CRITERIO A SEGUIR EN SUBRUTINAS NO HOJAS

Las subrutinas no hojas, es decir, las subrutinas que llaman a otras subrutinas, deben seguir ciertas precauciones dado que el registro ra, donde se almacena la dirección a la que regresar, es sobreescrito por la subrutina a la que llama.

Vimos una solución en clase en la que cada vez que llamábamos a subrutinas salvábamos en la pila todo el entorno. Veremos una alternativa más cómoda.

Recordemos que, según la ABI:

- s0 a s11, son registros PRESERVADOS, esto es, no pueden cambiar de valor cuando se ejecuta una subrutina. Si se utilizan en una subrutina, deben preservarse en la pila (lo hace el "callee", esto es, se hace dentro la subrutina que es llamada)

- t0 a t7 son registros NO PRESERVADOS, esto es, hay que dar por hecho de que pueden cambiar de valor cuando se llama a una subrutina. Si se utilizan en el programa principal o en otra subrutina, deben preservarse en la pila (lo hace el "caller", es decir, se hace fuera de la subrutina a la que se llama)

- el puntero de pila debe estar alineado en múltiplos de 16 bytes

Es decir, tanto el programa principal como las subrutinas (hojas y no-hojas), pueden usar cualquier tipo de registros. Sin embargo, algunos criterios generan más interacciones con la pila que otros:

Posibles alternativas: (uso de registros)

Solución 1a:
- programa principal: registros 's'(si usara temporales, tendría que preservarlos en la pila antes de llamar a una subrutina)
- subrutina no-hoja: registros 't' (preservando en la pila antes de llamar a otras subrutinas)
- subrutina hoja: registros 't'

Solución 1b:
- igual en esencia que la 1a pero preservando ra al principio de la subrutina no-hoja y recuperándolo solo al final

Solución 2: (nuestra propuesta)
- programa principal: registros 's' (si usara temporales, tendría que preservarlos en la pila antes de llamar a una subrutina)
- subrutina no-hoja: registros 's', preservando en la pila ra y registros 's' al principio de la subrutina y recuperándolos al final. (si usara temporales, tendría que preservar en la pila antes de llamar a una subrutina).
- subrutina hoja: registros 't'

de esta forma, las subrutinas hojas sólo necesitan un preámbulo donde apilen ra y los registros salvados (si) y un epílogo antes del ret donde recuperen los registros que se habían guardado.

En resumen:
- programa principal: registros s
- subrutinas hoja: registros t
- subrutinas no hoja: registros s (guardando al principio ra y los registros s que use, y sacando al final los registros).

SUBRUTINA MUL (multiplica números sin signo)

Argumentos: a0, a1 (factores)
Resultado: a0 (producto)

Sumamos a1 tantas veces como indica a0

```
mul:
        mv t0,zero
loopm:
        beq a0,zero,endm
        add t0,to,a1
        addi a0,a0,-1
        j loopm
endm:
        mv a0,t0
        ret
```

SUBRUTINA CUBO (cubo de números sin signo)

Argumento: a0 (factor)
Resultado: a0 (producto)

Veremos las tres opciones mencionadas (1a, 1b y 2). La más cómoda es la 2.

Opción 1a.

```
cubo:
        mv t0,a0        #copio factor en t0
        mv a1,a0
        addi sp,sp,-16  #reservo espacio en la pila (mínimo 16 bytes)
        sw ra,12(sp)    #apilo dir retorno
        sw t0,8(sp)     #apilo factor t0 (ya que llamo a otra subrutina)
        call Mul
        lw ra,12(sp)    #recupero dir retorno
        lw t0,8(sp)     #recupero factor
        addi sp,sp,16    #libero pila
        mv a1,t0
        addi sp,sp,-16  #reservo espacio en la pila (mínimo 16 bytes)
        sw ra,12(sp)    #apilo dir retorno
        sw t0,8(sp)     #apilo factor
        call Mul
        lw ra,12(sp)    #recupero dir retorno
        lw t0,8(sp)     #recupero factor
        addi sp,sp,16   #libero pila
        ret
```

Opción 1b.

```
cubo:
        mv t0,a0        #copio factor en t0
        mv a1,a0
        addi sp,sp,-16  #reservo espacio en la pila (mínimo 16 bytes)
        sw ra,12(sp)    #apilo dir retorno
        sw t0,8(sp)     #apilo factor t0 (ya que llamo a otra subrutina)
        call Mul
        lw t0,8(sp)     #recupero factor
        mv a1,t0
        sw t0,8(sp)     #apilo factor
        call Mul
        lw ra,12(sp)    #recupero dir retorno
        lw t0,8(sp)     #recupero factor
        addi sp,sp,16   #libero pila
        ret
```

Opción 2.

```
cubo:
        addi sp,sp,-16  #reservo espacio en la pila (mínimo 16 bytes)
        sw ra,12(sp)    #apilo dir retorno
        sw s0,8(sp)     #apilo el registro saved que voy a usar

        mv s0,a0        #copio factor en s0
        mv a1,a0
        call Mul
        mv a1,s0
        call Mul

        lw ra,12(sp)    #recupero dir retorno
        lw s0,8(sp)     #recupero registro saved
        addi sp,sp,16   #libero pila
        ret
```