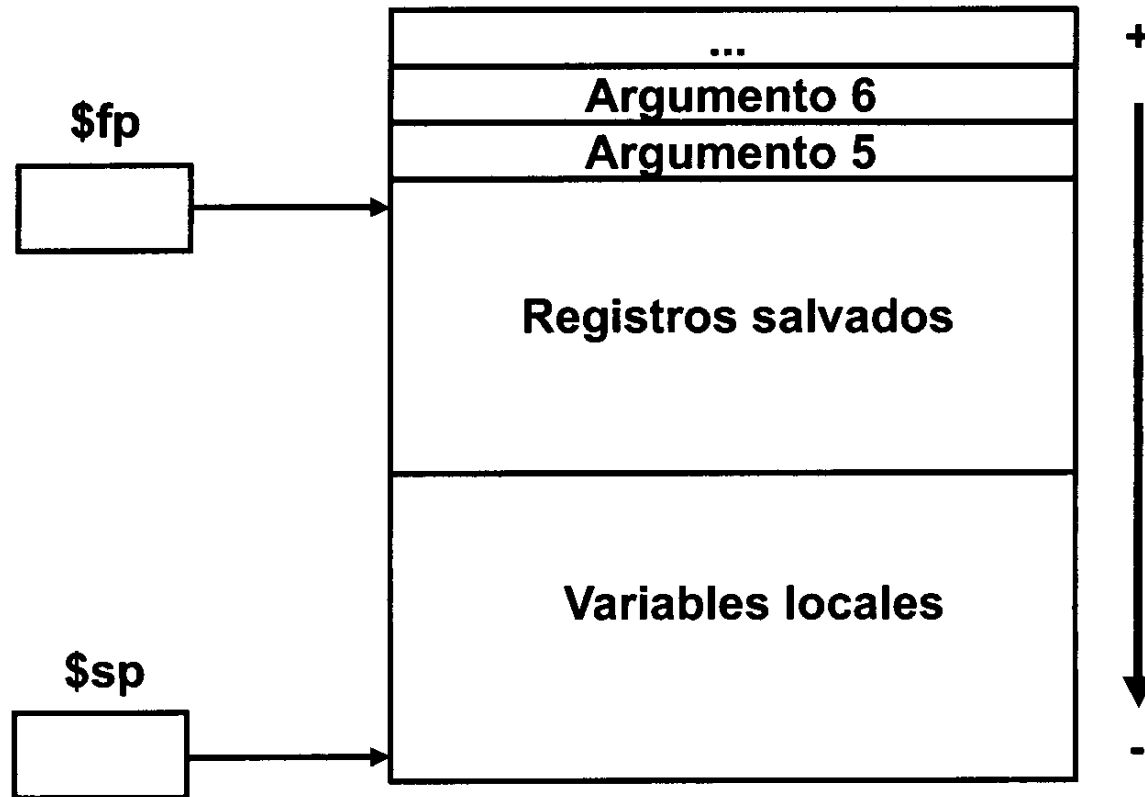


Gestión de subrutinas

- ✍ El segmento de pila que contine toda la información referente a una subrutina (registros salvados y variables locales) se llama bloque de activación de la subrutina
- ✍ Los programas basados en el ensamblador del MIPS usan un puntero, registro \$fp, para señalar la primera palabra del bloque de activación del procedimiento
 - ↳ Este registro proporciona un registro base estable para referenciar a todos los elementos localizados en la pila durante la llamada a un procedimiento
 - ↳ El puntero de pila apunta a la última palabra del bloque de activación

Gestión de subrutinas

- ✍ **Segmento de pila correspondiente al bloque de activación en una llamada a una subrutina**

