Convención de llamada a procedimientos y funciones

Default C Calling Convention (O32, para MIPS)

- Es una convención (acuerdo o práctica) que determina como las subrutinas reciben los parámetros desde su llamador y como devuelven el resultado.
- Indica como se deben utilizar los registros y la pila.
- Permite la integración de funciones y procedimientos desarrollados por distintos agentes (programadores o compiladores), de manera sencilla.
- Implementada por *software* y asistida por el *hardware*.

(en realidad la presentada aquí no es la convención *032* completa, pero es compatible)

Asistencia del hardware

- Se llama a una función o procedimiento con `jal etiqueta_función`:
  - En el registro `RA` se guarda `PC + 8` (la dirección de la próxima instrucción del código ensamblador).
  - El `PC` toma el valor de la dirección de la etiqueta.
  - Para retornar al llamador, basta con ejecutar la instrucción `jr $RA`.

Consideraciones que debe tener el programador/compilador

- Los registros `T0` a `T7` son para datos temporales y volátiles:
  - La función llamada puede utilizarlos libremente. Esto tiene como consecuencia que, luego de llamar a una función, pueden contener un valor distinto al que tenían antes.
  - La función llamadora, si desea conservar el valor, debe salvaguardarlo en algún lugar (probablemente en su espacio de pila).

Consideraciones que debe tener el programador/compilador

- Los registros S0 a S7 son para datos que deben ser resguardados:
  - Si la función llamada desea utilizar alguno de estos registros, antes de modificarlos debe resguardar el valor para restaurarlo antes de retornar al llamador.
  - El llamador puede estar seguro que los valores guardados en los registro serán los mismos luego de realizar una llamada a un procedimiento.

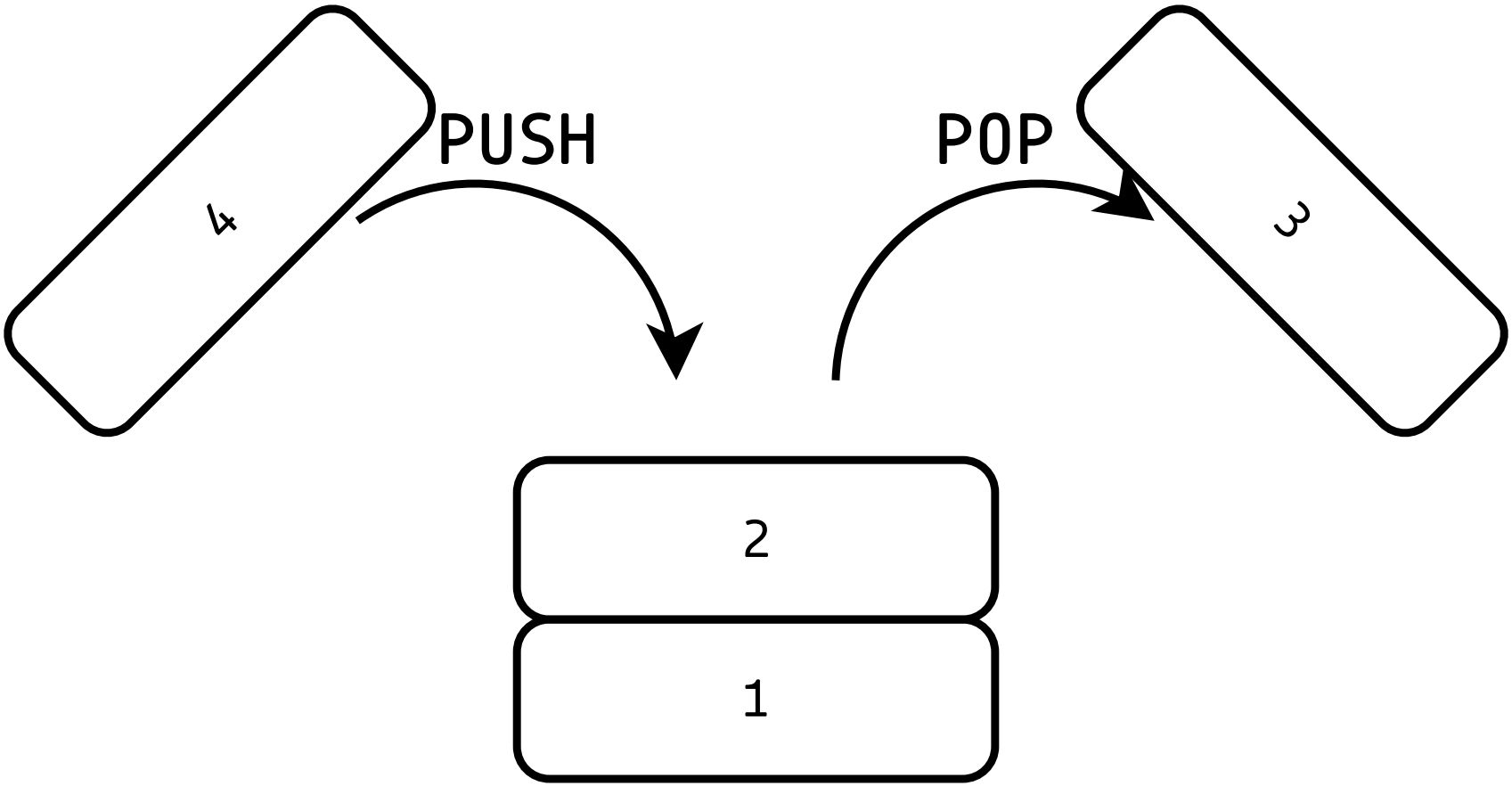
Consideraciones que debe tener el programador/compilador

