Subrutinas - MIPS

Arquitectura de Computadoras

Curso 2020

Prof. Jorge M. Runco

Llamados a subrutinas en MIPS

- jal = jump and link
  - Ej. jal fun
  - Hace 2 cosas:
  - Salta a fun, que es una etiqueta (dirección) de la primera instrucción de la subrutina.
  - Salva en $ra (R31), la dirección de retorno (dir de jal + 4)
- jr $ra
  - Carga en el PC la dirección salvada en $ra (como el ret)
- Con estas dos instrucciones podemos llamar y retornar de subrutinas.

Llamado y retorno

- Ej.

```
main:  ……
       ……
        jal fun
       …….

fun:  daddi  $v0, $a0, $a1
      jr  $ra
```

Uso de registros en llamadas a subrutinas

- Assembly en MIPS utiliza la siguiente convención para el uso de los registros:
  - $a0 - $a3 : cuatro registros para pasar parámetros (entrada) al procedimiento.
  - $v0 – Sv1 : dos registros para devolver parámetros (salida).
  - $ra : registro donde se salva la dirección de retorno.

Uso de registros en llamadas a subrutinas

- En general una función necesita usar varios registros para realizar cálculos. Se van a modificar registros dentro de la función.
- Se pueden usar $t0-$t9 libremente dentro de la función. El programa que llama a la función “no espera” que esos registros conserven sus valores.
- Pero el programa que llama a la función “espera” que los registros $s0-$s7 conserven sus valores antes y después de la llamada.
- Hay que evitar (si es posible) el uso de los registros $s0-$s7. Si no es posible hay que salvarlos

PUSH en MIPS

- Nosotros debemos ajustar al puntero de pila
- $sp := $sp – 8
- daddi $sp, $sp, -8
- Ahora enviamos el dato a la pila
- sd $t1, 0($sp)

POP en MIPS

- Sacamos el dato de la pila
- ld $t1, 0($sp)
- Ahora ajuatamos el puntero de pila
- $sp := $sp + 8
- daddi $sp, $sp, 8

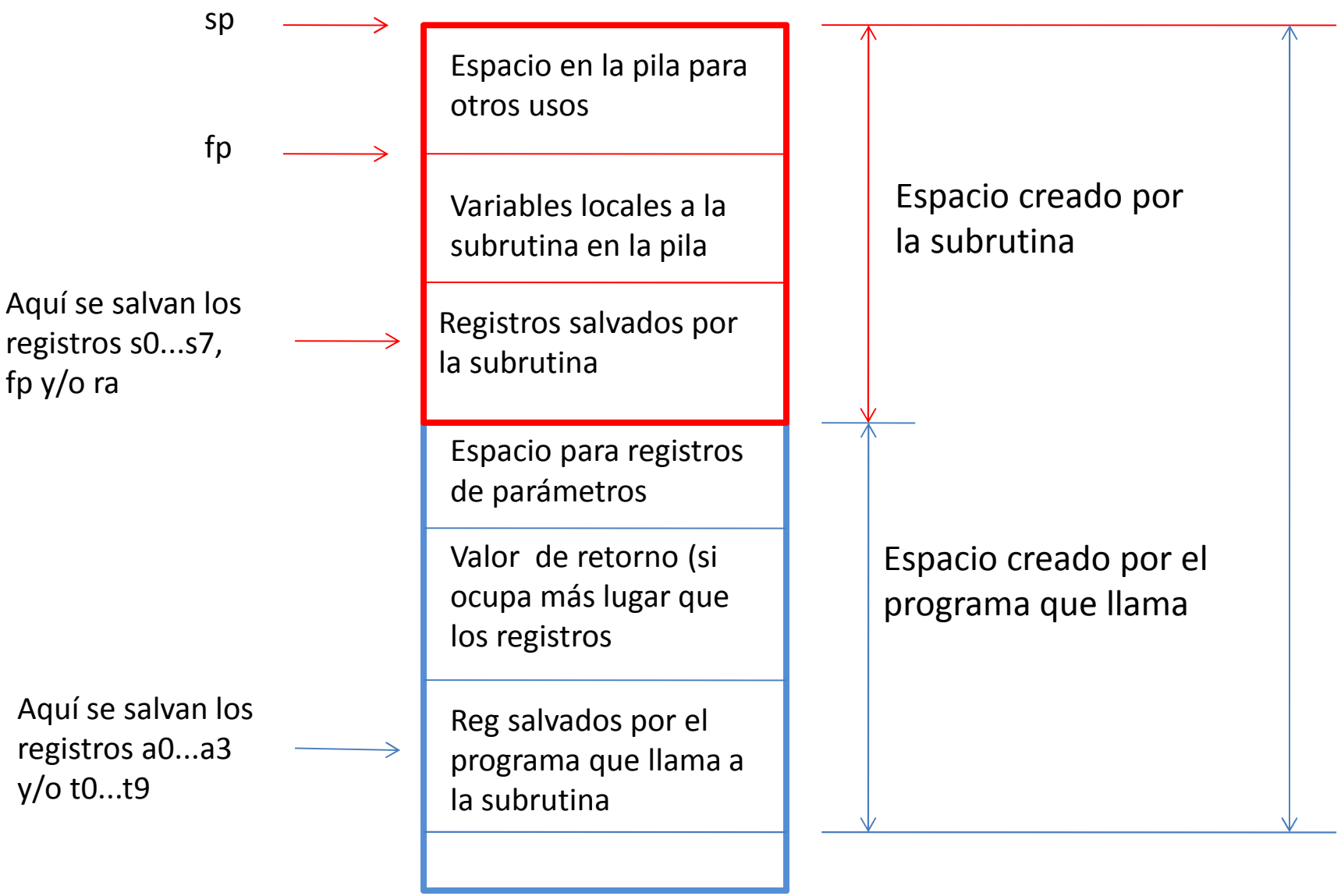
Nombre	Número	Uso
$zero	0	constante 0
$v0–$v1	2–3	resultados, evaluación de expresiones
$a0–$a3	4–7	argumentos
$t0–$t7	8–15	temporales
$s0–$s7	16–23	preservados en llamadas
$t8–$t9	24–25	temporales
$gp	28	puntero global
$sp	29	puntero de pila
$fp	30	puntero de cuadro
$ra	31	dirección de retorno

