Estructura de Computadores

Tema 3. Traducción de Programas

Subrutinas multinivel

- Subrutinas que llaman a otras subrutinas
- Contexto de una subrutina:
  - Parámetros (`$a0-$a3`)
  - Dirección de retorno (`$ra`)
  - Puntero de pila (`$sp`)
  - Cálculos intermedios
- Problema
  - ¿Cómo preservamos el contexto de la rutina que nos ha llamado?
  - ¿Cómo sabemos qué registros podemos modificar?

Subrutinas multinivel

```
funcA:
   addu $t1, $a0, $a1
   move $a0, $a2
   jal funcC
   addu $v0, $v0, $t1

funcB:
   addiu $a0, $a0, 5
   jal funcC
   subu $v0, $v0, $a1

funcC:
   ...
```

Salvar y restaurar registros

- Solución trivial e ineficiente
  - Garantizar que todos los registros tienen el mismo estado que tenían al invocar la subrutina
  - Obliga a guardar en el bloque de activación (pila) todos los registros que vamos a modificar en la subrutina

Salvar y restaurar registros

- Solución trivial e ineficiente
  - Garantizar que todos los registros tienen el mismo estado que tenían al invocar la subrutina
  - Obliga a guardar en el bloque de activación (pila) todos los registros que vamos a modificar en la subrutina
- Solución del ABI de MIPS
  - Dividir los registros en temporales y seguros

Temporales	Seguros
$t0-$t9	$s0-$s7
$v0-$v1	$sp
$a0-$a3	$ra

Cuando una subrutina termina, ha de dejar los registros seguros en el mismo estado que tenían cuando se invocó

Salvar y restaurar registros

- El ABI permite presevar el contexto salvando en la pila el mínimo número de registros
- Requiere dos pasos:
  1. Determinar los registros seguros
     - Identificar qué datos almacenados en registros se generan ANTES de una llamada a subrutina y se utilizan DESPUÉS de la llamada
  2. Salvar y restaurar los registros seguros
     - Salvar el valor anterior de los registros seguros en el bloque de activación (pila) al inicio de la subrutina
     - Restaurar el valor de los registros seguros al final de la subrutina

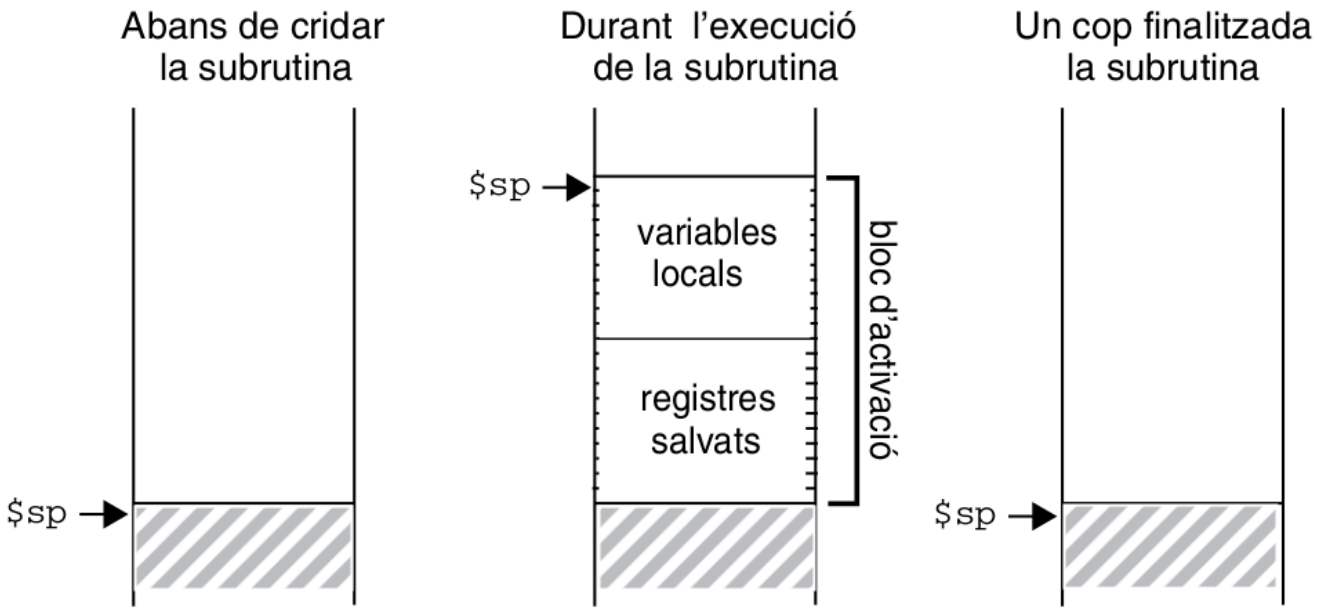
Estructura del bloque de activación

- El bloque de activación está ubicado en la pila e incluye la siguiente información:
  - Variables locales
    - De tipo estructurado (vectores, matrices...)
    - Escalares si se les aplica el operador unario `&`
  - Valores iniciales de los registros seguros
    - Solo se guardan los que se modifican durante la subrutina
    - Se asigna un registro seguro a cada dato almacenado en un registro temporal que debe "sobrevivir" a una llamada a subrutina

Estructura del bloque de activación

- El bloque de activación debe respetar las siguientes normas:
  - **Posición**
    - Las variables locales van al principio, seguidas de los registros seguros
  - **Ordenación**
    - Las variables locales se ubican en el orden en que se declaran en el código
  - **Alineación**
    - Las variables locales deben respetar las normas de alineación
    - Los registros seguros deben ir alineados a direcciones múltiplo de 4
    - El tamaño del bloque de activación ha de ser múltiplo de 4

Estructura del bloque de activación



Ejercicio

- Traduce a MIPS la subrutina `multi` y dibuja el bloque de activación.

```
int multi(int a, int b, int c) {
  int d, e;
  d = a + b;
  e = mcm(c, d);
  return c + d + e;
}
```

Ejercicio

- Traduce a MIPS la subrutina `exemple` y dibuja el bloque de activación.

```
int f(int m, int *n);
int g(char *y, char *z);

char exemple(int a, int b, int c) {
  int d, e, q;
  char v[18], w[20];
  d = a + b;
  e = f(d, &q) + g(v, w);
  return v[e + q] + w[d + c];
}
```