- Antes de llamar a una función con la instrucción `jal` se debe resguardar el valor contenido en el registro `RA`.

Consideraciones que debe tener el programador/compilador

- Los parámetros de la función de pasan en los registros `A0` a `A3`, y si se necesita pasar más parámetros, o estos son muy grandes para los registros, se deben guardar en la pila del llamador.
- La función guarda los valores a retornar en `V0` y `V1`.

Gestión de la pila en MIPS



Gestión de la pila en MIPS

- Se realiza por software.
- El registro `SP` apunta al tope de la pila, y el registro `FP` apunta a la dirección del tope de la pila del proceso anterior.
- Nunca se usa la palabra del tope de la pila.
- El tamaño mínimo de la pila es de 24 bytes, ya que se debe haber suficiente espacio para que el procedimiento pueda resguardar (si lo necesita) los valores de los registros `A0` a `A3`, `RA` y el tope de pila.
- La pila crece a direcciones de memorias más bajas, y siempre apunta a direcciones múltiplo de 8:
  - Si el tope de pila es 1040_{10}, y se reserva un espacio de pila de 24 bytes, el nuevo tope de pila sera 1016_{10}.

Gestión de la pila en MIPS

- El tamaño de la pila de un proceso se puede ampliar dinámicamente para resguardar valores de registros u otros datos.
- Cada función debe crear su propio espacio de pila, y debe eliminarlo antes de retornar al llamador.

Código fuente C

```
1  int fun(int a, int b, int c)
2  {
3          return 42;
4  }
5
6  int main()
7  {
8          return fun(1, 2, 3);
9  }
```

Código del llamador

```
 1  main:
 2       # Crear bloque de pila
 3       addiu    $sp,$sp,-32
 4       sw       $ra,28($sp)
 5       sw       $fp,24($sp)
 6       move     $fp,$sp
 7       # Código de MAIN
 8       li       $a0,1
 9       li       $a1,2
10       li       $a2,3
11       jal      fun
12       nop
13       # Destruir bloque de pila
14       move     $sp,$fp
15       lw       $ra,28($sp)
16       lw       $fp,24($sp)
17       addiu    $sp,$sp,32
18       jr       $ra
```

Código del llamado

```
 1   fun:
 2       # Crear bloque de pila
 3       addiu   $sp,$sp,-24
 4       sw      $ra,24($sp)
 5       sw      $fp,20($sp)
 6       move    $fp,$sp
 7       # Código de FUN
 8       li      $v0, 42
 9       # Destruir bloque de pila
10       move    $sp,$fp
11       lw      $ra,24($sp)
12       lw      $fp,20($sp)
13       addiu   $sp,$sp,24
14       jr      $ra
```