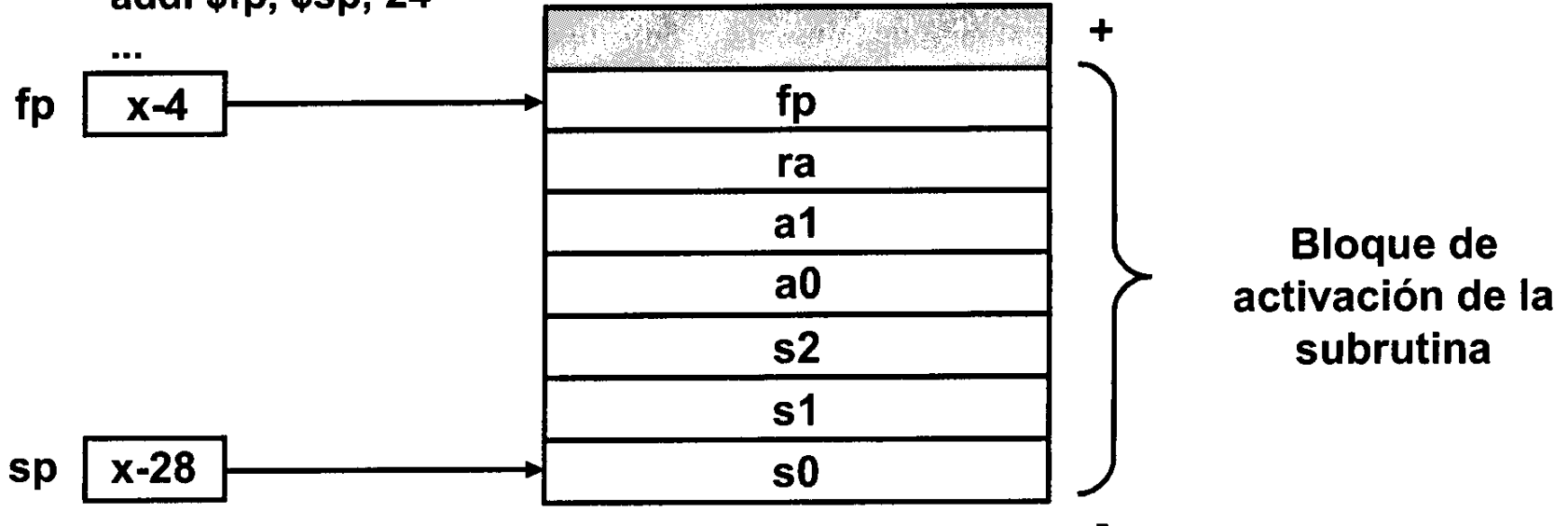


Gestión de subrutinas

SA: addi \$sp, \$sp, -28
 sw \$s0, 0(\$sp)
 sw \$s1, 4(\$sp)
 sw \$s2, 8(\$sp)
 sw \$a0, 12(\$sp)
 sw \$a1, 16(\$sp)
 sw \$ra, 20(\$sp)
 sw \$fp, 24(\$sp)
 addi \$fp, \$sp, 24

A: add \$a0, \$s0, \$0
 add \$a1, \$s1, \$0
A+8: jal SB
A+12: lw \$a0, -12(\$fp)
 lw \$a1, -8(\$fp)
 add \$s2, \$a0, \$a1
 add \$v0, \$v0, \$s2
 ...




lw \$ra, -4(\$fp)
lw \$s0, -24(\$fp)
lw \$s1, -20(\$fp)
lw \$s2, -16(\$fp)
addi \$sp, \$fp, 4
sw \$fp, -4(\$sp)
jr \$ra



Gestión de subrutinas

Pasos que resumen la llamada a una subrutina según el convenio establecido por el ensamblador del Mips (I)

El invocador

-  Paso de parámetros. Por convención los cuatro primeros parámetros se pasan a través de los registros \$a0-\$a3. Cualquier parámetro adicional se pone en la pila y está al principio del bloque de activación de la subrutina.
-  Cualquiera de los registros \$a0-\$a3 y \$t0-\$t9 que espera utilizar después de la llamada lo salva en la pila
-  Ejecuta la llamada

Gestión de subrutinas

Pasos que resumen la llamada a una subrutina según el convenio establecido por el ensamblador del Mips (II)

El invocado

- ✧ Reserva memoria para el bloque de activación, actualizando el registro `$sp` restándole el tamaño del mismo (bytes que ocupan los registros salvados por la subrutina y las variables globales que se van a almacenar en la pila)
- ✧ Salvar los registros `$s0-$s7`, `$fp` y `$ra` que va a modificar la subrutina
- ✧ Establece el puntero de bloque de activación, añadiendo el tamaño de bloque de activación menos cuatro al `sp` y guardando el resultado en el registro `$fp`

Antes de retornar el invocado

- ✧ El resultado de la subrutina se devuelve sobre los registros `$v0-$v1`
- ✧ Restaurar los registros apilados por el invocado
- ✧ Desapila el bloque restando el tamaño de bloque al `$sp`
- ✧ Retorna saltando a la dirección del registro `$ra`