- Es muy probable que una subrutina necesite más registros que los $a0–$a3 y $v0–$v1.
- Antes de poder usar otros registros, éstos se deben salvar.
- Para ello, se dedica un trozo de memoria, suficientemente grande, como lugar de almacenamiento temporal.
- Esta memoria se llama pila (en inglés, stack).
- Se usa un registro, $sp, para saber en donde comienza el espacio libre. Cada vez que se almacena o se extrae algo, se ajusta (nosotros lo hacemos).
- En general la pila progresa hacia las direcciones bajas de memoria.
- Las operaciones de almacenamiento y extracción las debemos hacer nosotros

- Para escribir una subrutina en ensamblador de MIPS en principio bastaría con conocer lo visto anteriormente:
  - Las instrucciones de llamada a subrutina.
  - La instrucción de retorno de la subrutina.
  - El convenio de uso de los registros.

- Casos que pueden complicar la situación:
  - Llamadas a subrutinas anidadas.
  - Más de 4 parámetros.
  - Muchas variables locales.

- El convenio de los compiladores para MIPS indica que en las llamadas a subrutina se almacena en la pila un bloque de activación (BA) que contiene:
  - Argumentos.
  - Dirección de retorno.
  - Puntero de marco de pila.
  - Registros salvados **s0 ... s7.**
  - Datos locales.
  - Otros: valores temporales para calcular expresiones, etc.
- Marco de pila (stack frame): fragmento del BA creado por la subrutina.
  - Puntero de marco (frame pointer): puntero que apunta al marco de pila.
  - Marco de pila: variables locales + registros salvaguardados por la subrutina.
  - El marco de pila es creado por la propia subrutina nada más comenzar, y es destruido por la propia subrutina justo antes de retornar.
- El tamaño del marco de pila en octetos debe ser múltiplo de 8.


sp
fp
Espacio en la pila para otros usos
Variables locales a la subrutina en la pila
Aquí se salvan los registros s0...s7, fp y/o ra
Registros salvados por la subrutina
Espacio para registros de parámetros
Valor de retorno (si ocupa más lugar que los registros
Aquí se salvan los registros a0...a3 y/o t0...t9
Reg salvados por el programa que llama a la subrutina
Espacio creado por la subrutina
Espacio creado por el programa que llama

Ej. suma de dos números de 64 bits

```
.code
 LD     $a0, NUM1(R0)
 LD     $a1, NUM2(R0)
 DADDI  $sp, $zero, 0x300
 DADDI  $sp, $sp, -16
 SD     $a0, 8($sp)
 SD     $a1, 0($sp)
 JAL    SUMA
 SD     $v1, C(R0)
 HALT
```

```
SUMA: DADD $fp, Szero, $sp
      LD  $t1, 0($fp)
      LD  $t2, 8($fp)
      DADD $v1, $t1, $t2
      JR  $ra
```

Rutinas anidadas

- Si hay rutinas anidadas, como la dirección de retorno se almacena en un registro ($ra) (r31), antes de invocar a la segunda subrutina hay que salvar la “vieja” dirección de retorno
- daddi $sp, Ssp, -8
- sd $ra, 0($sp)